John Krogstie
Stefanie Rinderle-Ma
Gerti Kappel
Hend*erik* A. Proper (Eds.)

LNCS 15702

Advanced Information Systems Engineering

37th International Conference, CAiSE 2025
Vienna, Austria, June 16–20, 2025
Proceedings, Part II

2 Part II

CAiSE '25

Springer

Lecture Notes in Computer Science **15702**

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.

John Krogstie · Stefanie Rinderle-Ma ·
Gerti Kappel · Hend*erik* A. Proper
Editors

Advanced Information Systems Engineering

37th International Conference, CAiSE 2025
Vienna, Austria, June 16–20, 2025
Proceedings, Part II

Springer

Editors
John Krogstie
NTNU – Norwegian University of Science and Technology
Trondheim, Norway

Gerti Kappel
TU Wien
Vienna, Austria

Stefanie Rinderle-Ma
Technische Universität München
Garching bei München, Germany

Hend*erik* A. Proper
TU Wien
Vienna, Austria

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-031-94573-1 ISBN 978-3-031-94571-7 (eBook)
https://doi.org/10.1007/978-3-031-94571-7

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume (of two) of the Lecture Notes in Computer Science series contains the second part of the proceedings of the 37th International Conference on Advanced Information Systems Engineering – CAiSE 2025, held in Vienna, Austria, during June 16–20. The CAiSE conference series was started in Sweden, and its first two editions were organized in Stockholm (1989 and 1990). Since then, it has been held in 21 countries, mostly in Europe but also in locations on other continents (Canada, Australia, and Tunisia). Despite its roots in Europe, soon after its inception, CAiSE became the flagship international conference on information systems engineering, attracting some of the most original and scientifically rigorous articles in the field. Furthermore, it has always been an attractive event for early-career researchers as well as practitioners. After almost four decades of continuous organization, CAiSE continues to attract articles that address fundamental and timeless problems in the field.

As a conference attuned to new theoretical and societal challenges, each year a new focal topic is chosen as the theme of the conference. CAiSE 2025 was organized with a special emphasis on the theme of **Bridging Silos**. Engineering real-world information systems requires a coherent design, encompassing human, organizational, economic, societal, and technological aspects. Information systems are utilized in increasingly diverse contexts such as business process management, geographical information systems, and digital twins. At the same time, the discipline continually evolves with trends in data science, machine learning, process mining, blockchain, mobile computing, sustainability, new regulations, cyber warfare, and military conflicts all influencing its development. Each application context and emerging trend can lead to specializations within information systems engineering, necessitating various research traditions and needs. While beneficial, these specializations risk creating silos within the field of information systems engineering. Thus, the CAiSE conference, the premier event for this discipline, aims to prevent such fragmentation.

CAiSE 2025 attracted a broad spectrum of classical and modern topics in Information Systems Engineering, including several papers directly focused on this year's theme. In total, we received 286 abstract submissions, 248 of which materialized as full submissions. We then desk rejected 19 papers due to issues related to paper formatting and length but, mostly, papers that we independently judged to be out of the scope of the conference. The remaining papers were then sent for independent reviews, each being allocated to two members of the international Program Committee. After the first phase of reviews, papers that received two negative informative reviews were not admitted to the second phase of reviews – 120 papers were rejected in this first phase. In the second phase of reviews, the remaining 110 papers were allocated to a third reviewer, who was necessarily a member of the conference Program Board (a body of senior scholars from the community, representing different research areas and geographical locations, the Program Committee chairs of the previous and succeeding editions of CAiSE, as well as members of the conference Steering Committee). After the second phase of reviews, the

submissions for which the three reviewers converged to a unanimous negative judgement were rejected (37 submissions). The remaining submissions (73) were then assigned to yet another member of the Program Board for discussions and meta-reviews. In an intense discussion period, these latter members of the Program Board moderated the discussions between the reviewers of the paper (the two Program Committee members and the reviewing Program Board Member). They then summarized and explained the recommendations for each submission in a meta-review consolidating the outcome of those discussions. As an outcome of this discussion and meta-reviewing process, we were able to decide on 29 papers: 16 which were directly accepted (these are papers that received three positive and informative reviews plus a recommendation from the discussant/meta-reviewer to accept without conditions); 13 papers were directly rejected (these were papers with three negative and informative final reviews plus a recommendation from the discussant/meta-reviewer to reject without an invitation to the Forum). After these different phases of reviewing, meta-reviewing, and online discussions, 44 papers (18% of the original submissions) were discussed in a face-to-face meeting organized in February 2025 at the Technische Universität Wien (TU Wien) in Austria. In that meeting, besides the Program Chairs and Program Board members, participated representatives of the General Chairs of the conference, the Forum Chairs, and the Workshop Chairs. After two days of intensive and informative discussions, 19 submissions were directly accepted based on the novelty of their work and the depth of contributions to the advancement of the state of the art. Furthermore, 17 submissions were accepted conditionally, subject to specific improvements the co-authors were able to address in a minor revision checked by a gatekeeper – in the vast majority of cases, a member of the Program Board who was involved in the reviewing or meta-reviewing of the paper. The outcome is presented in this volume, comprising 35 scientifically rigorous and innovative papers – a 14% acceptance rate. In addition, we recommended 16 submissions for consideration by the CAiSE Forum.

As a tradition at CAiSE, the conference program started with workshops as well as two traditional satellite conferences (EMMSAD and BPMDS). The main conference included sessions involving Research Papers, Keynotes, Tutorials, the Forum – for discussion triggering visionary work – the Doctoral Consortium, Project Exhibitions, as well as Journal-First papers. In addition, the program included a panel discussion. This volume collects the Research Papers of the conference, as well as companion abstracts for the keynote talks and the tutorials. The 35 accepted papers together with 9 accepted Journal-First papers were grouped into the following fourteen topical sessions: Modelling with LLM; Security; Sustainability; Chatbots and social networks; Process monitoring; IS development and usage; Pre-processing and forecasting; Comprehension, explanation and recommendation; Process discovery; Systems architecture and privacy; Conformance checking; Cloud systems; Extended process modelling; Ontologies and knowledge graphs.

The three invited keynote presentations were: *"Engineering the (Information) System of Life"* by Oscar Pastor from the Universidad Politécnica de Valencia; *"Reflections on the Early History of Information Systems Development Methodologies"* by Rudy Hirschheim, from Louisiana State University; and *"Business Transformation Management in the age of AI and agents"* by Gero Decker, from SAP Berlin. On behalf of the

information systems engineering community, we would like to thank our keynote speakers for their time dedicated to preparing profound, timely, and insightful talks. After an open call for tutorials, we accepted the following two proposals: *"Empowering Development and Use of DSMLs with the FMML^{X} and the XModeler^{ML}"* by Ulrich Frank from the University of Duisburg-Essen; and *"Addressing Vulnerabilities in Information Systems Engineering"* by Avi Shaked from the University of Oxford.

As the editors of both CAiSE 2025 volumes, we would like to express our immense gratitude to:all authors who chose CAiSE as a forum for submitting their research contributions; the members of the Program Committee for their professional and timely work; the members of the Program Board for their dedication and expertise in helping to shape an essential part of the final program of the conference; all the chairs of the conference for selecting an interesting program that complements the contributions collected here; the Organization Committee for professionally taking care of all the multiple details of running an international and sufficiently large conference like CAiSE, but also for bringing the conference to the fantastic city of Vienna in Austria.

John Krogstie
Stefanie Rinderle-Ma
Gerti Kappel
Hend*erik* A. Proper

April 2025

Organization

General Chairs

Gerti Kappel	TU Wien, Austria
Hend*erik* A. Proper	TU Wien, Austria

Program Committee Chairs

John Krogstie	Norwegian University of Science and Technology, Norway
Stefanie Rinderle-Ma	Technical University of Munich, Germany

Local Organization Chairs

Dominik Bork	TU Wien, Austria
Angela Edlinger	TU Wien, Austria

Forum Chairs

Kristina Rosenthal	Hochschule Niederrhein University of Applied Sciences, Germany
Luise Pufahl	Technical University of Munich, Germany

Workshop Chairs

Janis Grabis	Riga Technical University, Latvia
Yves Wautelet	KU Leuven, Belgium

Journal-First Chairs

Barbara Weber	University of St. Gallen, Switzerland
Janis Stirna	Stockholm University, Sweden

Tutorial and Panel Chairs

Henrik Leopold	Kühne Logistics University, Germany
Tiago Prince Sales	University of Twente, The Netherlands

Doctoral Consortium Chairs

Selmin Nurcan	Université Paris 1 Panthéon-Sorbonne, France
Sergio España	Universitat Politècnica de València, Spain

PhD Award Chairs

Giancarlo Guizzardi	University of Twente, The Netherlands
Pnina Soffer	University of Haifa, Israel

Research Projects Exhibition Chairs

Claudenir Fonseca	University of Twente, The Netherlands
Mattia Fumagalli	Free University of Bozen-Bolzano, Italy

Proceedings Chair

Marianne Schnellmann	TU Wien, Austria

Web Chair

Aleksandar Gavric	TU Wien, Austria

Publicity and Social Media Chairs

Asif Gil	University of Technology Sydney, Australia
Jaap Gordijn	Vrije Universiteit Amsterdam, The Netherlands
João Paulo Almeida	Federal University of Espírito Santo, Brazil
Roman Lukyanenko	University of Virginia, USA

Program Board

Arnon Sturm	Ben-Gurion University of the Negev, Israel
Barbara Pernici	Politecnico di Milano, Italy
Barbara Weber	University of St. Gallen, Switzerland
Flavia Santoro	Universidade do Estado do Rio de Janeiro, Brazil
Giancarlo Guizzardi	University of Twente, The Netherlands
Hajo Reijers	Utrecht University, The Netherlands
Iris Reinhartz-Berger	University of Haifa, Israel
Jan Mendling	Humboldt-Universität zu Berlin, Germany
Janis Stirna	Stockholm University, Sweden
Jelena Zdravkovic	Stockholm University, Sweden
João Paulo Almeida	Federal University of Espírito Santo, Brazil
Johann Eder	University of Klagenfurt, Austria
Jolita Ralyté	University of Geneva, Switzerland
Lidia Fuentes	Universidad de Málaga, Spain
Manfred Reichert	University of Ulm, Germany
Maribel Yasmina Santos	University of Minho, Portugal
Marta Indulska	University of Queensland, Australia
Mathias Weske	University of Potsdam, Germany
Monique Snoeck	KU Leuven, Belgium
Oscar Pastor	Universitat Politècnica de València, Spain
Paolo Giorgini	University of Trento, Italy
Pierluigi Plebani	Politecnico di Milano, Italy
Pnina Soffer	University of Haifa, Israel
Renata Guizzardi	University of Twente, The Netherlands
Selmin Nurcan	Paris 1 Panthéon-Sorbonne University, France
Shazia Sadiq	University of Queensland, Australia
Wolfgang Maaß	Saarland University, Germany
Xavier Franch	Universitat Politècnica de Catalunya, Spain

Program Committee

Adela Del Río Ortega	University of Seville, Spain
Alejandro Maté	University of Alicante, Spain
Alessandro Gianola	Universidade de Lisboa, Portugal
Andrea Burattin	Technical University of Denmark, Denmark
Andrea Marrella	Sapienza University of Rome, Italy
Andreas L. Opdahl	University of Bergen, Norway
Andrey Rivkin	Technical University of Denmark, Denmark
Anna Bernasconi	Politecnico di Milano, Italy

Antonio Ruiz-Cortés	University of Seville, Spain
Arik Senderovich	York University, Canada
Artem Polyvyanyy	University of Melbourne, Australia
Ashwin Viswanathan Kannan	Oklahoma State University, USA
Bernhard Thalheim	Christian-Albrechts-Universität zu Kiel, Germany
Boualem Benatallah	UNSW Australia, Australia
Camille Salinesi	Paris 1 Panthéon-Sorbonne University, France
Chiara Di Francescomarino	DISI - University of Trento, Italy
Claudenir Fonseca	University of Twente, The Netherlands
Claudio Di Ciccio	Utrecht University, The Netherlands
Corentin Burnay	University of Namur, Belgium
Cristina Cabanillas	Universidad de Sevilla, Spain
Devis Bianchini	University of Brescia, Italy
Dominik Bork	TU Wien, Austria
Emanuele Laurenzi	University of Applied Sciences and Arts Northwestern Switzerland, Switzerland
Ernest Teniente	Universitat Politècnica de Catalunya, Spain
Fabrizio Maria Maggi	Free University of Bozen-Bolzano, Italy
Felix Mannhardt	Eindhoven University of Technology, The Netherlands
Fernanda Baião	PUC-Rio, Brazil
Georg Grossmann	University of South Australia, Australia
Han van der Aa	University of Vienna, Austria
Hans-Georg Fill	University of Fribourg, Switzerland
Haralambos Mouratidis	University of Essex, UK
Henrik Leopold	Kühne Logistics University, Germany
Ingo Weber	Technical University of Munich, Germany
Istvan David	McMaster University, Canada
Jaap Gordijn	Vrije Universiteit Amsterdam, The Netherlands
Janis Grabis	Riga Technical University, Latvia
Jennifer Horkoff	Chalmers University of Technology, Sweden
Johannes De Smedt	KU Leuven, Belgium
João Araújo	Universidade NOVA de Lisboa, Portugal
Julio Cesar Leite	UFBA, Brazil
Kate Revoredo	Humboldt-Universität zu Berlin, Germany
Kurt Sandkuhl	University of Rostock, Germany
Leonardo Montecchi	NTNU IDI, Norway
Luca Piras	Middlesex University, UK
Luiz Olavo Bonino Da Silva Santos	University of Twente, The Netherlands
Luís Ferreira Pires	University of Twente, The Netherlands
Manfred Jeusfeld	University of Skövde, Sweden

Manuel Wimmer	Johannes Kepler University Linz, Austria
Marcela Ruiz	Zurich University of Applied Sciences, Switzerland
Marcelo Fantinato	University of São Paulo, Brazil
Maria Teresa Gómez López	University of Seville, Spain
Marite Kirikova	Riga Technical University, Latvia
Marlon Dumas	University of Tartu, Estonia
Massimiliano Leoni	University of Padua, Italy
Massimo Mecella	Sapienza University of Rome, Italy
Matthias Weidlich	Humboldt-Universität zu Berlin, Germany
Mattia Fumagalli	Free University of Bozen-Bolzano, Italy
Minseok Song	Pohang University of Science and Technology, South Korea
Oscar Diaz	University of the Basque Country, Spain
Palash Bera	Saint Louis University, USA
Paolo Giorgini	University of Trento, Italy
Paul Grefen	Eindhoven University, The Netherlands
Pedro Paulo F. Barcelos	University of Twente, The Netherlands
Peter Fettke	DFKI, Germany
Raimundas Matulevicius	University of Tartu, Estonia
Sagar Sunkle	Tata Consultancy Services, India
Sander Leemans	RWTH Aachen University, Germany
Sareh Sadeghianasl	Queensland University of Technology, Australia
Schahram Dustdar	TU Wien, Austria
Sebastian Link	University of Auckland, New Zealand
Seppe Van den Broucke	KU Leuven, Belgium
Shang Gao	Ørebro University, Sweden
Simon Hacks	Stockholm University, Sweden
Sobah Abbas Petersen	NTNU IDI, Norway
Stijn Hoppenbrouwers	HAN University of Applied Sciences, The Netherlands
Tiago Prince Sales	University of Twente, The Netherlands
Tong Li	Beijing University of Technology, China
Tony Wasserman	Carnegie Mellon Silicon Valley, USA
Veda Storey	Georgia State University, USA
Veruska Zamborlini	Universidade Federal do Espírito Santo, Brazil

Steering Committee Chairs

John Krogstie	Norwegian University of Science and Technology, Norway
Monique Snoeck	KU Leuven, Belgium
Xavier Franch	Universitat Politècnica de Catalunya, Spain

Advisory Board

Arne Sølvberg	Norwegian University of Science and Technology, Norway
Barbara Pernici	Politecnico di Milano, Italy
Colette Rolland	University of Paris 1 Pantheon-Sorbonne, France
Johann Eder	University of Klagenfurt, Austria
Oscar Pastor	Universitat Politècnica de València, Spain

Steering Committee

Barbara Weber	University of St. Gallen, Switzerland
Camille Salinesi	University of Paris 1 Pantheon-Sorbonne, France
Carlos Cetina	Universidad San Jorge, Spain
Eric Yu	University of Toronto, Canada
Ernest Teniente	Universitat Politècnica de Catalunya, Spain
Flavia Santoro	UERJ - Universidade do Estado do Rio de Janeiro, Brazil
Frederik Gailly	Ghent University, Belgium
Geert Poels	Ghent University, Belgium
Giancarlo Guizzardi	University of Twente, The Netherlands
Hajo Reijers	Utrecht University, The Netherlands
Haralambos Mouratidis	University of Essex, UK
Iris Reinhartz-Berger	University of Haifa, Israel
Janis Stirna	Stockholm University, Sweden
Jelena Zdravkovic	Stockholm University, Sweden
Marcello La Rosa	University of Melbourne, Australia
Marta Indulska	University of Queensland, Australia
Massimo Mecella	Sapienza University of Rome, Italy
Paolo Giorgini	University of Trento, Italy
Pnina Soffer	University of Haifa, Israel
Shazia Sadiq	University of Queensland, Australia

Forum Program Committee

Abel Armas Cervantes	University of Melbourne, Australia
Agnes Koschmider	University of Bayreuth, Germany
Andrea Marrella	Sapienza University of Rome, Italy
Ben Roelens	Open University of the Netherlands, The Netherlands
Christophe Feltus	Institute of Science and Technology, Ireland
Cinzia Cappiello	Politecnico di Milano, Italy
Cristina Cabanillas	Universidad de Sevilla, Spain
Drazen Brdjanin	University of Banja Luka, Bosnia and Herzegovina
Elena Kornyshova	Conservatoire National des Arts et Métiers, France
Evangelia Kavakli	University of the Aegean, Greece
Hans Weigand	Tilburg University, The Netherlands
Hans-Georg Fill	University of Fribourg, Switzerland
Henrik Leopold	Kühne Logistics University, Germany
Irene Vanderfeesten	KU Leuven, Belgium
Janis Grabis	Riga Technical University, Latvia
Janne J. Korhonen	Aalto University, Finland
Jose Ignacio Panach Navarrete	Universitat de València, Spain
Jānis Kampars	Riga Technical University, Latvia
Lawrence Chung	University of Texas at Dallas, USA
Manuel Resinas	University of Seville, Spain
Manuel Wimmer	Johannes Kepler University Linz, Austria
Marite Kirikova	Riga Technical University, Latvia
Martin Henkel	Stockholm University, Sweden
Mattia Salnitri	Politecnico di Milano, Italy
Maya Daneva	University of Twente, The Netherlands
Michael Fellmann	University of Rostock, Institute for Computer Science, Germany
Mohamad Gharib	University of Tartu, Estonia
Oscar Pastor	Universitat Politècnica de València, Spain
Patricia Martin-Rodilla	Instituto de Estudios Gallegos, Spain
Raimundas Matulevicius	University of Tartu, Estonia
Sergio de Cesare	University of Westminster, UK
Simon Hacks	Stockholm University, Sweden
Stefan Strecker	University of Hagen, Germany
Steven Alter	University of San Francisco, USA
Sybren de Kinderen	Eindhoven University of Technology, The Netherlands

Sérgio Guerreiro	University of Lisbon, Portugal
Tony Clark	Aston University, UK
Yves Wautelet	KU Leuven, Belgium

Additional Reviewers

Abasi-Amefon Affia-Jomants
Ada Slupczynski
Adam Banham
Adam Burke
Aleksandar Gavric
Alfonso Marquez-Chamorro
Anne Gutschmidt
Arsalan Ghasemi
Benjamin Nast
Carlos Müller
David Mosquera
Deepika Gopukumar
Diana Malakhova
Dominic Detering
Evellin Cardoso
Gabriel Morais
Gal Engelberg
Henryk Mustroph
Ijeoma Faustina Ekeh
Illr Murturi
Jan Niklas van Detten
José Antonio Parejo Maestre
José María García
José Miguel Horcas Aguilera
Karamjit Kaur
Maider Azanza
Mariia Bakhtina
Maximilian König
Maya Sappelli
Mónica Pinto
Mubashar Iqbal
Pablo Fernandez
Qian Chen
Quim Motger
Rainer Weinreich
Rebecca Morgan
Sandip Kumar Sarkar
Savandi Kalukapuge
Simone Agostinelli
Stefan Klikovits
Thomas Ricardo Pathe
Tian Li
Tom Lichtenstein
Victoria Sonnemans
Vitor Gaboardi dos Santos
Vjatšeslav Antipenko
Wenjun Zhou
Zsolt Kardkovacs

Contents – Part II

Conformance-Checking

Cloud Systems

Extending Process Modelling

Ontologies and Knowledge Graphs

Tutorials

Contents – Part I

Pre-processing and Forecasting

Comprehension, Explanation and Recommendation

How Do Experts Make Sense of Integrated Process Models?

Tianwa Chen[1(✉)], Barbara Weber[2], Graeme Shanks[3], Gianluca Demartini[1], Marta Indulska[4], and Shazia Sadiq[1]

[1] School of Electrical Engineering and Computer Science, The University of Queensland, Brisbane, Australia
tianwa.chen@student.uq.edu.au, g.demartini@uq.edu.au, shazia@eecs.uq.edu.au

[2] Institute of Computer Science, University of St. Gallen, St. Gallen, Switzerland
barbara.weber@unisg.ch

[3] School of Computing and Information Systems, The University of Melbourne, Melbourne, Australia
gshanks@unimelb.edu.au

[4] Business School, The University of Queensland, Brisbane, Australia
m.indulska@business.uq.edu.au

Abstract. A range of integrated modeling approaches have been developed to enable a holistic representation of business process logic together with all relevant business rules. These approaches address inherent problems with separate documentation of business process models and business rules. In this study, we explore how expert process workers make sense of the information provided through such integrated modeling approaches. To do so, we complement verbal protocol analysis with eye-tracking metrics to reveal nuanced user behaviours involved in the main phases of sensemaking, namely information foraging and information processing. By studying expert process workers engaged in tasks based on integrated modeling of business processes and rules, we provide insights that pave the way for a better understanding of sensemaking practices and improved development of business process and business rule integration approaches. Our research underscores the importance of offering personalized support mechanisms that increase the efficacy and efficiency of sensemaking practices for process knowledge workers.

Keywords: Business process modeling · Cued-Retrospective Think-Aloud · Sensemaking · Eye tracking

1 Introduction

The increase in data accessibility and complexity of organisational information systems has given rise to a persistent problem of information silos [43] wherein knowledge workers have to navigate across different systems and forms of information artefacts to perform their tasks. The diversity of information artefacts,

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 3–19, 2025.
https://doi.org/10.1007/978-3-031-94571-7_1

ranging from physical to digital, underscores their crucial role in the organization and manipulation of data and knowledge. For process knowledge workers, including process analysts, process users and process modellers, business process models and business rules are two commonly used artefacts. These two artefacts enable process knowledge workers to represent complex business requirements, as well as implement and improve processes. A notable illustration of the inherent complexity can be observed in the workflows of business analysts and process analysts. For instance, in the scenarios of company mergers and restructurings multiple variants of business processes and business rules need to be consolidated into a single process to eliminate redundancies and create synergies [23]. Even in a business-as-usual environment, if the artefacts are presented separately (i.e., related business rules often may not be part of the business model [50]), they may cause a disconnect in shared understanding, and potentially result in conflicts, inefficiencies and even compliance breaches [44,47,50]. The challenges facing process knowledge workers involve more than just accessing and understanding information from diverse artefacts. Advanced cognitive and analytical skills are needed to ensure comprehensive understanding, including effectively foraging and processing various artefacts from business process models and business rule repositories to achieve specific objectives. These foraging and processing processes involve seeking, filtering, reading, and extracting information and iteratively developing a mental model that serves as a foundation for comprehension and performance [31].

To overcome these challenges, previous studies have underscored the necessity of comprehensively integrating business rules into business process models [44], and various forms of integration have been proposed (e.g. diagrammatic integration, integration through text annotation, and linked rules). However, prior research has primarily focused on novice workers using students as proxy [10,11,44], which offers limited insights into the sensemaking processes that expert knowledge workers engage in. Yet, a deeper understanding of how expert knowledge workers forage and process information in integrated process models is key to adequately supporting the development of new process-oriented tools and systems that can more effectively support decision-making processes.

Drawing on the foundational theories of sensemaking and cognition [31,39], our research seeks to delve into sensemaking practices on how knowledge workers forage and process information in the varied forms of integrated representation of business process models and rules. It aims to unearth the underlying factors that drive these various sensemaking behaviours. To this end, we present the outcomes of empirical research conducted within a controlled laboratory study setting to investigate how process knowledge workers perform tasks based on integrated modelling of business processes and rules (i.e., using text annotation, diagrammatic, and linked rules). Specifically, we investigate experts' sensemaking practices in information foraging and processing phases, and compare the findings with existing literature. By leveraging verbal protocol analysis [16] with eye-tracking metrics, we reveal empirical insights into knowledge process worker behaviours in information foraging and processing phases. This exploration paves the way for offering personalized support mechanisms to process knowledge work-

ers through a deeper understanding of sensemaking practices in various settings and improved development of new process-oriented tools and systems.

In the following sections, we first review the research background of sensemaking and cognition as a lens to study process model understanding. Section 3 introduces our study design and the data analysis methods. Section 4 presents the results and discussion, and finally Sect. 5 summarizes the contribution of the paper, limitations of the study, and future extensions of this work.

2 Literature Review

2.1 Sensemaking and Cognition

Over the decades, sensemaking has been an active area of study in diverse disciplinary backgrounds, from collective organizational contexts (e.g., [22,45]) to individual settings (e.g., [27,35]). More recently, there has been an increased focus on understanding how sensemaking operates in the era of increasingly complex information artefacts (e.g., [9,31,46]). Despite the differences between the proposed models from the literature, all attempts describe the iterative process of individual or collective construction of knowledge. A number of models have been proposed to capture sensemaking as multiple loops [31,46], which consider a fundamental pattern between the interactions of information foraging and processing to schematize the knowledge into a mental model. For example, the Representation Construction Model [31] has two major loops of sensemaking: (1) the information foraging loop, which includes seeking, filtering, reading, and extracting information processes, and (2) the information processing loop, which includes iterative development of representational schemas to provide a basis for understanding and performance.

The individual settings of sensemaking are more relevant to our work, where the focus is on cognitive mechanisms that underpin individual sensemaking. Cognitive constructs of attention and memory have a natural and strong affinity to the two phases in sensemaking models, and cognitive load theory [39] provides proven mechanisms through which these constructs can be operationalized [8,39]. For example, attention and search behaviour have been measured through eye-tracking devices, which can capture data on visual scanning (eye movement) and attention (eye fixations) [12]. This data, in turn, can be used for various behavioural measurements, such as cognitive load, visual association, visual cognition efficiency, and intensity [6,32].

To the best of our knowledge, existing sensemaking studies are focused on qualitative or perceptive measures with limited use of behavioural and performance measures. Additionally, prior work that used quantitative analysis for studying sensemaking processes was limited to novice workers using university students as proxy [10,11,44]. The study of sensemaking practices of novices only provides limited understanding and is not fully reflective of the settings in which these sensemaking processes are undertaken. Hence, we make use of quantitative (eye-tracking devices as observation tool) methods to guide the exploration of qualitative (Cued Retrospective Think-Aloud (CRTA) interviews of expert

users [41]) in a controlled laboratory study. This combination of methods provides novel and objective means to capture and expose sensemaking behaviours and explore the interactive process of how expert knowledge workers forage and process information in various modelling integration approaches.

2.2 Integrated Modeling of Business Processes and Business Rules

Our study considers the specific context of business process and business rule modeling - two complementary approaches for modeling business activities, which have multiple integration methods [21] to improve their individual representational capacity. The integration methods can be categorized into three approaches with distinct format and construction, namely: text annotation, diagrammatic integration, and link integration [11]. Text annotation and link integration both use a textual expression to describe the business rules and connect them with the corresponding section of the process model. Text annotation integration is a way of representing business rules in business process models by adding textual descriptions of rules - e.g. in BPMN, using the BPMN text annotation construct. With link integration, visual links can explicitly connect corresponding rules with the relevant process section. Diagrammatic integration relies on graphical process model construction, such as sequence flows and gateways, to represent business rules in the process model. Each of these methods has strengths and weaknesses, and thus a potential impact on a knowledge worker's understanding of a process [44]. Despite the use of rigorous quantitative analysis in related works [10,11,44], we found that quantitative analysis alone was not sufficient to fully capture the nuanced behaviors involved in sensemaking. This observation underscores the necessity for rich qualitative insights to thoroughly understand sensemaking behaviors.

2.3 Process Model Understanding

Prior research has focused on a variety of factors that affect the understanding of a process, including process model factors [14] and human factors [26]. Process model factors relate to the metrics of the process models, such as modularization [33], block structuredness [48], and complexity. Studying the impacts of these involves investigation of the number of arcs and nodes [26], number of gateways [33], number of events [34], number of loops [14], and number of concurrencies [25], length of the longest path [25], depth of nesting [17], and gateway heterogeneity [25]. Human factors relate to the factors of process model users, such as individual's domain knowledge [40], modeling knowledge [14] and modeling experience [26].

To evaluate cognitive engagement and improve comprehension of process models, the think-aloud approach has served as a pivotal means to understand user interactions and cognitive processes during task completion [5,18,19,49]. The approach has several methods, including Concurrent Think-Aloud (CTA), Retrospective Think-Aloud (RTA), and Cued Retrospective Think-Aloud (CRTA) [41]. The CRTA approach integrates elements of RTA and CTA,

and mitigates the memory-related limitation in RTA and potential disturbance on task completion of CTA. By ensuring participants' natural interaction patterns are preserved during completing the tasks as well as providing them with concrete and task-specific stimuli cues (e.g. screen recording of completing the task), CRTA can facilitate participants to recall their thought process more accurately. In addition, cognitive load and visual cognition have been used as measures of process model understanding [26], with the use of eye tracking technology to capture eye movement and gaze patterns [6,29,30,37]. For example, researchers used eye tracking to investigate the visual cues of colouring and layout with performance in process model understanding [30], and the impact of the task type (local or global) on process model comprehension during information search and inference phases [37], as well as with the use of RTA to explore reading patterns and the strategies in DCR-HR [2].

Upon reviewing the literature, sensemaking emerges as a promising new perspective that has yet to be fully explored in the context of process model understanding. In light of these considerations, our study leveraged eye-tracking data as an observational tool and a cue during interviews, guiding participants to reflect on their recorded gaze behaviors, thus enhancing the recall and depth of their thought processes. This integration facilitates and advances a deeper qualitative exploration into the complex sensemaking behaviors of participants, offering enriched insights into their cognitive engagement with integrated process models, as "the best cues will likely come from the participants themselves" [7].

3 Study Design

In this study, we used a laboratory study method [3] and a between-subject design with purpose-built platforms. To capture the insights of sensemaking behaviors, we first collected eye-tracking data while participants performed the laboratory study. We did this to develop a "cue" to be used in the main method reported on in this paper—the cued retrospective think-aloud method (see Sect. 2.1), which involved using a semi-structured protocol for each expert. Cued with eye gaze movement recordings, CRTA enabled the participants to verbalize and explain their sensemaking practices and strategies during information foraging and processing. This approach facilitated an in-depth qualitative analysis of the participants' information needs and intentions, as well as the challenges and difficulties they encountered, providing valuable insights into their cognitive processes.

3.1 Participants

Our study is specifically focused on experts. All participants in our study were academic researchers with prior experience as practitioners in business process management. They were from the information systems and computer science disciplines in two universities and a research institute. They were required to have both prior work experience as practitioners and research experience using BPMN of over four years. In line with acceptable numbers of participants in qualitative

studies [19,24,28], 15 experts participated in this study, with 5 participants engaging in each integration approach.

3.2 Laboratory Study Materials and Procedure

While the main focus of this paper is the rich qualitative data offered by CRTA, the laboratory study data also consists of a pre-study questionnaire, eye tracking data, task performance and log data, and a post-study questionnaire using the NASA-TLX [20] to collect perceived task load. Our business process modeling language of choice was BPMN 2.0, due to its wide adoption and its standing as an international standard. The scenarios of the model and rules originated from a car insurance diagram included in OMG's BPMN 2.0 documentation[1]. For expert groups, we used the three integration approaches (one for each treatment group). We ensured, through multiple revisions, that we created informationally equivalent models for all three integration approaches. Due to space limitations, the models cannot be included in the paper, but the complete laboratory study instruments are available for download[2]. We ensured all confounding factors were constant. In particular, the model was adjusted to ensure consistency of format for each of the integration approaches (i.e., using text annotation, diagrammatic, and linked rules). In total, there are three questions in the laboratory study. The questions differed in terms of the modeling constructs a participant will have to review to answer them. The model constructs for Q1 and Q2 include sequence and AND gateways, while Q3 include sequence, AND gateways, and XOR gateways. This diversity allowed us to gain further insights into the relationship between integration approaches and task complexity (reflected by the coverage of the model required to answer a particular question). Moreover, questions differed in terms of their span and each question is related to different process areas and business rules (a participant may have to navigate only a specific section of the process model to answer the question for Q1 and Q2 (local question), or the whole process for Q3 (global question)) (see Sect. 3.3 on details of process areas related to each question answer).

The complete procedure consists of six steps, described as follows. (Step 1) We first used a screening form to recruit potential participants. Then we provided a pre-study questionnaire to eligible participants. To ensure group balance, we used a pre-study questionnaire to capture participants' prior knowledge and basic demographics, which we used to distribute participants across groups to avoid accidental homogeneity. (Step 2) We set up the laboratory study environment with eye tracking device. For each participant, we provided training on the instructions of the laboratory study, eye tracker device and calibration. (Step 3) After calibration, participants were first provided with a BPMN tutorial and were then offered a model using one of the three rule integration approaches. We encouraged each participant to ask questions during the tutorial session, to ensure their readiness for the laboratory study. (Step 4) In the laboratory

[1] Model originated from a travel booking diagram in OMG's BPMN 2.0 examples can be viewed in http://www.omg.org/cgi-bin/doc?dtc/10-06-02.

[2] Please view laboratory study instruments in https://bit.ly/CAiSE2025.

study, all participants had to answer 3 questions. We did not set a limit on the laboratory study duration nor a word count limit on participants' answers. We recorded the gaze movements while experts were working on the tasks. (Step 5) Upon task completion, participants were provided with a post-study questionnaire using the NASA-TLX [20] to collect perceived task load. (Step 6) For each participant, we replayed the recording of the task completion process with their gaze movements, and we asked the following semi-structured questions to understand their behaviours in line with our objectives of the study. (1) Would you summarize the process for answering the questions? And explain why you worked in this way. (2) Please watch the replay video, can you identify the point in the video when you knew the answer (indicate the point in time when identifying the answer)? Explain what made you realize the answer, giving specific details (e.g., which activity/rules make you feel you know the answer?) and how confident are you in your answer to this question? (scale 1–5) (3) Regarding the difficulty level of these three questions, did you find any questions that are easier or harder? (4) Overall, can you comment on difficulties or challenges you experience with regard to understanding the model and rules when answering the questions?

3.3 Setting

The eye tracking data was collected through a Tobii Pro TX300 eye tracker[3], which captures data on fixations, gazes, and saccades with timestamps. The laboratory study was conducted in full-screen mode and complete models were displayed without the function of zooming in or scrolling[4]. The visibility of the text and diagrams was examined carefully, with all text and diagrams being clear from a distance of 1.2 m. All laboratory studies were conducted in the same lab with the same eye tracker.

3.4 Analysis Approach

To uncover the significant differences in knowledge workers' behaviour between the three representations, we conducted the analyses on verbal protocol collected in cued-retrospective think-aloud interviews and complemented them with insights derived from the eye-tracking data. In line with sensemaking foundations (please see Sect. 2.1), we segment the laboratory study into two phases, namely the information foraging phase and the task-specific information processing and answering phase. The information foraging phase for a particular task commences when the participant first fixates on the laboratory study screen, and the information processing phase commences when the participant starts to type the answer in the question area for the first time.

[3] For more specifications of the eye tracker, please visit https://www.tobii.com/products.
[4] Laboratory study design can be viewed and downloaded from https://bit.ly/CAiSE2025.

Verbal Protocol Analysis. To analyse the user insights collected from the cued-retrospective think-aloud interviews, we used NVIVO 12[5] for verbal protocol analysis and we followed the procedure outlined by Gioia et al. [16], to show the "dynamic relationships among the emergent concepts that describe or explain the phenomenon of interest and one that makes clear all relevant data-to-theory connections" [16]. The verbal protocols were transcribed and provided to two independent coders for analysis to reduce bias in our analysis. We started the analysis following an inductive approach and then transitioned into a more abductive approach, to ensure "data and existing theory are now considered in tandem" [16]. The analysis process included four steps. In the first step, we aimed to "let the data do the talking", so we inductively analysed the verbal protocols to identify key processes and strategies on how participants foraged and processed information to solve each task. Then we conducted open coding [38] based on the key themes and foci that emerged from the first step, and then the authors discussed the codes with the independent coder iteratively until an agreement on the first-order coding was reached. In the third step, we aimed to answer "whether the emerging themes suggest concepts that might help us describe and explain the phenomena we are observing" [16]. Using existing theories of sensemaking and cognition combined as a theoretical frame of reference, we reflected on the way in which the second-order concepts represented or related to knowledge workers' sensemaking behaviours in information foraging and information processing. We reviewed the existing literature and theories to analyse and develop concepts that explain the data and we did not allow prior theoretical concepts and assumptions to restrict our interpretations.

Eye Tracking Analysis. In this study, the eye tracking data served as a preliminary observation tool that guided deeper qualitative exploration in CRTA interviews into the sensemaking behaviors of participants, where participants could reflect on their recorded gaze behaviors, enhancing the richness of the qualitative data by connecting it to observable actions. To reveal the insights on nuanced differences in how experts allocated attention across three types of integration approaches during tasks of varying complexity, we conducted complementary analyses of insights from eye tracking metrics.

We analyzed the efficiency of locating the answer during the information foraging phase on relevant areas (a measure of visual association of Areas of Interest (AOIs) [6]). The timestamp of locating the answers is indicated by each expert participant during the cued-retrospective think-aloud session. The duration of locating the answer begins when the participant starts each question until the timestamp when they indicate they found the relevant information on the model area. The AOIs were used for analysis and were invisible to participants. As shown in Fig. 1, for models featuring text annotation and diagrammatic integration, the screen was divided into 8 areas: seven different process model areas and a question area (which showed one question at a time). For models featuring link integration, there was an additional ninth area for rules, which displayed the corresponding business rules when participants clicked on each "R" icon in

[5] For the use of Nvivo 12, please view https://lumivero.com/products/nvivo/.

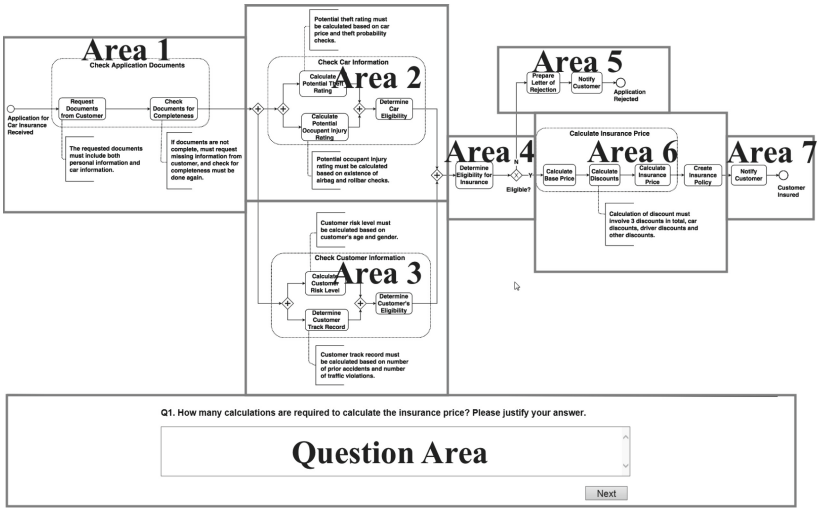


Fig. 1. Visual design in laboratory study - text annotation integration approach

the model. Each question answer is related to different process areas. For local questions Q1 and Q2, the answer is related to area 6 and area 2, respectively. For Q3 (global question), the answer is related to areas 1, 5 and 7.

4 Results and Discussion

As discussed in Sect. 3.4, we followed the analysis procedure outlined by Gioia et al. [16]. Our findings shown in the data structure (please view Fig. 2) revealed that expert knowledge workers have distinct sensemaking loops and strategies during exploratory information foraging, focused information foraging and tasks-specific information processing and answering processes. To suit different information needs, they optimized and switched strategies during these processes. We also found these sensemaking practices were not mutually exclusive, and usually were interrelated and embedded with each other and could be used concurrently based on the different information needs. In addition, all participants expressed that using their gaze recordings was an effective approach for them to remind and explain their attention.

Exploratory Information Foraging. All the experts (15/15) indicated that at the beginning, they would quickly start scanning either from the business process model (structure-oriented) or the task (task-oriented) to understand the meta-information about the context and structure of the business process model and question. E.g., *"I think, before I even like really worked on the first question. I've had the whole process to get an overview. And then I started going module by module (each activity group)."* (P15). Participants explained that they wanted to understand the big picture from the meta-information before zooming into

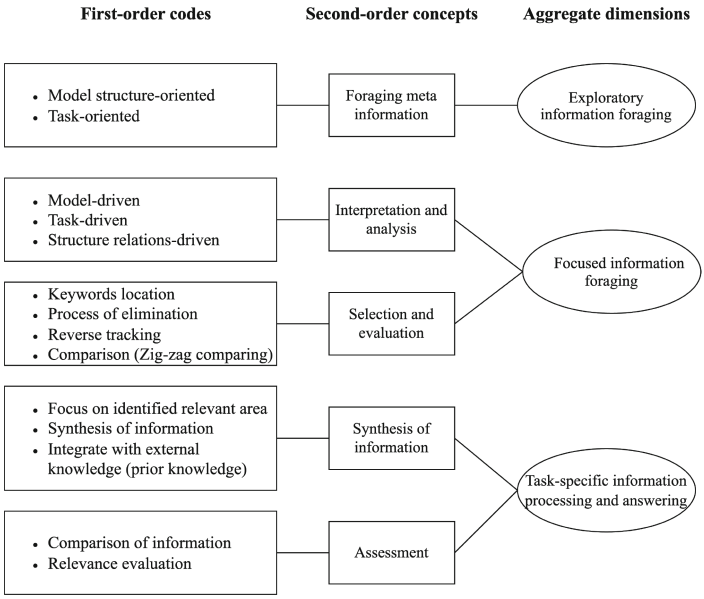


Fig. 2. Data structure of expert process knowledge workers' sensemaking practices

details. This insight is also supported by all experts presenting similar attention on the model area during the first information foraging process in Q1 irrespective of the integration group[6].

Focused Information Foraging. All experts (15/15) indicated that after they understood the meta information they would focus on the details, and we observed that they used various strategies during interpretation, analysis, selection, and evaluation to forage the information on the process model and business rules.

Interpretation and Analysis. The findings revealed several main strategies for gathering and interpreting information, encompassing task-driven, model-driven, and structure-relations driven approaches during the process of foraging information. In the first task (Q1, local question), the majority of the participants (11/15) indicated that they first read through the task and abstracted the key information from the question (**task-driven strategy**). E.g., *"So, in general, I need to look at the question first to locate what kind of information I need to process"* (P12). They motivated this preference for a task-specific strategy and for locating keywords by the complexity of the process model and the capacity of their memory. They argued that they would forget the details of the process model if reading the model first. E.g., *"So, I mean, the diagram is relatively complex and a lot of information is out there. If we look at the dia-*

[6] The normalized fixations duration in Q1: Text annotation (M = 0.195, SD = 0.125), Diagrammatic (M = 0.284, SD = 0.087) and Linked (M = 0.280, SD = 0.137).

gram first, it is possible that we read it from left to right, after we read it, we might even forget what we have read on the left part. So there will be a messy and catastrophes so that is why I choose to identify the key words. Yeah, the verbs are usually very important in phrases. So I believe that we can work out the question successfully." (P6). While going through the details of the process model, the participants following a task-specific strategy mentioned that they considered the relevance between the task, process model, and business rules **(structure relations-driven**) (e.g., *"I just looked down on the question, see whether there is a relation or not and then I will continue"* (P5). In addition, they revisited the tasks after they read through the whole process to remind them of the tasks.

We also observed the **model-driven strategy** in Q1 during focused information foraging, i.e., four participants mentioned they would read through the model first and then read the task. They explained that they wanted to get familiar with all the details in the process model and then focus on the question. *"I think is like that like a habit. Or do things like this, it's quite common like you don't familiar with something, you need to get through the total thing, and you need to get familiar with all the diagram, and then you need to focus on the question, what the questions is asking about."* (P8). All of them mentioned after they read the task, they considered the relevance between the model and the tasks **(structure relations-driven**).

Despite the differences in strategies (either tasks-specific or model-driven) during the interpretation and analysis of focused foraging information, all experts (15/15) reported that they read through all details when understanding the model and business rules in Q1. E.g. *"I'm going everywhere right now because it's the first time that I'm looking at this. And this is the first question. So I feel like I have to go through everything almost at this point and understand what is happening."* (P5). Meanwhile, while comprehending the model, all experts (15/15) indicated that they would both abstract the key part of the information from the model and remember its logic and structure (**structure-relations driven strategy**). E.g., *"I just remember the key parts of the model, so for example, this is not a key part, here, so I know that the key points are these three, so most of the work is done here."* (P5). The insights align with the findings of eye tracking data, where all the experts had similar efficiency in locating the answer for the local question (Q1)[7].

Selection and Evaluation. To select and evaluate the relevant information in different tasks, all experts (15/15) indicated they used different strategies on each task based on their information needs and work habits. These strategies include keyword location, the process of elimination, reverse tracking, and comparison of information and structure between model and task, and within the process model. They expressed that using process elimination, identifying the end of the relevant area, and reverse tracking information can facilitate them downsizing the process to locate the relevant information. Determining the end of the relevant area instead of reading through the whole model again is the

[7] The efficiency of locating the answer in Q1: Text annotation (M = 0.166, SD = 0.126), Diagrammatic (M = 0.116, SD = 0.013) and Linked (M = 0.213, SD = 0.145).

preference in answering local questions. They also would travel reversely and track the required sequence flows, activities and rules (e.g. *"I will go back to graph reversely to see what kind of path I need to go through to get that result."* (P12)). They explained that these strategies enabled them to better locate relevant information areas and eliminate irrelevant parts, e.g., *"I think I like work with the process of elimination... So that helps you look at smaller number of things, rather than going back again, and again, on the full chart. Every time if I go through every step, it will take a lot more time."* (P13).

In Q2 and Q3, we noticed they started to **optimize their strategies** in foraging and processing information based on different information needs for different integration tasks. All experts (15/15) used a task-specific strategy when working on Q2 and Q3. They expressed that they read questions first and adopted different strategies based on the evaluation of the tasks. E.g., *"it's a different question. It's a different question requires a different strategy."*(P3). After interpreting and analysing task types, all experts expressed they used a different strategy for local and global questions to select and evaluate the relevant information based on their prior experience in practice. In local questions (Q1 and Q2), the dominant strategy is from end to start (reverse tracking), but in global question Q3, all participants expressed they worked from start to end. E.g., *"actually, yes, question one and two are similar. Question three is a different approach. Question one and two, I work from end to start, but question three I work from start to finish."* (P3).

All of the experts (15/15) expressed that, since they understood the process model in Q1, they directly used the strategy of targeting the keyword to locate the relevant information on the model and rules from their memory in Q2 and Q3. They further stated that they read the model and business rules more efficiently and selectively to evaluate and target the information based on the specific information needs required from the question, instead of foraging for all details.

Based on expert participants' insights, we assume that the initial focused information foraging during the process model and rule understanding played a vital role in building a mental model in their working memory, which will directly influence their attention when completing the following tasks. E.g.,*"for this one, I already knew the process a little bit. And so I could directly search for keywords, determine eligibility for the car, here eligibility, it's quite in the centre of the screen."* (P15). All participants expressed that while locating the relevant area to answer questions, they assessed the information in multiple rounds of comparison and evaluation to ensure their answers were relevant. For global question Q3, all participants (15/15) mentioned that evaluating the task made them realise the answer requires more areas and rules when compared to the other two local questions, so they went through the whole process model and rules. E.g., *"Because here the question was not concerned with one specific outcome like determine eligibility, but it was concerned with overall the whole process, what is the minimum, so I had to go through the whole process."* (P15).

As task complexity increases, all experts presented similar efficiency in locating the answer in the global question (Q3)[8].

Task-Specific Information Processing and Answering. All experts (15/15) indicated that once they located and confirmed the relevant information in each question, they started to type the answers. We observed during this process of answering and assessment, all experts focused on the identified relevant area to synthesize and evaluate relevant information and integrated the information with any prior knowledge or external knowledge. All participants expressed that when they evaluated their answer, they would only target some relevant areas directly based on their memory, but they would not read through all models or rules to confirm again, and they only read it based on the information needed to verify and ensure that they did not overlook anything (e.g., *"for the minimum number, you kind of have to go through the whole process...it's based on the situation so I looked into each rule again quickly, so to make sure that I don't overlook anything and then I will short case."* (P15).

5 Conclusion and Outlook

In this paper, we advanced the understanding of sensemaking practices of process knowledge workers engaged in tasks on integrated modelling of business processes and rule. Utilizing cued-retrospective think-aloud interviews (with eye-tracking as the cue), our study conducted a deep qualitative exploration that revealed diverse sensemaking practices and strategies experts employed during information foraging and information processing. Our findings echoed the iterative character noted in prior sensemaking work [9,31,46], but we uncovered a number of specific observations in expert process workers' behaviours and strategies when performing tasks based on integrated modeling of business processes and rules, namely (1) We discovered that sensemaking practices unfold through various processes in information foraging and processing phases, notably highlighting that the information foraging phase consists of two distinct sub-phases, i.e., exploratory and focused information foraging phases. (2) Our study reveals that expert process workers exhibited a dynamic adaptation to different information needs by optimizing and switching strategies between these phases. Such strategic shifts are crucial in accommodating the varying degrees of complexity and specificity of the information foraged. (3) Further, we discovered that sensemaking practices are not mutually exclusive but are interrelated and can be concurrently employed.

While recent advancements in AI-assisted tools offer promising capabilities for process model analysis and decision support [13], they fall short in capturing the nuanced tacit and experiential knowledge [4,36] that expert knowledge workers apply in real-world scenarios. These limitations can lead to potential

[8] The efficiency of locating the answer in Q3: Text annotation (M = 0.047, SD = 0.035), Diagrammatic (M = 0.046, SD = 0.047), and Linked (M = 0.070, SD = 0.039).

misinterpretations of complex business processes and business rules, particularly in contexts where domain-specific expertise is essential but absent. Considering these challenges, gaining a deeper understanding of how expert knowledge workers forage for and process information within integrated process models becomes essential. Our work presents such insights, which are crucial for supporting the development of new, more effective process-oriented tools and systems that enhance decision-making processes. Drawing on our findings, we argue that the development of tools and systems in integrated process and rule modeling should be informed by the cognitive mechanisms uncovered through the analysis of sensemaking loops specific to process knowledge workers' expertise. In essence, our research underscores the importance of offering personalized support mechanisms to improve the sensemaking practices of different user groups.

Our study is not without limitations. First, our research focused on integrated process models, yet it did not explore how business rules [15,42] specifically influence expert process workers' information foraging and processing. This limitation presents an opportunity for future research to explore the impact of the full complexity of understanding business rules and impacts on information processes. Second, we only considered basic constructs in business process models whereas advanced constructs (e.g., events, loops and nesting structures) may introduce further complexities in sensemaking. Third, our participant pool of researchers with practitioner backgrounds was limited and the sample size may diminish the generalizability to industrial settings. While the sample size is considered adequate given the exploratory and qualitative nature of our study, which is in alignment with established qualitative research methodologies that value information richness over large sample sizes [1,24], a higher number of expert participants may provide better explanatory power. We consider our combined approach of using cued-retrospective think-aloud interviews with eye tracking as a methodological contribution that can inspire further research. Specifically, the methodology outlined in this study can also be applied and extended for research on other conceptual notations and domains with similar related but notationaly distinct artefacts, for example, database constraints and data models. In our future work, we plan to contribute to both the theoretical framework of sensemaking and the practical aspects of designing tools and systems that support effective information foraging, processing and decision-making in multi-artifact information tasks such as business process and rule integration.

We consider our combined approach of using cued-retrospective think-aloud interviews with eye tracking as a methodological contribution that can inspire further research. Specifically, the methodology outlined in this study can also be applied and extended for research on other conceptual notations and domains with similar related but notationaly distinct artefacts, for example, database constraints and data models. In our future work, we plan to contribute to both the theoretical framework of sensemaking and the practical aspects of designing tools and systems that support effective information foraging, processing and decision-making in multi-artefact information tasks such as business process and rule integration.

Acknowledgments. This research is partially supported by the Australian Research Council (ARC) Training Centre for Information Resilience (Grant No. IC200100022) and by an Australian Research Council (ARC) Future Fellowship Project (Grant No. FT240100022).

References

1. Abbad Andaloussi, A., Burattin, A., Slaats, T., Petersen, A.C.M., Hildebrandt, T.T.,Weber, B.: Exploring the understandability of a hybrid process design artifact based on DCR graphs. In: EMMSAD 2019, Proceedings 20, pp. 69–84 (2019)
2. Abbad Andaloussi, A., Zerbato, F., Burattin, A., Slaats, T., Hildebrandt, T.T., Weber, B.: Exploring how users engage with hybrid process artifacts based on declarative process models: a behavioral analysis based on eye-tracking and think-aloud. Softw. Syst. Model. **20**, 1437–1464 (2021)
3. Adams, J.K.: Laboratory studies of behavior without awareness. Psychol. Bull. **54**(5), 383 (1957)
4. de Almeida Rodrigues Gonçalves, J.C., Baiao, F.A., Santoro, F.M., Guizzardi, G.: A cognitive bpm theory for knowledge-intensive processes. Bus. Process Manag. J. **29**(2), 465–488 (2023)
5. Bera, P., Burton-Jones, A., Wand, Y.: Guidelines for designing visual ontologies to support knowledge identification. MIS Q. 883–908 (2011)
6. Bera, P., Soffer, P., Parsons, J.: Using eye tracking to expose cognitive processes in understanding conceptual models. MIS Q. **43**(4), 1105–1126 (2019)
7. Boren, T., Ramey, J.: Thinking aloud: reconciling theory and practice. IEEE Trans. Prof. Commun. **43**(3), 261–278 (2000)
8. Chen, F., et al.: Robust Multimodal Cognitive Load Measurement. Springer, Cham (2016)
9. Chen, T., Demartini, G., Indulska, M., Sadiq, S.: Exploring data workers' behaviours in data quality discovery. In: ACIS 2023 Proceedings, no. 99 (2023)
10. Chen, T., Sadiq, S., Indulska, M.: Sensemaking in dual artefact tasks - the case of business process models and business rules. In: International Conference on Conceptual Modeling, pp. 105–118 (2020)
11. Chen, T., Wang, W., Indulska, M., Sadiq, S.: Business process and rule integration approaches-an empirical analysis. In: BPM Forum 2018, pp. 37–52 (2018)
12. Duchowski, A.T.: Gaze-based interaction: a 30 year retrospective. Comput. Graph. **73**, 59–69 (2018)
13. Dumas, M., et al.: Ai-augmented business process management systems: a research manifesto. ACM Trans. Manag. Inf. Syst. **14**(1), 1–19 (2023)
14. Figl, K., Laue, R.: Influence factors for local comprehensibility of process models. Int. J. Hum. Comput. Stud. **82**, 96–110 (2015)
15. Figl, K., Mendling, J., Tokdemir, G., Vanthienen, J.: What we know and what we do not know about DMN. EMISAJ **13**, 2–1 (2018)
16. Gioia, D.A., Corley, K.G., Hamilton, A.L.: Seeking qualitative rigor in inductive research: notes on the Gioia methodology. Organ. Res. Methods **16**(1), 15–31 (2013)
17. Gruhn, V., Laue, R.: Adopting the cognitive complexity measure for business process models. In: 2006 5th IEEE International Conference on Cognitive Informatics, vol. 1, pp. 236–241. IEEE (2006)

18. Haisjackl, C., et al.: Understanding declare models: strategies, pitfalls, empirical results. Softw. Syst. Model. **15**(2), 325–352 (2016)
19. Haisjackl, C., Soffer, P., Lim, S.Y., Weber, B.: How do humans inspect BPMN models: an exploratory study. Softw. Syst. Model. **17**, 655–673 (2018)
20. Hart, S.G.: Nasa-task load index (NASA-TLX); 20 years later. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 50, pp. 904–908 (2006)
21. Knolmayer, G., Endl, R., Pfahrer, M.: Modeling processes and workflows by business rules. In: Business Process Management, pp. 16–29. Springer, Cham (2000)
22. Kurtz, C.F., Snowden, D.J.: The new dynamics of strategy: sense-making in a complex and complicated world. IBM Syst. J. **42**(3), 462–483 (2003)
23. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Business process model merging: an approach to business process consolidation. TOSEM **22**(2), 1–42 (2013)
24. Marshall, B., Cardon, P., Poddar, A., Fontenot, R.: Does sample size matter in qualitative research?: a review of qualitative interviews in is research. J. Comput. Inf. Syst. **54**(1), 11–22 (2013)
25. Mendling, J., Strembeck, M.: Influence factors of understanding business process models. In: International Conference on Business Information Systems, pp. 142–153 (2008)
26. Mendling, J., Strembeck, M., Recker, J.: Factors of process model comprehension-findings from a series of experiments. Decis. Supp. Syst. **53**(1), 195–206 (2012)
27. Naumer, C., Fisher, K., Dervin, B.: Sense-making: a methodological perspective. In: Sensemaking Workshop, CHI, vol. 8, pp. 506–513 (2008)
28. Nielsen, J.: Estimating the number of subjects needed for a thinking aloud test. Int. J. Hum. Comput. Stud. **41**(3), 385–397 (1994)
29. Petrusel, R., Mendling, J.: Eye-tracking the factors of process model comprehension tasks. In: 25th International Conference, CAiSE 2013, pp. 224–239. Springer, Cham (2013)
30. Petrusel, R., Mendling, J., Reijers, H.A.: Task-specific visual cues for improving process model understanding. Inf. Soft. Tech. **79**, 63–78 (2016)
31. Pirolli, P., Card, S.: The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In: Proceedings of International Conference on Intelligence Analysis, vol. 5, pp. 2–4 (2005)
32. Rayner, K.: Eye movements in reading and information processing: 20 years of research. Psychol. Bull. **124**(3), 372 (1998)
33. Reijers, H.A., Mendling, J., Dijkman, R.M.: Human and automatic modularizations of process models to enhance their comprehension. Inf. Syst. **36**(5) (2011)
34. Rolón, E., García, F., Ruiz, F., Piattini, M., Visaggio, C.A., Canfora, G.: Evaluation of BPMN models quality-a family of experiments. In: International Conference on Evaluation of Novel Approaches to Software Engineering, vol. 2, pp. 56–63 (2008)
35. Russell, D.M., Stefik, M.J., Pirolli, P., Card, S.K.: The cost structure of sensemaking. In: Proceedings of the INTERACT 1993 and CHI 1993, pp. 269–276 (1993)
36. Sanzogni, L., Guzman, G., Busch, P.: Artificial intelligence and knowledge management: questioning the tacit dimension. Prometheus **35**(1), 37–56 (2017)
37. Schreiber, C., Abbad-Andaloussi, A., Weber, B.: On the cognitive and behavioral effects of abstraction and fragmentation in modularized process models. Inf. Syst. **125**, 102424 (2024)
38. Strauss, A., Corbin, J.: Basics of qualitative research techniques (1998)
39. Sweller, J., Ayres, P., Kalyuga, S.: Measuring cognitive load. In: Cognitive Load Theory, pp. 71–85 (2011)

40. Turetken, O., Rompen, T., Vanderfeesten, I., Dikici, A., van Moll, J.: The effect of modularity representation and presentation medium on the understandability of business process models in BPMN. In: BPM, pp. 289–307 (2016)
41. Van Gog, T., Paas, F., Van Merriënboer, J.J., Witte, P.: Uncovering the problem-solving process: cued retrospective reporting versus concurrent and retrospective reporting. J. Exp. Psychol. Appl. **11**(4), 237 (2005)
42. Vanthienen, J.: Quality by design: using decision tables in business rules. Bus. Rules J. **5**(2), 7 (2004)
43. Vayghan, J.A., Garfinkle, S.M., Walenta, C., Healy, D.C., Valentin, Z.: The internal information transformation of IBM. IBM Syst. J. **46**(4), 669–683 (2007)
44. Wang, W., Chen, T., Indulska, M., Sadiq, S., Weber, B.: Business process and rule integration approaches-an empirical analysis of model understanding. Inf. Syst. **104**, 101901 (2022)
45. Weick, K.E.: Sensemaking in Organizations, vol. 3. Sage (1995)
46. Zhang, P., Soergel, D.: Cognitive mechanisms in sensemaking: a qualitative user study. J. Assoc. Inf. Sci. Tech. **71**(2), 158–171 (2020)
47. Zhou, H., Zerbato, F., Weber, B., Indulska, M., Sadiq, S.: Do process analysts care about the metadata of event logs? In: ACIS 2023 Proceedings, vol. 32 (2023)
48. Zugal, S., Pinggera, J., Weber, B., Mendling, J., Reijers, H.A.: Assessing the impact of hierarchy on model understandability–a cognitive perspective. In: Models in Software Engineering, pp. 123–133 (2012)
49. Zugal, S., Soffer, P., Haisjackl, C., Pinggera, J., Reichert, M., Weber, B.: Investigating expressiveness and understandability of hierarchy in declarative business process models. Softw. Syst. Model. **14**, 1081–1103 (2015)
50. Zur Muehlen, M., Indulska, M., Kittel, K.: Towards integrated modeling of business processes and business rules (2008)

Automated Recommender System Integration in Model-Based Ecosystems

Rickson Simioni Pereira[1(✉)], Claudio Di Sipio[2], Martina De Sanctis[1], and Ludovico Iovino[1]

[1] Gran Sasso Science Institute, L'Aquila, Italy
{rickson.pereira,martina.desanctis,ludovico.iovino}@gssi.it
[2] University of L'Aquila, L'Aquila, Italy
claudio.disipio@univaq.it

Abstract. Recommender systems (RSs) are a special class of information systems devoted to assisting end-users by providing valuable items according to the context of the application. Nevertheless, implementing those systems is time-consuming as developers must elicit the proper technology that may vary according to the context. Even though existing frameworks have been proposed to facilitate the deployment of those systems, automated integration with existing model-based systems is still an open challenge.

In this paper, we propose a model-driven engineering approach to automatically configure filtering-based RSs by relying on automated transformations and weaving modelling. In particular, we rely on two different metamodels to represent the application domain (variable) and the generic components of filtering-based RSs. Then, we define a weaving model to map each domain entity to a specific RS element e.g., algorithm or data encoding, thus realizing a conceptual link between the context and the system to be deployed. The proposed framework is then able to generate domain-specific model-based recommendations, that can be integrated in model-based systems. We used the proposed framework to implement two different types of filtering-based RSs evaluated using two different scenarios, i.e., movies and tourism recommendations. Our findings suggest that the proposed approach can be adopted to simplify the data ingestion of RS-based filtering and automate the generation of model-based recommendations.

Keywords: MDE · MBSE · recommender systems · integration

1 Introduction

Recommender systems (RSs) [37] are complex software systems that provide valuable items to end-users in several application domains, e.g., tourism [36], e-commerce [30], and code development [15,39]. However, they exploit heterogeneous components that increase the overall complexity in the deployment phase. Although a plethora of frameworks have been proposed to simplify their design

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 20–38, 2025.
https://doi.org/10.1007/978-3-031-94571-7_2

and evaluation [3,17,32], they neglect to provide abstract modules to handle input data, which is the prominent source of information in traditional systems, i.e., filtering based RSs. Moreover, developers are forced to update or even rewrite from scratch data encoding components according to the application domain as shown in [34]. For instance, creating RSs for different scenarios, such as an RS for tourism in a smart city with IoT sensors, an RS for tourism in a smart city without IoT infrastructure, and an RS for movies. Note that the first two scenarios are similar (both within the tourism domain) with slight modifications, while the third scenario diverges into an entirely different sector. Consequently, given the specifications imposed on each scenario and application domain, the development and design of the respective RSs demand different implementations and then redundant work. To fill this gap, we propose a novel approach to easily handle input data to enable the deployment of filtering-based RSs leveraging model-driven engineering (MDE) paradigm [44]. In particular, we adopted model weaving [5] that exploits two different metamodels, i.e., RS generic model and domain models. The former aims at representing the high-level features of three well-known RS algorithms, i.e., collaborative filtering [43] and content-based algorithm [41], and hybrid-based RS [9]. Meanwhile, the latter represents specific domain concepts that can be used flexibly, allowing flexible configuration to specific domain requirements without extensive re-engineering. Afterwards, we annotate RS-specific concepts in domain models to support weaving modelling as a knowledge base that will be used as the source of recommendation. We eventually extract relevant data directly by querying these models. To deploy the system and collect the recommendations from the configured RS, we exploit the Apache Mahout Java library [4] as it can support filtering-based RS and can be easily integrated into our MDE framework. The output of the process is a model embedding recommendations of the processed input models, that can be integrated into existing code generation approaches or used as input to print out the recommendations. In particular, our research is guided by the following Research Question (RQ):

RQ: To what extent can the proposed approach support the configuration of filtering-based RSs in different application domains?

Nevertheless, the proposed approach represents a valuable asset for mapping domain-specific concepts to generic RS components and automating the construction of a model-based RS. While we acknowledge that not all developers are confident in defining MDE artefacts from scratch, the definition of metamodels and models can be facilitated by relying on existing approaches [11,14,52], thus overcoming this gap. In addition, the specified modelling artefacts can be easily extended to cover RSs based on advanced algorithms, i.e. stochastic models [46], or neural networks [28,54]. The main contributions of the paper are *i)* a conceptualization of the multi-level relationship between domain-specific models and RSs concepts; *ii)* an MDE framework that assists developers in configuring processes to deploy filtering-based RSs; *iii)* a scenario-based evaluation to show the expressiveness of the proposed solution in two different application domains,

i.e., movie and tourism recommendations. The structure of the paper is as follows: in Sect. 2 we bring the grounds on which we are basing our work. Section 3 introduces the approach, raising as well two application examples demonstrating the feasibility of the approach. The validation and threats to validity are presented in Sect. 4 while related works are discussed in Sect. 5. Finally, we conclude the paper planning possible future works in Sect. 6.

2 Background and Motivation

The purpose of the section is two-fold. First, we introduce key MDE concepts and artefacts. Then, we overview existing approaches, highlighting the need to automate the handling of data sources in RSs.

Model Driven Engineering. MDE [44] paradigm relies on the so-called Meta-object facility (MOF), a formal standard provided by OGM to express abstract concepts using modelling artefacts to facilitate the overall development lifecycle and increase productivity by introducing automation [22,25]. In this context, a *model* is an abstraction of real-world entities that synthesize their crucial features. Models are built using Domain-Specific Modelling Languages (DSMLs) specified through a domain model (metamodel), expressing concepts, relationships, and constraints of the application domain [31]. The standard specification language for metamodels in the Eclipse Modeling Framework (EMF) [18] is called Ecore. It permits the specifying of concepts, relationships and constraints reflecting the language's abstract syntax with a dedicated language. The Epsilon Modeling Framework (EMF) is the standard-de-facto distribution based on Eclipse used by academia and industry [45]. *Epsilon* is a family of languages built on top of EMF. The core of these languages is the Epsilon object language (EOL), which can support different MDE tasks, i.e., model manipulation., model validation, model transformation, and model migration. EOL can also be used as a general-purpose standalone model management language for automating tasks that do not fall into the patterns targeted by task-specific languages. In EMF, models can be instantiated from defined metamodels in different formats and technical spaces. XMI is the format used by EMF, while Flexmi [27] is a generic and modular textual syntax for domain-specific modelling. A metamodel defines a "model type" and, at the same time, provides a way to validate the defined models, based on the typing relationship called *conformance* (C2) [20]. A *Model transformation* is an automation able to translate a model into another formalism. Executable descriptions are necessary to execute the transformation with an input model [10]. The specification of a transformation can be either textual (rule-based) or graphical. ETL [26] is the rule-based transformation language offered by Epsilon. In the family of transformation languages, EGL [40] is the Epsilon generation language that is used to develop template-based code generators. The Epsilon framework also provides a tool named Picto [53], which, using the model-to-text transformation, can transform domain-specific models into web-based views. Models are not isolated entities, but they live in ecosystems [23] where modelling artefacts cooperate for a specific purpose [7]. *Model weaving* is

a technique used to capture relationships between model elements [12,13] that can help in formalizing correspondences or mapping between models.

Filtering-Based RSs. RSs [37,38] embraces a wide class of complex systems devoted to providing valuable items given a certain domain. More specifically, filtering-based RSs are tailored to capture users' preferences, and they are still used by big players in IT and e-commerce services, e.g., Netflix, or Amazon. Among the main adopted techniques, *collaborative Filtering (CF)* RSs [43] relies on n user-item matrices that are used to retrieve the outcomes according to different similarity functions. Apache Mahout is one of the many frameworks available to assist in developing RSs; its features offer the possibility of creating scalable and efficient RSs that can be integrated into Java-based systems. It is widely used in academia and the industry showing its versatility and reliability in handling real-world recommendation scenarios, from personalized content delivery to large-scale e-commerce platforms [6,49,51]. However, this type of algorithm suffers from the so-called cold-start problem, i.e., when the system does not have an initial set of ratings or items to perform the training phase [8]. Thus, *content-based (CB)* [41] technique has been conceived to embody users' previous choices, suggesting items that are similar to their past preferences. To handle peculiar application domains, CF and CB algorithms can be combined to develop *hybrid-based RS (HB)* [9]. While we acknowledge that recent RS exploits AI-based algorithm [42], we limit ourselves in configuring the mentioned filtering-based RSs, posing the focus on mapping domain-specific concepts to generic components. Nevertheless, it is worth mentioning that the proposed approach can be easily adapted to configure the AI-based systems as this will impact the code generator module only, which can be easily updated to produce Python code as done in prior research [17].

3 Automated RS Integration in Model-Based Ecosystems

This section overviews the proposed approach and how we built a framework to support it. Figure 1 depicts the proposed approach that is compliant with the Meta-Object Facility (MOF) standard [35], combines different modelling artefacts and, in 7 steps (marked with circled numbers from ❶ to ❼), automates the generation of domain-specific model-based recommendations. We distinguish the proposed framework components as *User-specified* to identify artefacts that are specified by the modeller and are domain-dependent, i.e., variable; *Provided by the framework* to identify components which the modeller simply uses.

We intend to provide a way to integrate the recommender system with existing modelling languages that could be provided by the user. Indeed, the process starts with an annotation phase ❶ of a Domain Model where the modeller maps the domain-specific concepts with concepts of a recommender system. This is done thanks to the *concept inheritance* at m2. Indeed, EMF supports multiple inheritances, enabling the mixing of structural features like attributes at multiple places in the generated class hierarchy. This workaround poses the Recommender System's concept at one level above the domain model, which is often used in a

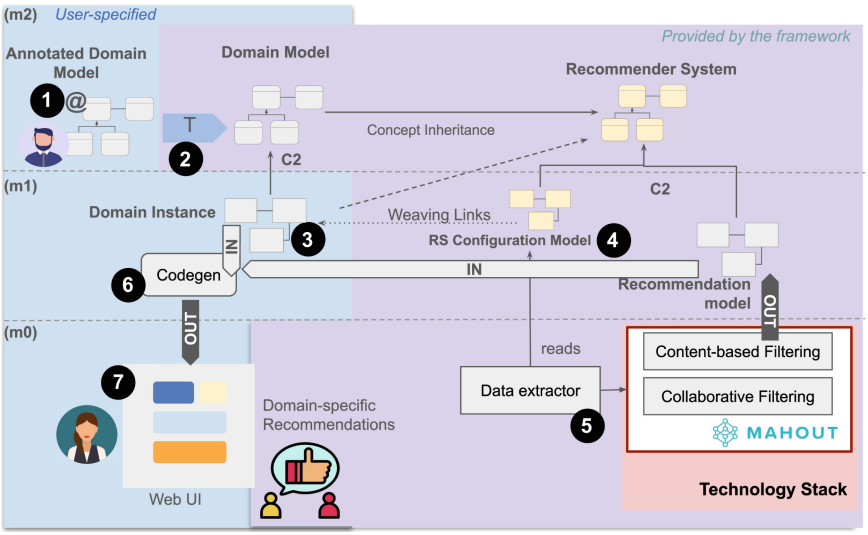


Fig. 1. Overview of the proposed approach

multi-level modelling setting [29], but for the sake of simplicity, we keep them at the same modelling level since technically they are treated both as metamodels.

Figure 2 shows the generic Recommender System metamodel that can be used to represent a `FilteringRS` and its main components, i.e., `CollaborativeFiltering` and `ContentBased` RSs. Concerning the configuration parameters, we specify the `SimilarityFunction` represented as an enumeration of state-of-the-art functions e.g., Cosine similarity or Jaccard distance. Moreover, on the CF instance, we appoint if the recommendations are user-based with the attribute: `isUserBased` and the number of `neighbors` utilized to recommend items to the end-user. Notably, the conceived metamodel allows the specification of `HybridBased`.

While we acknowledge that several metamodels for representing RS are in place [2,17,32], in this paper, we focus on handling three different types of user feedback expressed on the items, i.e., *ratings*, content-based *preferences* and implicit ones extracted from the user *profile*. The metamodel can be further extended to support other feedback. Concerning the first type, the metaclass `UserItemRow` represents a triple of an `User`, `Item`, and a float value that represents the rating. Roughly speaking, the metamodel allows the definition of a `UserItemMatrix` that represents a list of rows in which a user *expresses* a rating on a particular item. Such a matrix feeds the `CollaborativeFiltering` algorithm, represented by the corresponding metaclass. To handle preferences to train a `ContentBased` system, we used the `ContentBasedPreference`, which is a list of generic `Preference`.

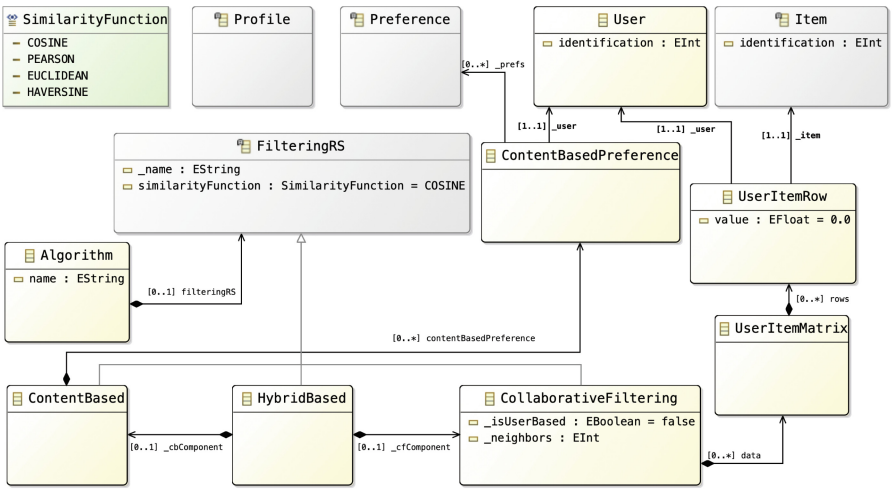


Fig. 2. Recommender system metamodel.

Since this concept strongly relies on the application domain, we intentionally left the class generic enough to be mapped using the weaving approach. Similarly, a user can have a `Profile` that needs to be customized according to the type of domain supported by the wanted RS. A model of a specific space can be consistent with a model from another space if there is a bijection between concepts of both spaces [47]. Extending a metamodel does not mean composing and merging two different metamodels [16], but in our case, we need a metamodel extension based on the conceptual correspondence. To this end, the transformation T at ❷ automates the concept inheritance specification and refactors the domain model enriched with annotation including the concepts inheritance. The RS concepts indirectly related to specific domains are multiple; for instance, the `Users`, or the `Items` are domain-specific and depend on the application domain they support. For instance, in a Movie RS, the user is a "viewer" and will be represented differently from those in a Tourism RS, who are "tourists". The same goes for the items to recommend; in the first case, they will be movies, while in the second, they will be itineraries or POIs. This applies also to other concepts such as preferences, profiles, etc.

This is an endogenous transformation refactoring the metamodel according to the mappings, *de-facto* extending concepts from the domain model with concepts from the RS, i.e., the *Concept inheritance* relationship. This transformation supports an "extension" of the domain instance to the RS metamodel. We represent this extension with a dotted C2 relationship since it is not formalized but conceptually valid. Nonetheless, it allows us to declare instances in m1 that will conform to both the domain model and the RS metamodel.

We reported an excerpt of these mappings in Table 1, where we indicate the annotation, the Ecore element that can be applied, the refactoring applied by

the transformation, and the RS target element. In a few words, the "viewer" for a movie's RS will be typed as a Viewer in the movie domain, but it will also be considered as a User of the RS, and so on.

Table 1. Excerpt of some of the applied mapping and applied refactorings in T

Annotation	Applied on	Refactoring	RS target
@user	EClass	The metaclass inherits from the User metaclass	CF/CB
@item	EClass	The metaclass inherits from the Item metaclass	CF/CB
@preference	EAttribute	The metaclass is created on the domain's side and inherits from the Preference metaclass	CB
@preference	EReference	The metaclass is created tracing the Algorithm preferences inheriting from the Preference metaclass	CB
@profile	EAttribute	The metaclass inherits from the Profile metaclass	CF/CB

We do not include the full transformation due to space constraints. Still, it is a single endogenous transformation written in the Epsilon Transformation Language (ETL) [19], with one rule for each row in Table 1. This transformation modifies the metamodel, enabling the creation of weaving links between RS model and application domain instances at the m1 level. For example, the RS metamodel supports the creation of Collaborative Filtering items, which are specified using the UserItemMatrix. Each UserItemRow links an RS user (via _user in UserItemRow → User in Fig. 2) to an RS item (via _item in Fig. 2). After applying the transformation, the modeller can associate a user with an item in the application domain. This is made possible by the transformation, which reads annotations to create the necessary mappings. For instance, the user annotation allows the domain-specific User metaclass to extend the RS-specific User metaclass, enabling the creation of typed weaving links in step ❹.

In Fig. 3, we represented the refactoring applied to the domain model after applying the first rule derived from the first mapping in Table 1. A metaclass in the domain model including an annotation user will be refactored as the right-hand side of Fig. 3, i.e., inheriting from the User metaclass of the RS metamodel in Fig. 2. This process is metamodel agnostic, i.e., it can then be replicated by switching the application domain and maintaining the RS metamodel fixed.

Once the modeller has specified a Domain Instance to work with and an RS Configuration Model, which expresses the configuration of the desired recommender system (in ❸), they can automatically configure the supported filtering-based RSs, i.e., CF, CB, or combine them in an HB RS.

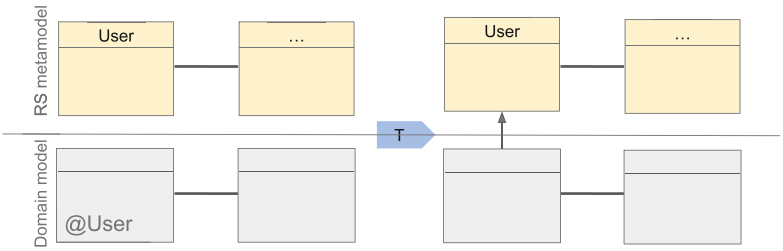


Fig. 3. Simplified effect of the application of the first rule mapping in Table 1

In ❺, the Data Extractor component extracts the training data from the weaving models composed in ❹. Specifically, it navigates the RS configuration model to the Domain instance and translates the models into a format suitable for processing by the RS framework in Mahout, which we use as a technology stack. Mahout seemed to be the natural integration since we use EMF as a modeling framework that is based on Java. The data, retrieved from the EOL Script, is then passed through the Apache Mahout framework to generate the recommendations.

In Listing 1, we provide an excerpt of the data extractor that reads the models, creates maps to be input into the RS engine, and generates an output with all the necessary data from the models, for then, consequently, generate the output Recommendation model. In this output model, the data extractor will be able to relink the original domain model instances to the recommendations. It is worth noting, for instance, as explained above the system retrieves the user ID (Line 10) by navigating the user instance linked to the application domain's model instance. Similarly, the recommendation items are extracted from the application domain (see Lines 15–18), with both pieces of information essential for generating meaningful final recommendations.

```
1  ..
2  var userData = new Map();
3  var itemData = new Map();
4  var ratingsData = new Sequence();
5  var categoriesData = new Sequence();
6  for (user in recommendersystemModel!User.allInstances()) {
7      var userDetails = new Map();
8      userDetails.put("userId", user.identification);
9      userDetails.put("userName", user.name);
10     userData.put(user.id, userDetails);
11 }
12 ...
13 for (Item in recommendersystemModel!Item.allInstances()) {
14     var itemDetails = new Map();
15     itemDetails.put("itemId", Item.identification);
16     itemDetails.put("itemName", Item.name);
17     itemDetails.put("category", Item.category.category);
18     itemData.put(Item.id, itemDetails);
19 }
20 ...
```

Listing 1. Excerpt of the EOL Data Extractor

Concerning the current implementation, we do not report for lack of space the Java code executing the data extractor and the output recommendation model

generation. These components load the RS model specifying the configuration and the domain model to generate the expected output as an XMI model.

The generated output model can be used in multiple ways: be parsed and print out the recommendations in the Eclipse console, generate a CSV, or be processed as an additional input of an existing Codegenerator ❻ to generate domain-specific recommendations in web UI (❼). This recommendation model in output is kept separated to ensure that an existing code generator can be re-used and likely integrated with this additional model to refine the generated Web UI with user-specific recommendations.

4 Validation

In this section, we first introduce the two application domains, tourism and movies, in which RS can be deployed. Then, we applied the proposed approach to answer *RQ* formulated in Sect. 1.

4.1 Scenarios and Datasets

➢ *Tourism domain.* In the context of the RASTA project[1], we developed a recommender system to suggest suitable itineraries in a natural park scenario. We are currently utilizing a dataset provided by other fellows involved in the project. The dataset contains twenty POIs of six different categories, all placed within the National Park of Abruzzo, Lazio, and Molise[2]. Since real users and the corresponding ratings are not available yet, we manually created twenty users with the corresponding ratings assigned to more than one POI. Overall, we populated the model with 200 different ratings that can produce 52 itineraries.
➢ *Movie domain.* To test the system, we employ the well-known MovieLens dataset [21] that represents a benchmark in the recommender systems domain [3,33]. In particular, the dataset contains more than 9,000 movies and 600 users, resulting in more than 10,000 ratings. This dataset offers the possibility to demonstrate how the RS would behave when applied to a scenario with more robust data.

4.2 Results

Tourism Domain. Figure 4[3] reports the conceived domain model (metamodel) designing a travel RS for Natural Parks. As prescribed by the scenario, this domain can include `Itinerary`, `POI` with the associated `Category`, and the `Tourist`.

The annotations that we specified to apply the approach are reported in the metamodel (with violet icon) and outline, for example, that the tourist is the User in our RS[4]. In contrast, the POI is considered as the Item to recommend

[1] https://rasta.unicampania.it/.
[2] https://www.parcoabruzzo.it/Eindex.php.
[3] We report directly the annotated metamodel processed by the transformation `T`.
[4] Some annotations are not visible in the Ecore Diagram, for instance, the ones specified on structural features.

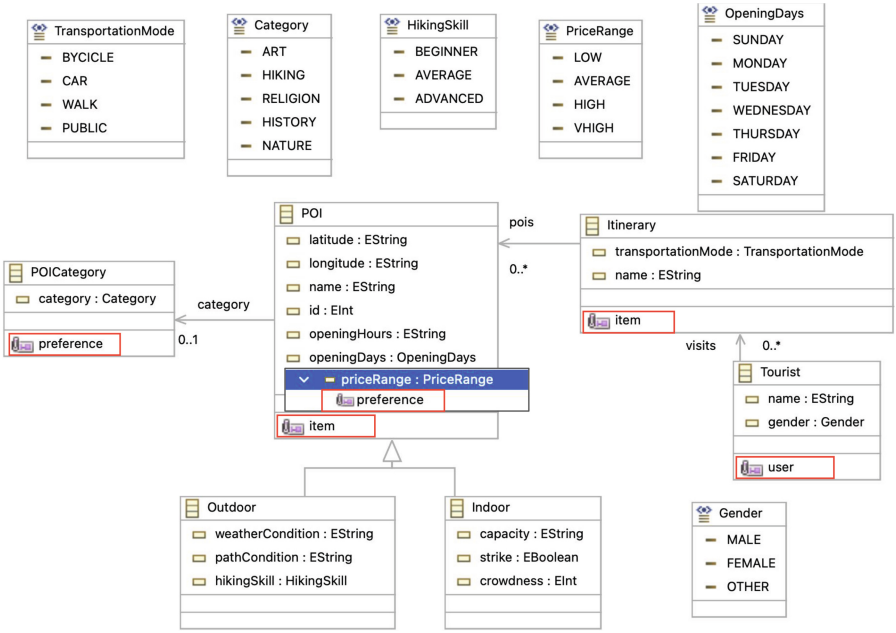


Fig. 4. Natural Park domain metamodel.

as the Itinerary and the preferences are specified on the attributes that can be considered, e.g., `priceRange` of the POI that will range from LOW to VERY-HIGH (VHIGH in the figure) as the enumeration specifies.

After we process this metamodel with the transformation T, the same will be modified as specified in Table 1, by adding supertypes to the involved meta-classes, e.g., POI and Tourist.

Figure 5 illustrates the weaving links between a natural park domain model and the RS in the tourism context through the weaving model, modelled at the level (m1) of Fig. 1 in step ❹. Furthermore, Fig. 5 shows an explicit user preference expressed on the available categories offered by the domain. Such explicit feedback can be embedded into the outcomes, thus adapting the recommendations using the explicit ratings from the matrix explained above.

In Fig. 6, we show how the proposed approach has been successfully integrated into an existing code generator, enabling the Picto view in **a)**. On top, a model conforms to the metamodel in Fig. 4 is shown in both textual (Flexmi) and XMI formats (tree-view based editor), whereas on the bottom, the Picto view shows a generated HTML view with the list of POIs. In Fig. 6 **c)** We show how the Picto editor shows a different ranked view of the POIs, based on the processed weaving model. Indeed, the list of POI is ranked for the user selected in the right dropdown menu (on the right). The application of the approach made the lightweight integration of domain-specific recommendations possible with a simple modification of the existing code generator. In Fig. 6 **b)** We highlight the

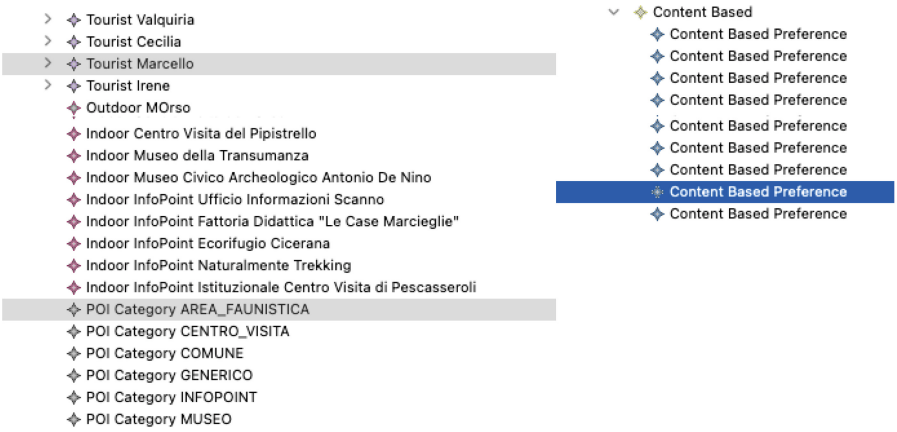


Fig. 5. An excerpt of the Natural Park Weaving model - Content-Based Preferences.

line of the code generator modified to obtain the desired result. Indeed, the line retrieving the POIs from the domain model instance was initially specified as s.pois, that is, the expression retrieving the POIs from the processed instance of RSDomain. This line is modified to navigate the output recommendation model and retrieve all the POIs (_item) through the weaving links, and then use the expression sortBy to rank the items based on the value.

This demonstrates that recommendations are tailored based on the highest scores and the user's preferred categories. Consequently, different recommendations will be generated depending on the category preference, highlighting the system's ability to dynamically adapt to various user interests and preferences.

Movie Domain. In Fig. 7 we report the second scenario aforementioned in Sect. 2, i.e., MovieLens.[5]. It is worth highlighting that this scenario counts with `Series` and `Movie`, and `Viewer`.

Following the same standard as in the previous scenario, the annotations in this metamodel trace the Viewer to the User in the RS model. Furthermore, the `Show` metaclass refers to the Item class in the RS model. After going through the transformation T, this metamodel will be altered to comply with the specification from Table 1 generating the final metamodel with all the correct references[6]

Figure 8 **a)** depicts the linkage between the shows/movies domain and the RS via the weaving model. In Fig. 8 it is possible to see the matrix of user-item interactions, although, this time, each row corresponds to a viewer, a show, and lastly, the score assigned to that given show by the viewer.

We demonstrated the feasibility of this approach, as it remains independent of the domain metamodel used. This is facilitated, reiterating, by the dotted conformance line depicted in Fig. 1, through which we enable the implementation

[5] Here, we also report directly the annotated metamodel processed by the transformation.

[6] Same as previous scenario.

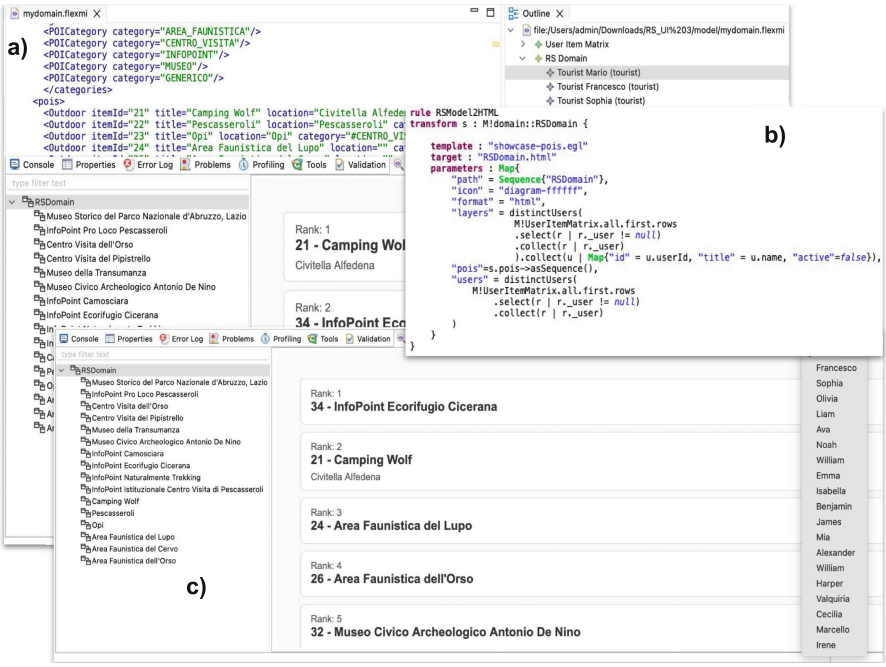


Fig. 6. Overview of the effect of applying the approach on an existing code generator.

of multiple inheritance with the RS metamodel by processing the annotations specified by the domain expert.

Running the Data Extractor on the input weaving models shown, we observe in Fig. 8 **b)** the Eclipse console displaying the results for this specific scenario. In the picture, we can see that the recommended item for the user is a blend of the highest score for the user's input and their preferred category, which is also user-defined. In this second example, we converted the output recommendation model into printed instructions in the Eclipse console, but also in this case, an existing code generator could be easily modified to embed the generated recommendations into a web-based view. This illustrates that recommendations will be tailored for different domains, respecting their specifications. Therefore, it saves time in development and makes the deployment of such systems more dynamic, underscoring the system's adaptability to different user interests and preferences through its multilevel modelling and generalizability. In the scope of this paper, the recommendations that were produced have been built on top of the Apache Mahout library. Nevertheless, it is our strong belief that the proposed approach can be adapted using different RS metamodels, such as the ones specified in prior works [17,32], to cover additional RS algorithms.

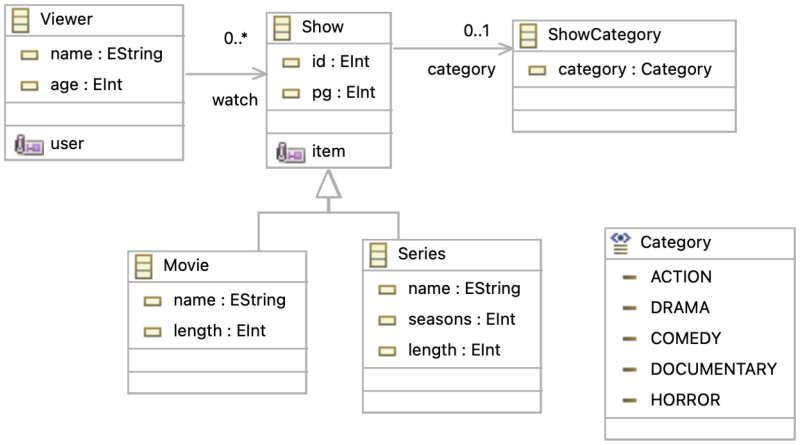


Fig. 7. MovieLens Domain Model.

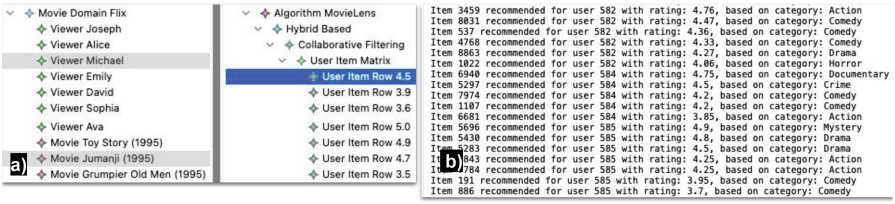


Fig. 8. Excerpt of MovieLens Weaving model - Collaborative Filtering Ratings (a) and generated console output (b).

Answer to *RQ*: By applying the modelling approach to the proposed scenarios, the results show that recommendations have been generated predicting over the given domain-specific models. Even though a specific library is bound to the implementation, the approach allows the usage of a more complex RS metamodel, thus easily supporting the specification of additional algorithms.

Threats to Validity. This section discusses threats that may hamper the results of our work. *Internal validity* concerns the expressiveness of the conceived metamodels, i.e., the RS metamodel may not be suitable to handle additional domains. To mitigate this, we exploit the EMF annotations that link generic filtering RS concepts to two different application domains, i.e., tourism and movie recommendations. Another possible threat is related to the EOL scripts employed to generate the data, as they are tailored for the conceived weaving model. In this respect, the scripts can be easily adapted by configuring a different metamodel during the EOL configuration.
Threats to *external validity* concern the generalizability of the results, i.e., the proposed approach can be applied only to filtering-based RS. Nonetheless, the EMF annotations and the weaving models can be reused using a more complex

RS metamodel, thus making the approach suitable to support additional RS algorithms, i.e., AI-based ones. While we acknowledge that the current version of the *Data Extractor* component supports the Apache Mahout implementation, the proposed framework can be generalized to support additional libraries. Another issue is represented by the conducted evaluation process, i.e., the obtained results may vary if additional scenarios are considered. To mitigate this, we rely on a state-of-the-art dataset in the RS domain, i.e., MovieLens, and a novel application domain, i.e., tourism recommendations developed in the RASTA project.

5 Related Works

UbiCARS [32] is an MDE framework that supports the specification of context-aware RSs in the e-commerce domain. The authors define a custom domain-specific modelling language (DSML) to represent relevant RS concepts, e.g. user feedback, purchase history, and rating.

`RECOLIBRY SUITE` [24] is a configuration framework built on top of an RS ontology. First, developers can select and configure a set of ready-to-use RS. Afterwards, a dedicated web service supports the deployment of the selected RS. Almonte et al. [2] develop DROID, an Eclipse plugin to configure, test, and run RSs for modeling activities. Given an initial set of models and metamodels, the framework allows users to run different state-of-the-art filtering RS by relying on RankSys Java library [48]. The system automatically computes accuracy metrics on the recommended items, showing the best algorithm for a given configuration. The same authors propose IRONMAN [1], a framework to configure and deploy different RS for supporting modelling tasks into a Sirius-based environment. Elliot framework [3] automatize the evaluation of RSs using a configuration file written in YAML. In such a way, non-expert users can easily train and test different RSs using different configurations. LEV4REC [17] exploits a dedicated feature model to configure RSs for software engineering tasks. After configuring high-level features, the system generates a coarse-grain model that can be refined within the Eclipse IDE. Afterwards, the corresponding source code is generated from the refined model using the Acceleo engine. The most relevant work to ours is [50]. In particular, the paper proposes the Lavoisier tool, an MDE-based approach based on textual domain-specific language (DSL) specifically conceived to automate the data management pipeline. In particular, the proposed language offers high-level primitives to transform any specific-domain data into tabular data, ready to be ingested by data analysis tools and frameworks.

Remark: To the best of our knowledge, prior works neglect to manage data handling flexibly. Compared with the above-mentioned tools, our approach offers a generic weaving model that links domain-specific concepts to RS ones. In addition, we provide an automated approach that can produce training data directly from the specified models, thus assisting developers in configuring two different types of filtering-based RSs. Lastly, AI-based RSs can be easily integrated by updating the code generator component.

6 Conclusion and Future Work

Motivated by the lack of tools and frameworks that facilitate the configuration of data sources in filtering-based recommender systems, this paper proposes a novel modelling approach based on weaving modelling and metamodel extension to support the deployment of filtering-based RSs tested on two well-known application scenarios, i.e. tourism and movies RS. We first define metamodels to represent domain concepts and generic components of two different types of RSs. Then, metamodel annotations were used to tie the domain-specific concepts to RSs generic component, assisting RSs designers in supporting different scenarios by reusing generic components. To assess the approach, we develop an initial prototype implementing model management operations and the Mahout RS library, showing that the conceived modelling artefacts can automate the mapping between the two above-mentioned scenarios and the RS metamodel and the generated recommendations can be used in existing code generators or as generated indications. In future works, we plan to enlarge the evaluation by considering additional algorithms, e.g., AI-based RSs, and distribute the weaving approach on a web-based UI that can be seen as a low-code platform for configuring RSs. In addition, we plan to conduct a user study in which participants can assess the expressiveness of the modelling framework in representing domain-specific concepts, also including a complete assessment of the performance of the specified RSs.

Acknowledgments. The work was partially supported by the research project RASTA: Realta Aumentata e Story-Telling Automatizzato per la valorizzazione di Beni Culturali ed Itinerari, Italian MUR PON Proj. ARS01 00540; the MUR (Italy) Department of Excellence 2023–2027 for the GSSI; the PRIN 2022 PNRR project "FRINGE: context-aware FaiRness engineerING in complex software systEms" grant n. P2022553SL; the EMELIOT national research project, which has been funded by the MUR under the PRIN 2020 program grant n. 2020W3A5FY.

References

1. Almonte, L., Garmendia, A., Guerra, E., de Lara, J.: Reuse and automated integration of recommenders for modelling languages. In: Proceedings of the 16th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2023, pp. 97–110. ACM (2023). https://doi.org/10.1145/3623476.3623523
2. Almonte, L., Pérez-Soler, S., Guerra, E., Cantador, I., de Lara, J.: Automating the synthesis of recommender systems for modelling languages. In: Proceedings of the 14th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2021, pp. 22–35. ACM (2021). https://doi.org/10.1145/3486608.3486905
3. Anelli, V.W., et al.: Elliot: a comprehensive and rigorous framework for reproducible recommender systems evaluation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2021, pp. 2405–2414. ACM (2021). https://doi.org/10.1145/3404835.3463245

4. Apache: Apache mahout. https://mahout.apache.org/
5. Balasubramanian, K., Godkhale, A., Lin, Y., Zhang, J., Gray, J.: Weaving deployment aspects into domain-specific models. Int. J. Softw. Eng. Knowl. Eng. **16**(03), 403–424 (2006)
6. Bamnote, G., Agrawal, S.: Evaluating and implementing collaborative filtering systems using apache mahout. In: 2015 International Conference on Computing Communication Control and Automation, pp. 858–862. IEEE (2015). https://doi.org/10.1109/iccubea.2015.171
7. Bastarrica, M.C., Simmonds, J., Silvestre, L.: Using megamodeling to improve industrial adoption of complex MDE solutions. In: Proceedings of the 6th International Workshop on Modeling in Software Engineering, ICSE 2014, pp. 31–36. ACM (2014). https://doi.org/10.1145/2593770.2593773
8. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**, 109–132 (2013)
9. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adap. Inter. **12**(4), 331–370 (2002)
10. Bézivin, J., Büttner, F., Gogolla, M., Jouault, F., Kurtev, I., Lindow, A.: Model transformations? Transformation models! In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MODELS 2006. LNCS, vol. 4199, pp. 440–453. Springer, Heidelberg (2006). https://doi.org/10.1007/11880240_31
11. Chen, K., Yang, Y., Chen, B., Hernández López, J.A., Mussbacher, G., Varró, D.: Automated domain modeling with large language models: a comparative study. In: 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 162–172. IEEE (2023). https://doi.org/10.1109/models58315.2023.00037
12. Del Fabro, M.D., Bézivin, J., Valduriez, P.: Weaving models with the eclipse AMW plugin. In: Eclipse Modeling Symposium, Eclipse Summit Europe, vol. 2006, pp. 37–44. Citeseer (2006)
13. Del Fabro, M.D., Valduriez, P.: Semi-automatic model integration using matching transformations and weaving models. In: Proceedings of the 2007 ACM Symposium on Applied Computing, SAC 2007, pp. 963–970. ACM (2007). https://doi.org/10.1145/1244002.1244215
14. Di Rocco, J., Di Ruscio, D., Di Sipio, C., Nguyen, P.T., Pierantonio, A.: Memorec: a recommender system for assisting modelers in specifying metamodels. Softw. Syst. Model. **22**(1), 203–223 (2022)
15. Di Rocco, J., Di Ruscio, D., Di Sipio, C., Nguyen, P.T., Rubei, R.: Development of recommendation systems for software engineering: the crossminer experience. Empirical Softw. Eng. **26**(4) (2021). https://doi.org/10.1007/s10664-021-09963-7
16. Di Ruscio, D., Malavolta, I., Muccini, H., Pelliccione, P., Pierantonio, A.: Developing next generation ADLs through MDE techniques. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, ICSE 2010, vol. 1, pp. 85–94. ACM (2010). https://doi.org/10.1145/1806799.1806816
17. Di Sipio, C., Di Rocco, J., Di Ruscio, D., Nguyen, P.T.: LEV4REC: a feature-based approach to engineering RSSEs. J. Comput. Lang. **78**, 101256 (2024). https://doi.org/10.1016/j.cola.2023.101256
18. Eclipse: Eclipse modeling framework. https://eclipse.dev/modeling/emf/
19. Eclipse: Epsilon. https://eclipse.dev/epsilon/
20. Egea, M., Rusu, V.: Formal executable semantics for conformance in the MDE framework. Innov. Syst. Softw. Eng. **6**(1–2), 73–81 (2009)
21. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. **5**(4), 1–19 (2015)

22. Hutchinson, J., Whittle, J., Rouncefield, M.: Model-driven engineering practices in industry: social, organizational and managerial factors that lead to success or failure. Sci. Comput. Program. **89**, 144–161 (2014). https://doi.org/10.1016/j.scico.2013.03.017
23. Iovino, L., Pierantonio, A., Malavolta, I.: On the impact significance of metamodel evolution in MDE. J. Object Technol. **11**(3), 3:1 (2012). https://doi.org/10.5381/jot.2012.11.3.a3
24. Jorro-Aragoneses, J.L., Díaz-Agudo, B., Recio-García, J.A., Jimenez-Díaz, G.: RecoLibry suite: a set of intelligent tools for the development of recommender systems. Autom. Softw. Eng. **27**(1–2), 63–89 (2020)
25. Kelly, S., Tolvanen, J.P.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley, Hoboken (2007)
26. Kolovos, D.S., Paige, R.F., Polack, F.: The epsilon transformation language. In: Vallecillo, A., Gray, J., Pierantonio, A. (eds.) ICMT 2008. LNCS, vol. 5063, pp. 46–60. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69927-9_4
27. Kolovos, D., de la Vega, A.: Flexmi: a generic and modular textual syntax for domain-specific modelling. Softw. Syst. Model. **22**(4), 1197–1215 (2022). https://doi.org/10.1007/s10270-022-01064-3
28. Korotaev, A., Lyadova, L.: Method for the development of recommendation systems, customizable to domains, with deep GRU network. In: Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, pp. 231–236. SCITEPRESS - Science and Technology Publications (2018). https://doi.org/10.5220/0006933302310236
29. Lara, J.D., Guerra, E., Cuadrado, J.S.: When and how to use multilevel modelling. ACM Trans. Softw. Eng. Methodol. **24**(2), 1–46 (2014)
30. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003). https://doi.org/10.1109/mic.2003.1167344
31. López-Fernández, J.J., Cuadrado, J.S., Guerra, E., de Lara, J.: Example-driven meta-model development. Softw. Syst. Model. **14**(4), 1323–1347 (2013)
32. Mettouris, C., Achilleos, A., Kapitsaki, G., Papadopoulos, G.A.: The UbiCARS model-driven framework: automating development of recommender systems for commerce. In: Kameas, A., Stathis, K. (eds.) AmI 2018. LNCS, vol. 11249, pp. 37–53. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03062-9_3
33. Paullier, A., Sotelo, R.: A recommender systems' algorithm evaluation using the lenskit library and movielens databases. In: 2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1–7. IEEE (2020). https://doi.org/10.1109/bmsb49480.2020.9379914
34. Pereira, R.S., Di Sipio, C., De Sanctis, M., Iovino, L.: On the need for configurable travel recommender systems: a systematic mapping study. In: 2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 56–63. IEEE (2024). https://doi.org/10.1109/seaa64295.2024.00057
35. Poernomo, I.: The meta-object facility typed. In: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC 2006, pp. 1845–1849. ACM (2006). https://doi.org/10.1145/1141277.1141710
36. Ricci, F.: Travel recommender systems. IEEE Intell. Syst. **17**, 55–57 (2002)
37. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 1–35. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_1

38. Rich, E.A.: Building and exploiting user models. Ph.D. thesis, USA (1979). aAI7925024
39. Robillard, M.P., Maalej, W., Walker, R.J., Zimmermann, T. (eds.): Recommendation Systems in Software Engineering. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-45135-5
40. Rose, L.M., Paige, R.F., Kolovos, D.S., Polack, F.A.C.: The Epsilon Generation Language, pp. 1–16. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69100-6_1
41. Salter, J., Antonopoulos, N.: Cinemascreen recommender agent: combining collaborative and content-based filtering. IEEE Intell. Syst. **21**(1), 35–41 (2006). https://doi.org/10.1109/mis.2006.4
42. Savary-Leblanc, M., Burgueño, L., Cabot, J., Le Pallec, X., Gérard, S.: Software assistants in software engineering: a systematic mapping study. Softw. Pract. Exp. **53**(3), 856–892 (2022). https://doi.org/10.1002/spe.3170
43. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 291–324. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_9
44. Schmidt, D.: Guest editor's introduction: model-driven engineering. Computer **39**(2), 25–31 (2006). https://doi.org/10.1109/mc.2006.58
45. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: Eclipse Modeling Framework. Pearson Education (2008)
46. Streviniotis, E., Chalkiadakis, G.: Multiwinner election mechanisms for diverse personalized Bayesian recommendations for the tourism domain. In: Proceedings of the 2022 Workshop on Recommenders in Tourism, RecTour, Seattle, WA, USA, vol. 22 (2022). https://ceur-ws.org/Vol-3219/
47. Stünkel, P., König, H., Lamo, Y., Rutle, A.: Multimodel correspondence through inter-model constraints. In: Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming, pp. 9–17. ACM (2018). https://doi.org/10.1145/3191697.3191715
48. Vargas, S., Castells, P.: Improving sales diversity by recommending users to items. In: Proceedings of the 8th ACM Conference on Recommender systems, RecSys 2014. ACM (2014). https://doi.org/10.1145/2645710.2645744
49. Vats, D., Sharma, A.: A collaborative filtering recommender system using apache mahout, ontology and dimensionality reduction technique. In: 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), pp. 1–12. IEEE (2022). https://doi.org/10.1109/accai53970.2022.9752604
50. de la Vega, A., García-Saiz, D., Zorrilla, M., Sánchez, P.: Lavoisier: a DSL for increasing the level of abstraction of data selection and formatting in data mining. J. Comput. Lang. **60**, 100987 (2020)
51. Walunj, S.G., Sadafale, K.: An online recommendation system for e-commerce based on apache mahout framework. In: Proceedings of the 2013 Annual Conference on Computers and People Research, SIGMIS-CPR 2013, pp. 153–158. ACM (2013). https://doi.org/10.1145/2487294.2487328
52. Weyssow, M., Sahraoui, H., Syriani, E.: Recommending metamodel concepts during modeling activities with pre-trained language models. Softw. Syst. Model. **21**(3), 1071–1089 (2022)

53. Yohannis, A., Kolovos, D., García-Domínguez, A., Candel, C.J.F.: Picto web: a tool for complex model exploration. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS 2022, pp. 56–60. ACM (2022). https://doi.org/10.1145/3550356.3559094
54. Zhu, G., et al.: Neural attentive travel package recommendation via exploiting long-term and short-term behaviors. Knowl.-Based Syst. **211**, 106511 (2021). https://doi.org/10.1016/j.knosys.2020.106511

The Role of Explanation Styles and Perceived Accuracy on Decision Making in Predictive Process Monitoring

Soobin Chae, Suhwan Lee, Hanna Hauptmann, Hajo A. Reijers, and Xixi Lu(✉)

Utrecht University, Utrecht, The Netherlands
{s.lee,h.j.hauptmann,h.a.reijer,x.lu}@uu.nl

Abstract. Predictive Process Monitoring (PPM) often uses deep learning models to predict the future behavior of ongoing processes, such as predicting process outcomes. While these models achieve high accuracy, their lack of interpretability undermines user trust and adoption. Explainable AI (XAI) aims to address this challenge by providing the reasoning behind the predictions. However, current evaluations of XAI in PPM focus primarily on *functional metrics* (such as fidelity), overlooking *user-centered* aspects such as their effect on *task performance* and decision-making. This study investigates the effects of explanation styles (feature importance, rule-based, and counterfactual) and perceived AI accuracy (low or high) on decision-making in PPM. We conducted a decision-making experiment, where users were presented with the AI predictions, perceived accuracy levels, and explanations of different styles. Users' decisions were measured both before and after receiving explanations, allowing the assessment of objective metrics (Task Performance and Agreement) and subjective metrics (Decision Confidence). Our findings show that perceived accuracy and explanation style have a significant effect.

Keywords: Explainable AI · Feature importance · Rule base explanations · Counterfactuals · User evaluation

1 Introduction

Predictive Process Monitoring (PPM) is a set of techniques that uses historical event logs to predict the future behavior of ongoing processes using machine learning [11]. These techniques enable predictions across various dimensions, such as forecasting the remaining time of a case, the next process step, and outcomes of ongoing processes [26]. By integrating these predictive capabilities into decision support systems, PPM can empower organizations to make data-driven decisions, enhancing efficiency and productivity across various domains, including healthcare, finance, and business operations.

S. Chae and S. Lee—Contributed equally as co-first authors.

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 39–56, 2025.
https://doi.org/10.1007/978-3-031-94571-7_3

Recent PPM techniques often use complex predictive models, such as deep learning, to achieve high prediction accuracy. This improved accuracy often comes at the expense of interpretability, as these "black box" models are inherently difficult for users to understand. This lack of interpretability and transparency can lead to hesitancy among users—such as decision-makers and process owners—to trust and adopt such systems [13]. This issue is particularly critical in the PPM domain, where predictions can directly influence crucial decisions, impacting individual cases and potentially entire workflows [18].

Explainable AI (XAI) has emerged as a promising solution to address this interpretability challenge. XAI aims to explain the "how" and "why" behind predictions while preserving the strong predictive performance of complex models [2]. In the PPM domain, recent advances have adapted XAI techniques, such as LIME [23] and DiCE [20], to explain process predictions [12,14,19,32] or evaluate predictive models [26]. By offering interpretable insights, these techniques have the potential to increase user trust and foster the adoption of PPM technologies.

Despite these advancements, user evaluations of XAI explanations remain scarce, particularly very few studies focus on the *effectiveness* of explanations rather than their *understandability* [1,9]. Much of the current XAI evaluation focuses on *functional* metrics (e.g., fidelity) rather than the *user-centered* effects of explanations. Many questions remain unanswered. For example, how do these explanations affect users' *task performance*, *agreement* with AI predictions, or *confidence* in their decisions?

In the PPM domain, only a few studies [12,25] have conducted user-centered evaluations, focusing on a single XAI technique and focusing on usability or understandability. There remains a critical gap in understanding the effects of different explanation styles and perceived AI accuracy on decision-making, such as task performance, agreement, and decision confidence.

In this paper, we aim to address this gap through an empirical, user-centric evaluation focusing the effectiveness. We investigate these two factors, namely (1) *explanation styles* and (2) *perceived AI accuracy*. Our motivation for focusing on these two factors is the following. Explanation styles vary in their logical structure and how they present information to users, influencing users' reasoning processes [5,34]. Perceived AI accuracy, on the other hand, impacts user trust and reliance, with prior research suggesting a relation between perceived correctness and the information users incorporate in decision-making [7,17]. We investigate the following research question: *How do (1) different explanation styles and (2) the perceived accuracy of AI predictions affect decision-making in PPM?*

Our evaluation was conducted in a decision-making context where participants determine whether to accept or reject loan applications based on AI predictions. Specifically, we trained a "black-box" predictive model on the loan application log from BPIC 2017 [8]. and used XAI techniques representing three distinct styles to generate corresponding explanations. To create the survey, we randomly selected *four cases* (two predicted correctly and two incorrectly) along with their predictions and explanations. The study involved 179 participants, divided into

independent groups. Each participant received an introduction, detailed descriptions of the task, and then the four selected cases (shown sequentially). For each case, participants were asked to make decisions both before and after being presented with the explanations. To evaluate the impact of these explanations, we measured a combination of objective metrics (*Task Performance* and *Agreement*) and a subjective metric (*Decision Confidence*). Our findings contribute to a deeper understanding of how users interact with XAI explanations, providing insights for designing more effective decision-support systems in PPM.

The remainder of this paper is structured as follows: Sect. 2 discusses the current XAI techniques in the PPM domain and their evaluation methods. Section 3 presents the research method. Section 4 covers the evaluation of the developed system. Finally, Sect. 5 summarizes our key findings and discusses future work.

2 Related Work

Recently, explainability approaches have also been adapted and investigated in the field of PPM [3,4,14,16] Only a few studies in the PPM domain are concerned with evaluating the explanations generated with XAI techniques. As the importance of explaining PPM results gains recognition, the evaluation of these explanations is expected to gain more interest. Explainability evaluation can be categorized into three types [9]: Function-grounded, Application-, and Human-evaluations. In Table 1, relevant studies are classified based on their evaluation approaches and whether they compare XAI methods.

Function-grounded evaluation assesses explanations based on their inherent characteristics, such as stability and fidelity. *Stability* refers to the consistency of explanations generated for the same data sample under identical conditions [33]. They proposed metrics to evaluate the stability of the top-K feature subset and their respective weights across multiple explanations for certain process instances [31]. *Fidelity*, on the other hand, pertains to the ability of XAI methods to accurately mimic the behavior of the explained ML model in the vicinity of the explained process instance. Velmurugan et al. [30] introduced an approach to evaluate local XAI methods for their internal fidelity, which compares the decision-making process of the explainer proxy model with the explained black-box model. Among the Function-grounded evaluation studies in the PPM domain, Stevens et al. [28] and Elkhawaga et al. [10] proposed evaluation approaches for various XAI methods applied to PPM results. Stevens et al. introduced four out-of-the-box metrics from relevant XAI evaluation research and applied them to different attributes of process mining data. Elkhawaga et al. [10] proposed an approach for evaluating the consistency of XAI methods. While useful, we lack insights into how these explanations affect user's decision-making.

Only two studies [12,25] focused on *user evaluations*, specifically *application-grounded* evaluation. Rizzi et al. [25] are among the first to investigate whether users understand the explanation plots. Instead of applying actual XAI methods, they generate three levels of different plots in event, trace, and event log

Table 1. Explainability Evaluation Approaches in PPM

Year	Ref.	Evaluation Metric	Function grounded	Application grounded	Human Evaluation	XAI Methods Comparison
2020	Velmurugan et al. [30]	Fidelity	✓			✓
2021	Velmurugan et al. [31]	Stability	✓			✓
2022	Huang et al. [15]	Fidelity Domain Knowledge	✓			
2023	Stevens et al. [28]	Parsimony Functional Complexity Importance Ranking Correlation Level of Disagreement	✓			✓
2024	El-Khawaga et al. [10]	Consistence	✓			✓
2022	Rizzi et al. [25]	Understandability		✓		
2023	Galanti et al. [12]	Accuracy Perceived Task Difficulty Usability		✓		

levels. While the study involved participants from both the PPM and ML fields, comprehension and usage levels of explanations varied based on domain knowledge and experience. However, the study relied on qualitative evaluation with a limited number of participants without employing a consolidated user-interface evaluation methodology. Galanti et al. [12] proposed a framework to evaluate the understanding and comfort level of process analysts with results from an explainable predictive monitoring framework. Their evaluation focused on accuracy in task execution, perceived task difficulty, usability, and user experience dimensions. Regarding the comparison of XAI methods, no user study compares the effectiveness of different XAI styles in user evaluation in the PPM domain.

We identified two key gaps that motivate our work. First, there are limited comparative user evaluations in the PPM domain. Existing user evaluations of XAI techniques for PPM primarily focus on individual XAI methods or qualitative evaluations with limited generalizability. Second, there is a lack of research that studies the effect of different explanation styles on decision-making, not to mention the perceived AI accuracy. Existing research primarily focuses on the understandability and usability of a single explanation style. There is no systematic investigation into how different explanation styles and the accuracy of a prediction model affect a user's decision process.

Outside the area of PPM, there have been studies that show how different explanation styles and AI accuracies can impact the user's decision confidence, task performance, and agreement. In [6], two explanation styles are studied in the domain of stock market trading. The study shows that both rule-based and feature-importance explanations improved task performance, but only in cases of high AI accuracy. A follow-up paper [5] on text and image classification tasks showed that the domain of the prediction model can change the way explanation styles impact the user's task performance, indicating that a new assessment in the PPM domain is required. Both papers further propose expanding such comparisons by including counterfactual examples as an explanation technique.

Given these gaps, this paper aims to conduct an empirical user evaluation to investigate the effect of different explanation styles and perceived AI accuracy on decision-making.

3 Research Method

In this section, we outline the research methods for conducting the empirical user evaluation. The section is organized as follows: we introduce the evaluation objectives, describe the experimental design, define the hypotheses and analytical approach, and detail the overall experimental procedure.

3.1 Evaluation Objectives

The objective of this study is to examine the *effect* of explanation styles and the *perceived AI accuracy* on decision-making within the context PPM. Specifically, the evaluation is structured around the following objectives:

1. To investigate the effect of perceived AI accuracy on task performance, agreement with AI predictions, and decision confidence.
2. To investigate the effect of explanation styles on task performance, agreement, and decision confidence by comparing results before and after the provision of explanations.

3.2 Experimental Design

The experiment was designed to address the evaluation objectives outlined in the previous section. As illustrated in Fig. 1, a two-phase decision-making setup was implemented to examine the effects of perceived accuracy and explanation styles. First, participants were randomly assigned into two groups; the perceived AI accuracy was manipulated depending on the group, i.e., the high-accuracy group was shown a 96% accuracy, whereas the low-accuracy group was shown a 63% accuracy. This division allowed us to assess the effect of perceived accuracy on participants' initial decisions (EXP1). Within each group, participants were further assigned to one of three commonly used explanation-style subgroups: Feature Importance (FI), Rule-based, or Counterfactual (CF) [22], thus in total, 6 independent groups. This setup facilitated comparisons of pre- and post-explanation decision-making across explanation styles.

Each participant is randomly assigned to follow one of these six branches in Fig. 1 (e.g., High-accuracy and FI-based style) and sees four cases sequentially: two cases with correct predictions and two cases with incorrect predictions. For each case, participants' *decisions*, i.e., *whether they agree with or are against the AI decision*, were recorded at two points: **before** explanations were provided (initial decisions) and **after** receiving the assigned explanation style (post-explanation decisions). This design enabled both within-group (pre- versus post-explanation) and cross-group (accuracy and explanation style) comparisons.

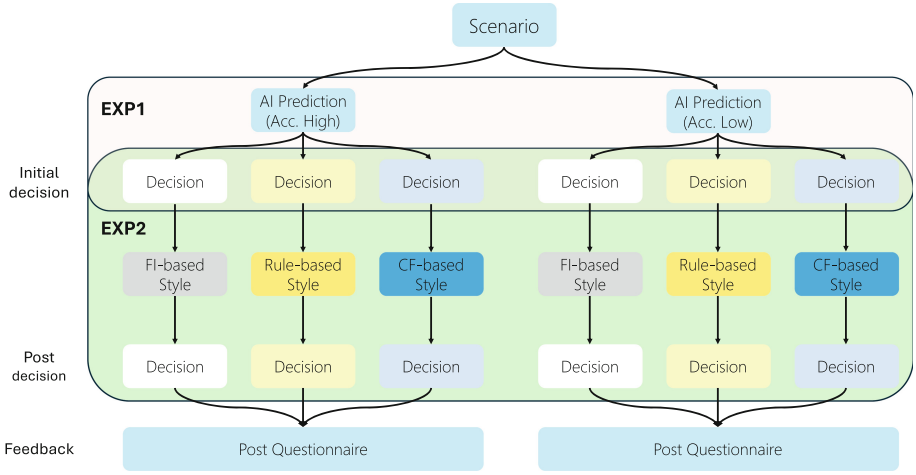


Fig. 1. Experiment Setting.

Two *independent variables* were recorded in this experiment: (1) **Perceived AI Accuracy**: high (96%) or low (63%), determining the accuracy level participants were exposed to; (2) **Explanation styles**: the assigned explanation style, categorized as Feature Importance, Rule-based, or Counterfactual.

To evaluate the effect of explanation styles and perceived AI accuracy on decision-making, the following three *dependent variables* were measured for each participant across the four cases, both *before* and *after* explanations:

1. **Task performance**: The number of correct decisions participants make across the four cases, measured separately before and after.
2. **Agreement**: The number of decisions that align with AI predictions, measured separately for the four cases before and after.
3. **Decision Confidence**: Participants' confidence in their decision-making, rated on a 5-point Likert scale ranging from 1 (completely unsure) to 5 (extremely confident), also measured before and after across the four cases.

After completing the decision-making task, participants filled in a post-experiment questionnaire to indicate their subjective opinions regarding the effectiveness of the explanations. This included satisfaction and perceived simplicity, as well as rankings of the explanations they relied on most. Participants' educational background and experience with process mining or XAI were also collected to explore their potential influence on explanation effectiveness.

3.3 Training Black-Box Model and Generating Explanations

We use a real-life log to generate realistic explanations of different styles, train an outcome prediction model, and generate explanations. We select the BPIC

2017 event log [8], which is a real-life event log that records the loan application process of a Dutch financial institution. This dataset was selected for two primary reasons. First, it is a widely used dataset in the PPM and explainable PPM domains (e.g., [3,16,29]), showing its applicability for generating process predictions and explanations, ensuring the feasibility of this study. Second, the loan application process provides a familiar and relatable context, making it accessible for participants, even those without prior knowledge of PPM.

To prepare the data for model training, we followed established preprocessing steps [21,35]. Figure 2 shows an overview of our PPM workflow. We used prefix extraction and state-based bucketing with the activity *O_Returned* as the mile-stone state, representing the decision point. We used aggregation encoding, counting all events up to the mile-stone state for each case, and extracting the relevant case attributes. Additionally, we generated three features (i.e., income, employment status, credit score, and age). Half of the feature values were randomly generated to introduce variability, independently of the outcome labels. The other half was generated based on the rules to ensure some useful features when generating explanations. We then trained a Random Forest model as the black-box model to predict the process outcome, determining whether a loan application would be accepted, canceled, or rejected. The model achieved an overall accuracy of 0.85.

To generate explanations for the black-box model, we employed three established XAI techniques, each of them representing a major explanation style used in PPM [14,15,18],:

- Feature Importance (FI)-based explanations: LIME [23], which identify variables influencing a prediction and highlight the most critical features.
- Rule-based explanations: Anchor [24] which provide interpretable if-then rules that describe decision boundaries.
- Counterfactual-based explanations: DiCE [20], which offer hypothetical scenarios to illustrate how a different outcome could occur under changed inputs.

Examples of each explanation style are shown in Figs. 3, 4 and 5. The full experimental setup, including detailed preprocessing steps and model configurations, is made available at https://github.com/ghksdl6025/evaluating_explanation_styles.

3.4 Survey Procedure

Before distributing the survey, we conducted pilot testing with five individuals, representing a mix of those with and without process mining knowledge and those with and without a Business IT background. This pilot test aimed to estimate the average time required to complete the task and assess the task difficulty. Participants without these backgrounds took approximately 20 min to complete the survey. Recognizing the varying levels of knowledge and anticipating that a long survey duration might reduce the quality of responses, we decided to reduce the number of tasks. Initially, the survey included five scenarios. Based

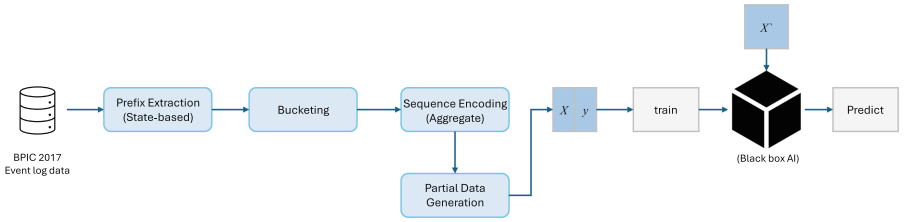


Fig. 2. PPM workflow.

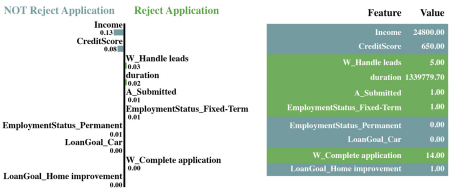


Fig. 3. Feature example (LIME)

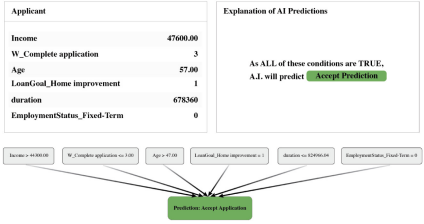


Fig. 4. Rule example (Anchor)

on the pilot test feedback, we reduced this to four scenarios, focusing only on *accept* and *reject* predictions, each with one correct and one incorrect predictions. After these adjustments, the median completion time for the pilot survey was approximately 10 min.

The survey distribution was divided into multiple phases. The first batch consisted of approximately 20 Business Informatics master's students at Utrecht University attending a Process Mining lecture. The second batch included other master's students from backgrounds related to Business and IT, including Business Informatics. The third batch consisted of PhD researchers in the process mining field. The final batch was recruited from the Prolific platform[1] to fill in the required number of participants. In total, we gathered 222 participants. After excluding responses that are incomplete or the completion time under 5 min, we

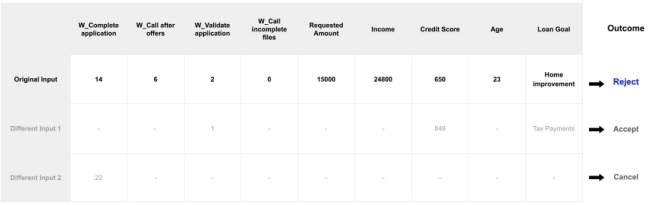


Fig. 5. CF example (DiCE).

[1] Copyright ©[2024] Prolific. https://www.prolific.com.

Table 2. Participant demographics (N = 179)

Characteristics		*N*	%
Education (highest completed)	High School	3	1.7
	Bachelor or equivalent	98	54.1
	Master or equivalent	69	38.1
	Ph.D. or higher	9	5
STEM Background	Yes	138	22.7
	No	41	76.2
Process Mining Experience	Never worked with it	107	59.1
	Less than 1 year	41	22.7
	1–2 years	17	9.4
	2–3 years	6	3.3
	3–5 years	4	2.2
	5+ years	4	2.2
XAI Experience	Never worked with it	107	59.1
	Less than 1 year	32	17.7
	1–2 years	24	13.3
	2–3 years	6	3.3
	3-5 years	8	4.4
	5+ years	2	1.1

retained data from 179 participants. The median time to complete the survey for these qualified participants was 566.5 s, approximately 9–10 min.

Participants were randomly assigned to one of six groups (approximately 30 participants per group) using the Qualtrics survey tool[2] All participants followed the same procedure, which included the following stages:

1. **Introduction**: Participants received an overview of the study objectives and the loan application process to provide context for the experiment.
2. **Informed Consent**: A consent form was provided, detailing the study's purpose, participants' rights, and data confidentiality measures.
3. **Experiment**: Participants completed four decision-making tasks, making predictions both *before* and *after* receiving explanations.
4. **Post Questionnaire**: Participants rated their satisfaction and their (perceived) simplicity with the explanations (1–5 scale). Open-ended responses captured additional feedback on challenges and suggestions for improvement.
5. **Demographic Questionnaire**: Demographic data such as education level, STEM background, and experience with XAI and Process Mining were collected for analysis. Table 2 summarizes the demographic distribution of the 179 participants.

An example of the full survey is publicly available[3]. The survey is published at https://survey.uu.nl/jfe/form/SV_789q74Z184LzGJ0.

[2] Copyright ©[2024] Qualtrics. https://www.qualtrics.com.
[3] see https://github.com/ghksdl6025/evaluating_explanation_styles.

4 Results

This section presents the quantitative results obtained from the participants and details the analyses performed. The analyses were conducted using the Python Statsmodels package [27], which supports statistical tests. The code for the statistical tests is publicly available (see footnote 3).

4.1 Descriptive Statistics

Table 3 provides a summary of the results for *task performance*, *agreement*, and *decision confidence* across the three explanation styles and the two perceived accuracy levels.

As listed in Table 3, task performance varied within the high-accuracy group, depending on the type of explanation provided. The survey participants who received Feature importance (FI) explanations scored an average of 2.14 out of 4 before the explanation. This means that out of 4 decisions made before showing the explanation, 2.14 are made correctly on average across 29 participants (as the ground truth regardless of the prediction). This number slightly increased to 2.21 after showing the explanation. However, the standard deviations are rather high (0.68 before vs 0.71 after). Those who received Rule-based explanations scored 2.27 before and 2.23 after, slightly higher than FI. Those who received Counterfactual explanations had the lowest task performance in the group, with averages of 1.70 before and 1.85 after the explanation.

In the low-accuracy group, we observed the opposite for counterfactual, of which the task performance was the highest among the three groups, 2.39 before the explanation, and 2.84 after the explanation (with an increase of 0.45). Interestingly, participants in the low-accuracy group showed more noticeable improvements in their performance across all explanation styles after seeing the explanation. In the low-accuracy group, a mild increase of 0.32, 0.13, 0.45 respectively for FI, Rule, and CF, whereas only a slight increase of 0.07, -0.04, 0.15 in the high-accuracy group for the three styles.

In the case of *Agreement*, the results are more mixed, as listed in Column 5 and 6 in Table 3. The participants in the high-accuracy group who received Counterfactual explanations had the highest agreement with AI predictions, with

Table 3. Before and After Explanation: Task Performance, Agreement, Decision Confidence among Accuracy and Exp. Styles

Accuracy	Exp. Styles	Task Performance			Agreement			Decision Confidence		
		Before	After	Diff.	Before	After	Diff.	Before	After	Diff.
High	FI	2.14 (0.68)	2.21 (0.71)	0.07	3.17 (1.02)	2.90 (1.18)	−0.28	3.57 (0.88)	3.61 (0.91)	0.04
	Rule	2.27 (0.89)	2.23 (0.67)	−0.03	3.13 (0.96)	3.30 (0.90)	0.17	3.88 (0.63)	4.07 (0.65)	0.19
	CF	1.70 (0.53)	1.85 (0.59)	0.15	3.59 (0.78)	3.41 (0.78)	−0.19	4.02 (0.53)	3.97 (0.54)	−0.05
Low	FI	2.13 (0.66)	2.45 (0.66)	0.32	3.42 (0.79)	3.16 (0.68)	−0.26	3.78 (0.93)	3.87 (0.75)	0.09
	Rule	2.39 (0.87)	2.52 (0.88)	0.13	3.29 (0.89)	3.23 (0.97)	−0.06	3.73 (0.82)	3.81 (0.84)	0.08
	CF	2.39 (0.66)	2.84 (0.72)	0.45	3.16 (0.88)	3.48 (0.76)	0.32	3.78 (0.54)	3.66 (0.62)	−0.12

FI = Feature importance, CF = Counterfactual, Diff. = After - Before.

3.59 before and 3.41 after showing the explanation. In the low-accuracy group, participants with Counterfactual explanations showed an increase in agreement scores (from 3.16 to 3.48), reaching the highest agreement among the three styles. Another interesting observation is that after showing FI explanations, the agreement score decreased in both high and low-accuracy groups, while for the rule-based explanation, the results are mixed.

An interesting observation is that the average agreement score is consistently higher than the task performance, meaning out of the four cases, users agree more than three times with the prediction. Since two of these predictions are incorrect, these impact their task performance. This suggests that the users lost performance due to overreliance on the prediction.

Regarding *Decision confidence* (see the last two columns in Table 3) scores varied across explanation styles and accuracy levels. In the high-accuracy group, Rule-based explanations led to the highest decision confidence after the explanation (4.07), followed by Counterfactual (3.97), and FI (3.61). In comparison, the low-accuracy group showed lower decision confidence scores, except for FI. When comparing the score before and after the explanation, the decision confidence increased for rule-based and FI, except for Counterfactual.

4.2 Hypotheses Result

In this section, we present the results of our hypothesis testing. Table 4 outlines the variables and statistical methods applied for each hypothesis.

EXP1: Effect of Perceived AI Accuracy on Initial Task Performance (H1), Agreements (H2), and Decision Confidence (H3). We test the three hypotheses to measure the effect of perceived accuracy on decision-making in the before explanation setup. Independent sample t-tests were used for all three hypotheses.

H1: *Effect of Perceived Accuracy on Task Performance.* H1 investigated whether perceived AI accuracy levels influence initial task performance. The independent variable is the accuracy group (high vs. low), and the dependent variable is the average task performance of the two groups prior to explanations.

Table 4. Statistical Analysis and Variables for Each Hypothesis

Independent Variable	Dependent Variable	Statistical test
Perceived AI Acc. (**EXP1**)	Before explanation TP (**H1**)	Independent t-test
	Before explanation Ag. (**H2**)	
	Before explanation DC (**H3**)	
Exp. Styles (**EXP2**)	Before vs. After explanation TP (**H4**)	Paired t-test
	Before vs. After explanation Ag. (**H5**)	
	Before vs. After explanation DC (**H6**)	

TP = Task Performance, Ag. = Agreement, DC = Decision Confidence.

The test yielded a p-value of 0.026 and an effect size of 0.336 (Cohen's d), indicating a small to medium effect. The results suggest a significant difference in task performance between the high-accuracy and low-accuracy groups. Interestingly, participants in the high-accuracy group performed *worse* than those in the low-accuracy group, with a mean difference of -0.2558. This finding indicates that lower perceived AI accuracy is associated with *higher initial task performance* before explanations are provided.

H2: *Effect of Perceived Accuracy on Agreement.* For H2, we investigated whether perceived AI accuracy influences participants' agreement with AI predictions. The analysis showed no significant difference between the high-accuracy and low-accuracy groups, with a p-value of 0.998 and a negligible mean difference of 0.0349. These findings suggest that perceived AI accuracy does not significantly affect agreement with AI predictions in the pre-explanation setup.

H3: *Effect of Perceived Accuracy on Decision Confidence.* H3 evaluates the impact of perceived AI accuracy on participants' confidence in their decisions. The analysis yielded a p-value of 0.638, with a mean difference of 0.0698, indicating no significant difference in decision confidence between the two accuracy groups. These results suggest that perceived AI accuracy does not influence participants' confidence in their decisions before explanations are provided.

EXP2: Effect of Explanation Provision on Task Performance (H4), Agreements (H5), and Decision Confidence (H6). This section analyzes how the provision of explanations impacts decision-making, comparing pre- and post-explanation decisions. The hypotheses focus on how different explanation styles influence decision-making across high- and low-accuracy groups. Paired t-tests were conducted to assess these effects. The statistical results are summarized in Table 5.

H4: *Effect of Explanation Provision on Task Performance.* H4 examines whether providing explanations influences participants' task performance. The indepen-

Table 5. P-Values (and Effect Sizes) for Paired t-Test Results: Changes in Task Performance, Agreement, and Decision Confidence

Target Groups	Task performance (H4)	Agreement (H5)	Decision Confidence (H6)
Feature Importance (FI)	0.096 (0.218)	0.025 (0.296)	0.396 (0.110)
Rule-based (Rule)	0.689 (0.051)	0.717 (0.047)	0.135 (0.194)
Counterfactual (CF)	0.002 (0.425)	0.497 (0.090)	0.218 (0.164)
High Acc. FI	0.663 (0.082)	0.187 (0.251)	0.655 (0.084)
High Acc. Rule	0.839 (0.037)	0.433 (0.145)	0.076 (0.336)
High Acc. CF	0.294 (0.206)	0.259 (0.222)	0.557 (0.114)
Low Acc. FI	0.077 (0.329)	0.043 (0.379)	0.476 (0.130)
Low Acc. Rule	0.489 (0.126)	0.712 (0.067)	0.582 (0.100)
Low Acc. CF	0.002 (0.625)	0.086 (0.318)	0.285 (0.196)

Note: Yellow cells indicate significant results where p-values are below 0.05.

dent variable is the explanation provision, and the dependent variable is task performance.

Grouping the data by explanation styles (merging the low- and high-accuracy groups), the results of Counterfactual explanations show a significant increase in task performance. The paired t-test showed a t-value of -3.236 (df = 57), with p = 0.002 and a medium effect size of 0.425. The average increase in task performance is 0.310. For FI-based and Rule-based styles, no significant changes were observed.

To confirm these results, we conducted further analysis, testing the high and low-accuracy groups separately (see the last six rows of Table 5). The test results show a significant improvement in task performance for the Counterfactual-based style, particularly in the low-accuracy group, with a p-value of 0.002 and a medium effect size of 0.625. This was also observed in the descriptive statistics in Table 3.

These results allow us to reject the null hypothesis ($H4_0$), which posits that explanations do not influence task performance. The data support the alternative hypothesis ($H4_1$), indicating that **Counterfactual-based explanations significantly improve task performance**, particularly within the low-accuracy group, highlighted by a medium effect size.

H5: *Effect of Explanation Provision on Agreement.* H5 investigates whether explanations influence participants' agreement with AI predictions. The independent variable is explanation provision, and the dependent variable is agreement before and after explanations.

Across all participants, the FI-based style showed a significant decrease in agreement, with a mean difference of 0.27 (p = 0.025) and a small to medium effect size of 0.296. No significant changes in agreement were observed for the Rule-based or Counterfactual-based styles, as their p-values exceeded 0.05.

When testing the high and low-accuracy groups separately, the FI-based style in the low-accuracy group remains significant, showing a decrease of -0.26 in agreement and a p-value of 0.043. The test results in the high-accuracy group became insignificant, with a p-value of 0.187. No significant changes in agreement were observed for the Rule-based or Counterfactual-based styles, as their p-values exceeded 0.05. These findings suggest that **FI-based explanations may reduce agreement in low-accuracy scenarios**, but explanations generally have limited influence on agreement in high-accuracy scenarios.

H6: *Effect of Explanation Provision on Decision Confidence.* H6 evaluates whether explanations influence participants' decision confidence. The independent variable is the explanation style, and the dependent variable is the change in confidence before and after explanations.

Across all participants, no significant changes in decision confidence were observed for any explanation style. The p-values were as follows: FI-based (p = 0.396), Rule-based (p = 0.135), Counterfactual-based (p = 0.218). Effect sizes were small, indicating negligible impact.

When analyzing the high and low accuracy groups separately, the results were non-significant for both groups: high accuracy (FI-based p=0.655, Rule-based

p=0.076, Counterfactual-based p=0.557); low accuracy (FI-based p=0.476, Rule-based p=0.582, Counterfactual-based p= 0.285). Separate analysis for the high- and low-accuracy groups yielded non-significant results for all explanation styles. Based on these results, we fail to reject the null hypothesis ($H6_0$), which states that explanations do not influence participants' decision confidence.

Satisfaction and Simplicity. Users' subjective evaluations of explanations regarding satisfaction and simplicity revealed that the Rule-based explanation style achieved the highest satisfaction scores and was rated the easiest to understand across both high- and low-accuracy groups (averages of 3.80 and 3.61, respectively). The Feature Importance (FI) style ranked second, with satisfaction scores of 3.66 and 3.39. In contrast, the Counterfactual explanation style received the lowest satisfaction ratings (3.41 and 3.19) and was perceived as slightly more difficult to understand. Interestingly, despite their effectiveness in enhancing task performance (as discussed previously), counterfactual explanations presented challenges in terms of user satisfaction and simplicity. These findings suggest that Rule-based explanations balance satisfaction and simplicity, while Counterfactual explanations may trade off user satisfaction and simplicity for performance gains.

4.3 Discussion

Interestingly, task performance was consistently better in the low-accuracy group compared to the high-accuracy group, even when the same explanation styles were applied. This counterintuitive result may indicate that users in the low-accuracy group engaged more critically with the provided explanations. With the high-accuracy label, users might have trusted the AI predictions too readily, relying less on the explanations to guide their decisions. This overreliance may have contributed to lower task performance in the high-accuracy group. This may suggest that it is important to foster appropriate skepticism among users when interacting with AI systems, especially in cases where predictions are uncertain or may be incorrect.

Furthermore, the greater effect of explanations in the low-accuracy group (i.e., the improvement in task performance before and after the explanation) also aligns with the hypothesis that users are more likely to scrutinize explanations when they perceive the model as less reliable. In contrast, users in the high-accuracy group may view explanations as supplementary or only consult them to resolve doubts. This behavioral difference underscores the need to design explanations that can effectively support users in both scenarios, ensuring the explanations are used as intended.

Our findings also show that counterfactual explanations significantly enhance task performance, particularly in the low-accuracy group. This suggests that counterfactual-based explanations provide insights that enable users to more effectively identify incorrect decisions.

It is important to note that the model's accuracy was a label rather than an inherent characteristic of the underlying system, which remained the same

across conditions. This design choice suggests that perceived accuracy significantly influences how users interact with explanations. The stronger impact of explanations in the low-accuracy group underscores the importance of managing user perceptions when presenting AI systems, particularly in contexts where interpretability and trust are critical.

It is worth highlighting an important consideration regarding perceived AI accuracy. Initially, we explored the idea of using actual model performance by training two separate models-one with high accuracy and one with low accuracy. However, we found that the explanations generated by XAI techniques differed significantly between the two models. This shifted the focus of our study toward the content of the explanations (i.e., which features best explain the prediction) rather than the comparison of explanation styles. To ensure a fair comparison across groups, we chose to use a single model while manipulating only the perceived accuracy. Future work could investigate how true model accuracy affect user trust and decision-making.

5 Conclusion

This study aimed to evaluate the *effect* of explanation styles (i.e., Feature importance-based, Rule-based, and Counterfactual-based) and perceived AI accuracy on decision-making within the predictive process monitoring (PPM) domain. The evaluation was conducted in a decision-making scenario where participants assessed whether to accept or reject a loan application based on the AI's predictions. The effects were measured across three metrics: task performance, agreement, and decision confidence.

The findings revealed that perceived AI accuracy significantly influenced initial task performance, with lower perceived accuracy associated with higher initial task performance. Among the explanation styles, Counterfactual explanations proved to be particularly effective in enhancing task performance and agreement. In contrast, Feature importance explanations showed the lowest levels of agreement across all styles.

Rule-based explanations stood out in terms of user satisfaction and perceived simplicity, as they achieved the highest ratings. These characteristics also translated into the highest decision confidence. However, despite their positive reception, Rule-based explanations did not appear to significantly improve task performance. These results suggest there may be no correlation between the perceived satisfaction of an explanation style and its objective effectiveness in enhancing decision-making outcomes.

Building on these findings, future work could focus on conducting application-grounded evaluations involving domain experts to further validate the effectiveness of these explanation styles in real-world contexts. Additionally, future research could consider more advanced explainable predictive process monitoring techniques in the evaluation. Exploring novel explanation styles, such as human language-based explanations powered by Large Language Models (LLMs), represents another promising avenue for improving both interpretability and user engagement.

References

1. Anjomshoae, S., Najjar, A., Calvaresi, D., Främling, K.: Explainable agents and robots: results from a systematic literature review. In: Elkind, E., Veloso, M., Agmon, N., Taylor, M.E. (eds.) Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2019, Montreal, QC, Canada, 13–17 May 2019, pp. 1078–1088. International Foundation for Autonomous Agents and Multiagent Systems (2019)
2. Arrieta, A.B., et al.: Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion **58**, 82–115 (2020)
3. Buliga, A., Francescomarino, C.D., Ghidini, C., Maggi, F.M.: Counterfactuals and ways to build them: evaluating approaches in predictive process monitoring. In: CAiSE. Lecture Notes in Computer Science, vol. 13901, pp. 558–574. Springer, Cham (2023)
4. Buliga, A., et al.: Uncovering patterns for local explanations in outcome-based predictive process monitoring. In: BPM. Lecture Notes in Computer Science, vol. 14940, pp. 363–380. Springer, Cham (2024)
5. Cau, F.M., Hauptmann, H., Spano, L.D., Tintarev, N.: Effects of AI and logic-style explanations on users' decisions under different levels of uncertainty. ACM Trans. Interact. Intell. Syst. **13**(4), 22:1–22:42 (2023)
6. Cau, F.M., Hauptmann, H., Spano, L.D., Tintarev, N.: Supporting high-uncertainty decisions through AI and logic-style explanations. In: Proceedings of the 28th International Conference on Intelligent User Interfaces, IUI 2023, Sydney, NSW, Australia, 27–31 March 2023, pp. 251–263. ACM (2023). https://doi.org/10.1145/3581641.3584080
7. Cecil, J., Lermer, E., Hudecek, M.F., Sauer, J., Gaube, S.: Explainability does not mitigate the negative impact of incorrect AI advice in a personnel selection task. Sci. Rep. **14**(1), 9736 (2024)
8. van Dongen, B.: BPI challenge 2017 (2017)
9. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. CoRR abs/1702.08608 (2017)
10. El-Khawaga, G., Elzeki, O.M., Abu-Elkheir, M., Reichert, M.: Why should I trust your explanation? An evaluation approach for XAI methods applied to predictive process monitoring results. IEEE Trans. Artif. Intell. **5**(4), 1458–1472 (2024)
11. Francescomarino, C.D., Ghidini, C.: Predictive process monitoring. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook, Lecture Notes in Business Information Processing, vol. 448, pp. 320–346. Springer, Cham (2022)
12. Galanti, R., et al.: An explainable decision support system for predictive process analytics. Eng. Appl. Artif. Intell. **120**, 105904 (2023)
13. Hamm, P., Klesel, M., Coberger, P., Wittmann, H.F.: Explanation matters: an experimental study on explainable AI. Electron. Mark. **33**(1), 17 (2023)
14. Hsieh, C., Moreira, C., Ouyang, C.: Dice4el: interpreting process predictions using a milestone-aware counterfactual approach. In: Ciccio, C.D., Francescomarino, C.D., Soffer, P. (eds.) 3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, 31 October–4 November 2021, pp. 88–95. IEEE (2021)
15. Huang, T., Metzger, A., Pohl, K.: Counterfactual explanations for predictive business process monitoring. CoRR abs/2202.12018 (2022)

16. Hundogan, O., Lu, X., Du, Y., Reijers, H.A.: CREATED: generating viable counterfactual sequences for predictive process analytics. In: Indulska, M., Reinhartz-Berger, I., Cetina, C., Pastor, O. (eds.) Advanced Information Systems Engineering - 35th International Conference, CAiSE 2023, Zaragoza, Spain, 12–16 June 2023, Proceedings. Lecture Notes in Computer Science, vol. 13901, pp. 541–557. Springer, Cham (2023)
17. Kenny, E.M., Ford, C., Quinn, M.S., Keane, M.T.: Explaining black-box classifiers using post-hoc explanations-by-example: the effect of explanations and error-rates in XAI user studies. Artif. Intell. **294**, 103459 (2021)
18. Lee, S., Comuzzi, M., Kwon, N.: Exploring the suitability of rule-based classification to provide interpretability in outcome-based process predictive monitoring. Algorithms **15**(6), 187 (2022)
19. Mehdiyev, N., Fettke, P.: Prescriptive process analytics with deep learning and explainable artificial intelligence. In: Rowe, F., et al. (eds.) 28th European Conference on Information Systems - Liberty, Equality, and Fraternity in a Digitizing World, ECIS 2020, Marrakech, Morocco, 15–17 June 2020 (2020)
20. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Hildebrandt, M., Castillo, C., Celis, L.E., Ruggieri, S., Taylor, L., Zanfir-Fortuna, G. (eds.) FAT* 2020: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, 27–30 January 2020, pp. 607–617. ACM (2020)
21. Povalyaeva, E., Khamitov, I., Fomenko, A.: BPIC 2017: density analysis of the interaction with clients. BPI Challenge **2017** (2017)
22. Raufi, B., Finnegan, C., Longo, L.: A comparative analysis of shap, lime, anchors, and dice for interpreting a dense neural network in credit card fraud detection. In: World Conference on Explainable Artificial Intelligence, pp. 365–383. Springer, Cham (2024)
23. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": explaining the predictions of any classifier. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016, pp. 1135–1144. ACM (2016)
24. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: high-precision model-agnostic explanations. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI 2018), the 30th innovative Applications of Artificial Intelligence (IAAI 2018), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, 2–7 February 2018, pp. 1527–1535. AAAI Press (2018)
25. Rizzi, W., et al.: Explainable predictive process monitoring: a user evaluation. CoRR abs/2202.07760 (2022)
26. Rizzi, W., Di Francescomarino, C., Maggi, F.M.: Explainability in predictive process monitoring: when understanding helps improving. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNBIP, vol. 392, pp. 141–158. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58638-6_9
27. Seabold, S., Perktold, J.: Statsmodels: econometric and statistical modeling with python. In: 9th Python in Science Conference (2010)
28. Stevens, A., De Smedt, J.: Explainability in process outcome prediction: guidelines to obtain interpretable and faithful models. Eur. J. Oper. Res. (2023)
29. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 17:1–17:57 (2019)

30. Velmurugan, M., Ouyang, C., Moreira, C., Sindhgatta, R.: Evaluating fidelity of explainable methods for predictive process analytics. In: Nurcan, S., Korthaus, A. (eds.) Intelligent Information Systems - CAiSE Forum 2021, Melbourne, VIC, Australia, 28 June–2 July 2021, Proceedings. Lecture Notes in Business Information Processing, vol. 424, pp. 64–72. Springer, Cham (2021)
31. Velmurugan, M., Ouyang, C., Moreira, C., Sindhgatta, R.: Evaluating stability of post-hoc explanations for business process predictions. In: Hacid, H., Kao, O., Mecella, M., Moha, N., Paik, H. (eds.) ICSOC 2021. LNCS, vol. 13121, pp. 49–64. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91431-8_4
32. Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM Trans. Intell. Syst. Technol. **10**(4), 34:1–34:34 (2019)
33. Visani, G., Bagli, E., Chesani, F., Poluzzi, A., Capuzzo, D.: Statistical stability indices for lime: obtaining reliable explanations for machine learning models. J. Oper. Res. Soc. **73**(1), 91–101 (2022)
34. van der Waa, J., Nieuwburg, E., Cremers, A., Neerincx, M.A.: Evaluating XAI: a comparison of rule-based and example-based explanations. Artif. Intell. **291**, 103404 (2021)
35. Wickramanayake, B., He, Z., Ouyang, C., Moreira, C., Xu, Y., Sindhgatta, R.: Building interpretable models for business process prediction using shared and specialised attention mechanisms. Knowl. Based Syst. **248**, 108773 (2022)

Process Discovery

eST^2 Miner - Process Discovery Based on Firing Partial Orders

Sabine Folz-Weinstein[1(✉)], Christian Rennert[2], Lisa Luise Mannel[2], Robin Bergenthum[3], and Wil van der Aalst[2]

[1] Chair of Data Science, University of Hagen, Hagen, Germany
sabine.folz-weinstein@fernuni-hagen.de
[2] Chair of Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany
{rennert,mannel,wvdaalst}@pads.rwth-aachen.de
[3] Faculty of Mathematics and Computer Science, University of Hagen, Hagen, Germany
robin.bergenthum@fernuni-hagen.de

Abstract. Process discovery generates process models from event logs. Traditionally, an event log is defined as a multiset of traces, where each trace is a sequence of events. The total order of the events in a sequential trace is typically based on their temporal occurrence. However, real-life processes are partially ordered by nature. Different activities can occur in different parts of the process and, thus, independently of each other. Therefore, the temporal total order of events does not necessarily reflect their causal order, as also causally unrelated events may be ordered in time. Only partial orders allow to express concurrency, duration, overlap, and uncertainty of events. Consequently, there is a growing need for process mining algorithms that can directly handle partially ordered input. In this paper, we combine two well-established and efficient algorithms, the eST Miner from the process mining community and the Firing LPO algorithm from the Petri net community, to introduce the eST^2 Miner. The eST^2 Miner is a process discovery algorithm that can directly handle partially ordered input, gives strong formal guarantees, offers good runtime and excellent space complexity, and can, thus, be used in real-life applications.

Keywords: Business Process Modeling · Process Discovery · Event Data · Partial Orders · Petri Nets

1 Introduction

Process mining gains insights into business processes by analyzing recorded behavior [18]. The goal of process discovery is to generate a process model based on an event log [2–4]. An event log is a multiset of traces, where each trace is a sequence of events, i.e., executed activities of a process. Traditionally, these traces are totally ordered based on the timestamps of the events. To illustrate

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 59–75, 2025.
https://doi.org/10.1007/978-3-031-94571-7_4

this, consider an example from an Educational Process Mining project at the RWTH Aachen which analyzes study behavior [10,29]. Here, every trace in the event log represents the sequence of courses that a student took, ordered by the timestamps when the student passed the course exams. Figure 1 shows three traces of the event log, where student 1 took *General Chemistry*, *Mathematics*, etc. in the depicted order. Note that these traces suggest that, e.g., *Technical Chemistry* is always completed after *Spectroscopy* because this relation exists in all traces.

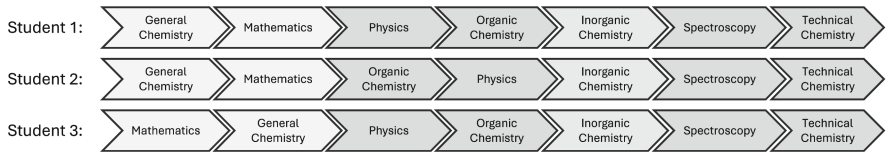


Fig. 1. Example event log with three traces. Each trace is a sequence of courses taken by one student, totally ordered based on course exam dates.

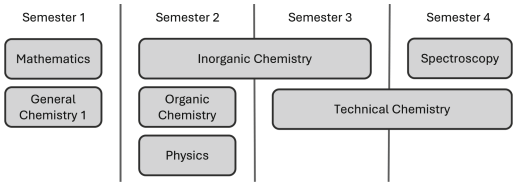


Fig. 2. Partially ordered representation of the three traces in Fig. 1.

In real-life processes, however, activities are often only partially ordered, and some activities can occur independently of each other. Consequently, an observed temporal order of the events does not necessarily reflect their causal relation, as also causally unrelated events may be ordered in time. In our example, we know that students usually do not take courses in a strict sequential order (one after the completion of another), but take several courses in parallel per semester, and that courses have a duration. Thus, if we group all exam timestamps by semester and include course durations, we receive the partially ordered trace in Fig. 2 for all traces of Fig. 1, depicting that the students took several courses concurrently. Note that the order relation between *Spectroscopy* and *Technical Chemistry*, inferred by the totally ordered traces, is accidental and caused by later exam dates within the same semester. The information that *Technical Chemistry* overlaps with *Inorganic Chemistry* and, thus, is not dependent on the completion of *Inorganic Chemistry* could not be reflected by the totally ordered traces either.

This example illustrates that if we discover models based on event data that is totally ordered based on timestamps, on the one hand, we may unwillingly infer dependencies between activities to the discovered model which are purely accidental. On the other hand, we may lose valuable information on the underlying process that is available in the event log, but a total order cannot represent. The more concurrent behavior there is in the underlying process, the higher the risk that the discovered model will not depict the process correctly.

Quite often, the temporal order based on the timestamps of the events is not a total order either. Most real-life event logs have severe data quality problems, leading to timestamps that are unreliable, incomparable, have too coarse or different granularities, especially if data from different source systems and/or manual input must be combined [11,17]. A well-known example is healthcare data [25,26], where many events require manual input to the system, usually done at the end of a shift. Consequently, the timestamp does not necessarily reflect the occurrence of the event itself. Moreover, several events can have the same timestamp, e.g., a date only. Many BPI challenge event logs illustrate the same problem: In the BPIC 2011 log, 87%, and in the BPIC 2012 log, 5% of all events have the same timestamp as their predecessors [5]. In this case, the events are partially ordered due to uncertainty, and any total order would be random.

In turn, there is an increasing amount of additional data available in information systems which can be used to identify causal relations within a process. An example is lifecycle data, which reflects the duration and overlap of events (e.g., contained in the BPIC 2012 log). A lifecycle attribute for events is defined in the event log standard format XES [30], but this information cannot be represented using totally ordered traces.

Thus, to obtain meaningful process mining results and discover models that better reflect the underlying process, we find a growing amount of work in which a trace is a partial order of events [5,7,14,16,20,21,27]. Using partial orders, we can explicitly express both uncertainty and concurrency [19,28], as well as represent duration and time overlaps of events. Furthermore, partial orders often provide a much more compact representation of the recorded behavior, which can improve runtime and space complexity of discovery algorithms.

From a practical point of view, the use of partially ordered event logs has three consequences concerning process discovery applications. First, we need to distinguish between data in event logs that reflects technical information, i.e., generated or required by the information system, and data that reflects characteristics and dependencies of the underlying process. Second, we need an additional preprocessing step for partial order extraction/event log transformation, using data identified in the first step. Third, we need process discovery algorithms that can directly process partially ordered input in real-life settings, which is the focus of this paper. Applying sequential trace-based algorithms on partially ordered event logs is not an efficient option, because we must process all possible interleavings (i.e., sequentializations) of all partially ordered traces, and one partially ordered trace can induce a significant number of interleavings. Furthermore, semantically, concurrency is not the same as interleaving.

In [20], the authors present an overview of partial order-based process discovery. The existing work primarily refers to synthesis or folding, e.g., Prime Miner [7], ILP^2 Miner [16], unfolding-based process discovery [21], multi-phase process mining approaches [13], and folding-based approaches [9]. These approaches give strong formal guarantees, i.e., they produce models with high fitness and precision. However, they come with considerable space and runtime complexity, which is problematic when working with real-life event logs.

To close this gap, in this paper, we combine two well-established and efficient algorithms, the eST Miner and the Firing LPO algorithm from Petri net theory, to introduce the eST^2 Miner. The eST Miner [22] is a replay-based process discovery algorithm. To find the places of the result Petri net, it enumerates and evaluates all possible places of the net in linear time by firing every trace in the event log. Thanks to a special strategy for ordering, traversing, and pruning the set of possible candidate places, the algorithm is time-efficient and only needs to store the input event log and the resulting net. Working with partially ordered event logs, however, it is not possible any more to simply replay traces from start to end. Therefore, in the eST^2 Miner, we adapt the currently most efficient verification algorithm from Petri net theory [8] to verify whether a partial order is replayable. The new eST^2 Miner handles totally ordered event logs just like the eST Miner and provides the same guarantees for the discovered process models, but it can also directly handle partially ordered input.

In this paper, we address the problem of discovering a process model based on an event log which is a multiset of labeled partial orders. We introduce the eST^2 Miner which is based on two well-established algorithms. We implement the eST^2 Miner and evaluate the new approach based on public and private logs.

2 Preliminaries

A multiset m over X is a function $m\colon X \to \mathbb{N}$. We write $m = \sum_{x \in X} m(x) \cdot x$ to denote all multiplicities of m. We extend the notion of a subset to the concept of multisets, where a multiset is considered a subset of another multiset if the cardinality of every element is equal to or less than that in the other multiset.

We model observed partially ordered behavior as a partially ordered set of activities (see [4] for a detailed introduction).

Definition 1 (Labeled Partial Order, lpo). *Let A be a set of activities. A labeled partial order (lpo) is a triple $(V, \prec, l)$ where V is a finite set of nodes, $\prec \subseteq V \times V$ is a transitive and irreflexive relation, and $l\colon V \to A$ a labeling function. Let $n \in V$ be a node. We denote $\{n' \in V \mid n' \prec n\}$ the set of predecessors and $\{n' \in V \mid n \prec n'\}$ the set of successors of n.*

We define an event log as a multiset of labeled partial orders.

Definition 2 (Event Log). *An event log is a multiset of labeled partial orders.*

To model business processes, we use the concept of workflow nets [1], a subclass of marked Petri nets [12].

Definition 3 (Workflow Net). *A workflow net N is a tuple (P, T, F) where P is a finite set of places, T is a finite set of transitions such that $P \cap T = \emptyset$ holds, $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, $i, o \in P$ are two places for which $\bullet i = o\bullet = \emptyset$ holds, and where every node $n \in (P \cup T)$ is on a directed path from i to o. Let $n \in (T \cup P)$ be a node of a workflow net. We denote $\bullet n = \{n' \in (T \cup P) \mid (n', n) \in F\}$ the preset of n and $n\bullet = \{n' \in (T \cup P) \mid (n, n') \in F\}$ the postset of n.*

For workflow nets, there is a simple firing rule. A marking of N is a multiset $m\colon P \to \mathbb{N}$. A transition t can fire at marking m if $\bullet t \subseteq m$ holds. Once transition t fires, the marking of N changes from m to m', where for m' it holds that $m'(p) = m(p) - 1$ if $p \in (\bullet t \setminus t\bullet)$ or $m'(p) = m(p) - 1$ if $p \in (t\bullet \setminus \bullet t)$ or $m'(p) = m(p)$ otherwise.

The behavior of a workflow net is the set of all possible partially ordered sets of firing events that bring the workflow net from its initial marking i to its final marking o. Formally we can define this language as the set of labeled partial orders for which a so-called valid tokenflow exists for every place [8]. A *compact* tokenflow is a distribution of tokens on the skeleton arcs of the lpo. This distribution is valid if it satisfies certain conditions. In the following, we adopt the original definition of compact tokenflows to workflow nets.

Definition 4 (Behavior of a Workflow Net). *Let $N = (P, T, F)$ be a workflow net, let $lpo = (V, \prec, l)$ be a labeled partial order, and $l(V) \subseteq T$. Let $<$ be the smallest relation for which the transitive closure is $\prec$. A compact tokenflow is a function $x\colon\ <\ \to \mathbb{N}$. Fix a $p \in P \setminus \{i, o\}$. Place p is valid for lpo if and only if there is a compact tokenflow x such that the following conditions hold:*

(i) $v \in V, (p, l(v)) \in F \implies \sum_{v' < v} x(v', v) \geq 1$, *and*

(ii) $\forall_{v \in V}\colon \sum_{v \prec v'} x(v, v') = \begin{cases} \sum_{v' \prec v} x(v', v) - 1, (p, l(v)) \in F \wedge (l(v), p) \notin F, \\ \sum_{v' \prec v} x(v', v) + 1, (l(v), p) \in F \wedge (p, l(v)) \notin F, \\ \sum_{v' \prec v} x(v', v) \quad , \text{ otherwise.} \end{cases}$

Place i is valid for lpo if and only if

(iii) $\left|\{v \in V \mid (i, l(v)) \in F\}\right| = 1.$

Place o is valid for lpo if and only if

(iv) $\left|\{v \in V \mid (l(v), o) \in F\}\right| = 1.$

If and only if all places of N are valid for lpo, lpo is in the language of N.

Condition (*i*) ensures that every transition receives enough tokens to fire, (*ii*) that the firing rule applies, (*iii*) and (*iv*) that the initial token is consumed and the final token is produced.

An event log is replayable in a workflow net if every lpo of the event log is in the language of the workflow net.

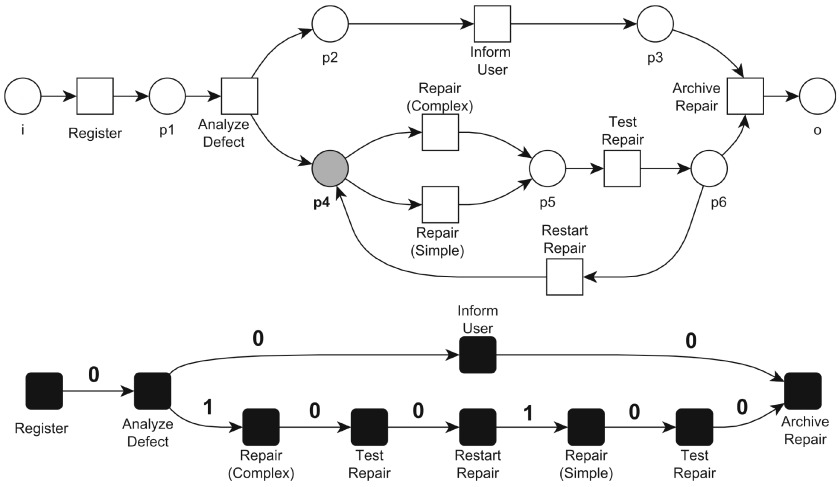


Fig. 3. A workflow net (top) and a labeled partial order with a compact tokenflow for p4 (bottom).

Example 1. Figure 3 shows a workflow net and an lpo. The numbers denoted on the arcs of the lpo depict a compact tokenflow for place *p4*. In this tokenflow, both *Repair (Complex)* and *Repair (Simple)*, which are in the postset of *p4*, receive enough tokens to fire (Definition 4(*i*)). The firing rule (Definition 4(*ii*)) applies because *Analyze Defect* and *Restart Repair*, which are in the preset of *p4*, increase the amount of tokens that they received from their predecessors in the lpo by one token. *Repair (Complex)* and *Repair (Simple)*, which are in the postset of *p4*, decrease the amount by one token. All other events leave the amount unchanged. Definition 4(*iii*) and Definition 4(*iv*) are satisfied because in N, there are arcs from i to *Register* and from *Archive Repair* to o, such that the initial token can be consumed and the final token be produced. Thus, all conditions are fulfilled and *p4* is valid for this lpo.

3 Replay-Based Process Discovery

In this section, we recapitulate the basic ideas of replay-based process discovery and the original eST Miner as presented in [22], which serves as the foundation for our new eST^2 Miner approach.

To start with, we initialize a workflow net by simply creating one transition for each activity in the event log. This initial placeless net can replay any given event log because no place restricts the firing of the transitions. Then, we successively add places and related arcs to prune the behavior of the workflow net such that it matches the behavior of the given event log. This idea is similar to discovery algorithms based on the theory of regions [4].

To find the places of the workflow net, we enumerate and traverse all possible places of the net. In this context, we assume that each place also defines its preset

and postset, i.e., how it connects to a set of transitions. The set of all possible places of the net is called the set of candidate places. For each candidate place, we evaluate if it is valid for the given event log by replaying the event log on the place. Only if the place is valid, it can be added to the final result. If a candidate is not valid, we move on to check the next candidate place.

The main idea of the eST Miner is that, during this evaluation of a candidate place, we do not only evaluate if a place is valid or not. For places which are not valid, we distinguish if this is due to a lack of tokens or due to remaining tokens on the place. Thus, while replaying each trace in the event log on the candidate, we keep track of the number of tokens in every intermediate marking. If the number of tokens ever becomes negative, the behavior is not replayable, and we call the candidate place *underfed*. If, after replaying the behavior of the event log, the place is not empty, we call the candidate place *overfed*. Note that a place can be underfed and overfed at the same time. The additional underfed and overfed information is used for an efficient traversal of the set of candidates.

Although the set of potential candidate places is finite, it is still exponential in the number of transitions, a number impossible to efficiently store, retrieve, and evaluate. Therefore, the eST Miner deterministically calculates the next candidate place to be evaluated and prunes the candidate space based on the evaluation results of the current candidate place. To achieve this, the set of candidate places is represented in the form of a tree, with different branches reflecting the addition of incoming and outgoing arcs to transitions. If a candidate place is underfed, adding outgoing arcs will not help make it a valid candidate place. Similarly, if a candidate is overfed, adding incoming arcs will not help make it valid. Using the underfed and overfed information of the current candidate place, the algorithm can prune the tree and skip the traversal of entire unfitting subtrees. This drastically decreases the overall runtime on real-life event logs (by 40–95%), still guaranteeing that all valid places are visited.

By applying such a replay-based process discovery strategy, we only add valid places to the net. As a consequence, the language of the discovered workflow net will always include the behavior of the event log, and the model will be perfectly fit. However, for practical applications, this is too restrictive, as event logs are known to also contain noise. To address this, the eST Miner uses an additional, second evaluation step on log level and a configurable noise handling parameter τ. In the log level evaluation step, the results of all traces of the event log are summarized. Only if a place is valid for a fraction of τ or more of the traces in the event log, the place is added to the result net. The log level candidate evaluation using τ is also applied on the underfed and overfed values used for the pruning and traversal of the candidate tree.

A high-level overview of the algorithmic framework is provided in Fig. 4. For implementation details of the original eST Miner, including post-processing of the result net, we refer the interested reader to [22,24].

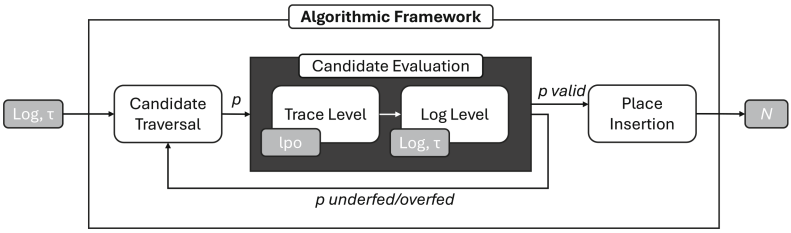


Fig. 4. High-level overview of the eST^2 Miner framework.

4 The eST^2 Miner

To apply the concept of replay-based process discovery to partially ordered event logs, we introduce the eST^2 Miner. The eST^2 Miner follows all the concepts outlined in the previous section. Thus, the eST^2 Miner is a variant of the eST Miner, but it addresses the candidate evaluation specifically for labeled partial orders. It decides whether a candidate place is valid, underfed, or overfed for a given lpo by calculating compact tokenflows (Definition 4). Since we evaluate each lpo of the entire partially ordered event log on every candidate place, the runtime and efficiency of this evaluation part are vital.

To verify whether a valid compact tokenflow exists for a place and a given lpo, a maximal flow problem must be solved in $\mathcal{O}(n^3)$ time [8], where n is the number of nodes of the lpo. The maximal flow algorithm explores all possible distributions of tokens on the arcs of the lpo, and directly indicates if a candidate place is valid, underfed, or overfed. However, solving a maximal flow problem in cubic time is not an efficient option for large real-life event logs.

Thus, for the eST^2 Miner, we adapt the currently most efficient approach presented in [8], which we will refer to as the *Firing LPO* algorithm in the remainder of this paper. The main idea of the Firing LPO algorithm is that for certain places, it is easy to determine whether a valid compact tokenflow exists for a given lpo or not. To identify such places, the algorithm applies two heuristics steps first, the so-called *forward* and *backward strategies*. Both of these heuristics steps run in linear time. Consequently, the exact maximal flow algorithm only needs to be applied on places which cannot be resolved by the heuristics steps.

Experimental results of the Firing LPO algorithm show that it runs in quadratic or even linear time for most practical use cases [8], and that it is especially fast on restricted Petri nets like workflow nets. In workflow nets, there are no arc weights, and a lot of places are empty most of the time. Therefore, the number of possible distributions of tokens decreases significantly, and the probability of finding a valid compact tokenflow in the two heuristics steps increases considerably. Therefore, in most cases, the replayability of an lpo on a place can be decided after forward or backward strategy, i.e., in linear time (like the token replay strategy for totally ordered traces in the original eST Miner).

Note that it is also possible to evaluate whether totally ordered traces are replayable using the Firing LPO algorithm. In totally ordered traces, only one

single possible distribution of tokens exists. Therefore we can always decide whether a place is valid or not after executing the first heuristics step, which in fact is equivalent to the original eST Miner replay strategy [22]. For detailed information on the Firing LPO algorithm, we refer the interested reader to [8].

Necessary Adaptations. Although certain aspects of the original Firing LPO algorithm must be adapted to the scope and setting of the eST^2 Miner, the two algorithms are quite a perfect match.

The *input* to our adapted Firing LPO algorithm within the eST^2 Miner is an lpo and a workflow net which only consists of the candidate place p and the transitions in its preset and postset. As the original Firing LPO algorithm was designed for general marked Petri nets, and workflow nets are a restricted form of general marked Petri nets, these one-place nets can be handled by the algorithm without any modifications.

However, it is no longer sufficient to evaluate whether a place is valid or not for a specific lpo. Therefore, we must adapt the *output* of our adapted algorithm to provide the more specific information whether the place is underfed and/or overfed. This means, if a place is not valid, we need to distinguish if this is due to missing or remaining tokens on the place.

The original algorithm verifies if a place is valid or not for a specific lpo by evaluating whether or not a compact tokenflow (i.e., a distribution of tokens along the arcs of the lpo) exists such that every node of the lpo receives enough tokens for its related transition to fire. If no such distribution exists, one or several nodes of the lpo do not receive enough tokens, which can be directly translated to the place being underfed.

The original algorithm does not evaluate yet if tokens remain on a place, i.e., whether or not a place is overfed, because this is irrelevant for general marked Petri nets. However, as a coincidence, the original Firing LPO algorithm already calculates the final marking, i.e., the amount of tokens which are not consumed by the last node of the lpo, and which remain on the place. This final marking is proven to be unique even if no valid compact tokenflow can be constructed [8]. We can use this final marking to identify if a place is overfed.

Place Evaluation Using the Adapted Firing LPO Algorithm. The eST^2 Miner extends every lpo of the partially ordered event log by a unique start node ▶, which is earlier than all other nodes of the lpo, and a unique final node ■, which is later than all other nodes of the lpo. In our result workflow net, we connect the unique place i to ▶ and the unique place o to ■ such that $i\bullet = \{\blacktriangleright\}$ and $\bullet o = \{\blacksquare\}$ hold. Thus, we ensure that the initial token on i is consumed and the final token on o is produced (Definition 4(*iii*) and (*iv*)) by construction, and that we only evaluate inner places of a workflow net.

The evaluation starts with the *forward strategy* heuristics. In this heuristics step, we process all nodes of the lpo in one sequential order which respects the $\prec$-order. We brute-force construct one possible distribution of tokens along the arcs of the lpo, i.e., one possible compact tokenflow. The algorithm verifies if every node receives enough tokens to fire (Definition 4(*i*)), and ensures that the firing rule applies (Definition 4(*ii*)). To guarantee a linear runtime, the algorithm

can only explore one possible distribution of tokens in the heuristics step, i.e., one compact tokenflow, but it identifies and stores the information if alternative distributions are possible. At the end of the forward heuristics step, it calculates the unique final marking.

After the forward strategy heuristics step, we can always decide if a place is **overfed** or not. A place is overfed if the final marking is positive; otherwise, it is not overfed. For some places, we can also decide if they are **underfed** or not:

1. If the final marking is negative, we can deduce that in all possible distributions of tokens, there will be a lack of tokens. Thus, it is impossible to fulfill Definition 4(i), and we can classify the place as underfed.
2. If we found a distribution such that for each node of the lpo, there are enough tokens for its related transition to fire, then Definition 4(i) is fulfilled for all nodes of the lpo, and the place is not underfed.
3. If Definition 4(i) is violated for one or several nodes of the lpo, we must still consider whether the algorithm detected that alternative distributions of tokens are possible. If no alternative distributions are possible, the place is underfed.

We cannot decide whether the place is underfed in case Definition 4(i) is violated, but alternative distributions are possible. In this case, one of these alternative distributions may be a valid distribution. Therefore, we apply the second heuristics step of the algorithm, the *backward strategy*. The backward strategy works just like the forward strategy but in the reverse direction, i.e., it processes all nodes of the lpo in the reverse total order used in the forward strategy, except that it does not calculate and evaluate a final marking anymore.

If no decision can be made by the backwards heuristics step either, the maximal flow algorithm with worst-case cubic runtime must be applied to decide whether or not the place is underfed.

5 Evaluation

Resuming our Educational Process Mining example, Fig. 5 depicts two workflow nets discovered by the eST^2 Miner on a partially ordered (left) and by the eST Miner on a totally ordered event log (right) based on study behavior data. The depicted nets are not intended to be readable in detail, but to highlight their structural differences. As expected, significantly more dependencies exist in the net that was discovered based on totally ordered input (right). As an example, consider the *Bachelor Thesis* marked in red. In the net based on totally ordered input (right), we find order relations between the *Bachelor Thesis* and eight other courses, marked in blue, while only two courses, marked in teal, are concurrent to *Bachelor Thesis*. In the net discovered on partially ordered input (left), all of these courses (marked teal and blue) are concurrent to *Bachelor Thesis*. Project owners confirm that students frequently complete courses while already working on their bachelor thesis. Presumably, the order relations are found because the bachelor colloquiums are usually held at the very end of a semester, after all

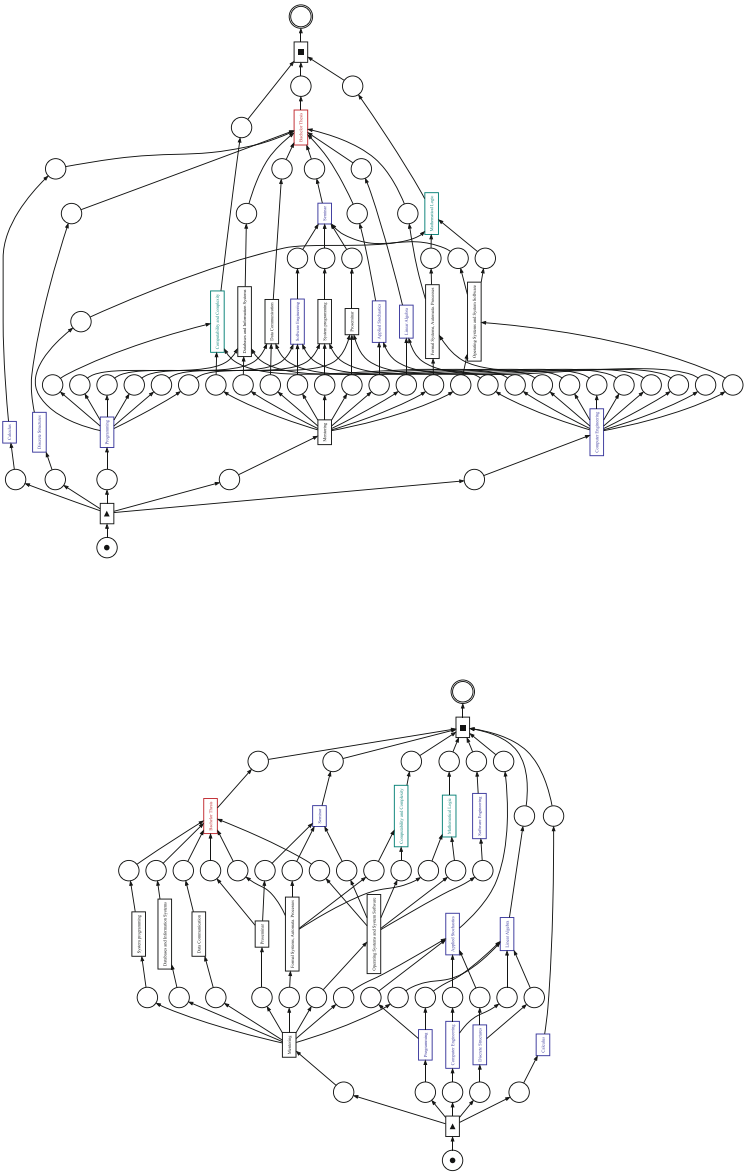


Fig. 5. Workflow nets discovered by the eST^2 Miner based on a partially ordered event log (left) and by the eST Miner based on a totally ordered event log (right) containing the study behavior data of 355 students of the RWTH Aachen which completed their Bachelor's degree of Computer Science and started their studies in winter semesters 2015–2018.

other course exams. Since the event log data of this project is not public and the project is still ongoing, we are unable to further evaluate the quality of these two process models in more detail. As the project progresses, a more detailed validation will certainly provide more valuable insights.

Therefore, to evaluate our approach in a transparent, reproducible way and on a broader basis, we use several well-known public totally ordered event logs which we transform to partially ordered event logs using a so-called concurrency oracle. Concurrency oracles [6,7,20] use heuristics to derive a partial order relation on top of the total order relation, based on additional attributes of the event log. As the focus of our evaluation is not on the performance of oracles, we apply the same oracle on all our test event logs. We use the so-called Alpha oracle [7] implemented in the CCO tool[1] [15], which evaluates directly-follows relations to identify concurrency. Note that deriving partially ordered logs from a totally ordered event log using a concurrency oracle may infer some additional behavior that was not recorded in the event log, such that process models discovered based on the two log versions are not entirely comparable. However, due to the characteristics of the Alpha oracle, the partially ordered version of the log essentially contains the same behavior as the totally ordered event log but in another representation. Each partial order represents several traces of the totally ordered log (i.e., sequentializations of the partial order). Thus, we expect to discover the same process models for the two event log versions.

To this date, no comparable process discovery approaches for partially ordered input exist that run on real-life event logs, nor do conformance checking metrics for partially ordered input. Thus, to evaluate our approach, we compare the eST^2 Miner (based on the partially ordered version of our test event logs) to the eST Miner (based on the totally ordered version). This comparison still allows us to assess the overall quality of the discovered process models discovered by the eST^2 Miner, as well as to compare the runtimes. We evaluate the quality of the workflow nets discovered by the eST^2 Miner using standard quality metrics based on the behavior described by the totally ordered event log version, since no other metrics exist yet. Note that we compare the runtimes of eST^2 Miner and eST Miner mainly to assess the general efficiency of the eST^2 Miner approach. As the main goal of partial-order based process discovery is not a speedup, but the ability to directly process partially ordered input for the reasons described in the introduction, we are interested in a comparison of the algorithm runtimes in relation to the sizes of the event log versions. Often, partial orders are a more compact representation of the behavior, since several total orders may be interleavings of the same partial order.

Implementation. The eST^2 Miner is implemented in ProM[2], built on the basis of the most recent implementation of the eST Miner [23], accessible on GitHub[3]. Note that specific parameters required by this implementation remain fixed during all our experiments ($\delta = 1$, $s = 5$, candidate space tree depth of 5). To

[1] https://github.com/sabinefw/ConfigurableConcurrencyOracleTool.
[2] https://promtools.org/.
[3] https://github.com/promworkbench/eST2-miner.

exclude a possible runtime impact with respect to different existing eST Miner implementations, we use the eST^2 Miner implementation on all event log versions (totally and partially ordered), since for totally ordered traces, the trace evaluation step in the eST^2 Miner is equivalent to the replay in the eST Miner.

Experimental Setup. The evaluation is performed single-threaded on a 16-core Intel i7-1260P 2.10 GHz machine with 32 GB of main memory.

Table 1. Fitness and precision values of the workflow nets discovered by the eST^2 Miner with τ thresholds of 1.0, 0.8, and 0.5.

event log	$\tau = 1.0$		$\tau = 0.8$		$\tau = 0.5$	
	fitness	precision	fitness	precision	fitness	precision
Repair	1.00	0.64	1.00	0.64	0.88	0.84
Teleclaims	1.00	0.31	0.96	0.53	0.82	0.40
Reviewing	1.00	0.48	1.00	0.48	0.97	0.49
RTFM	1.00	0.15	0.94	0.68	0.79	0.51
BPI12(a)	1.00	0.20	0.95	0.35	0.80	0.79
BPI12(o)	1.00	0.20	0.97	0.24	0.85	0.87
BPI19(c)	1.00	0.14	0.94	0.44	0.88	1.00
Sepsis(40)	1.00	0.09	0.99	0.14	0.97	0.16

Table 2. Runtime in seconds for the eST^2 Miner on the partially ordered event log version and for the eST Miner on the totally ordered event log version with $\tau = 1.0$.

event log	#cases	#variants		relative	runtime in s		speedup
		lpos	traces	difference	eST^2	eST	
Repair	1,000	9	39	77%	1.03	4.59	78%
Teleclaims	3,512	8	12	33%	2.91	4.71	38%
Reviewing	100	93	96	3%	204.38	202.35	−1%
RTFM	150,370	85	231	63%	25.00	83.05	70%
BPI12(a)	13,087	12	17	29%	1.96	3.44	43%
BPI12(o)	5,015	75	168	55%	7.48	15.54	52%
BPI19(c)	14,498	206	281	27%	135.37	172.61	22%
Sepsis(40)	1,050	435	846	49%	1,074.61	1,768.46	39%

We use the artificial event logs Repair example[4] and Teleclaims [3] as well as the real-life event logs Reviewing [3], Road Traffic Fine Management (RTFM),

[4] Example event log file for: https://promtools.org/.

Sepsis, and the BPI Challenge logs 2012 (A), 2012 (O) and 2019[5] for our experiments. The BPI Challenge log 2019 is filtered for the "Consignment" trace attribute, and the Sepsis log for traces up to a maximal length of 40.

Fitness and Precision. Table 1 depicts the fitness and precision values for the workflow nets discovered by the eST^2 Miner with noise handling thresholds τ of 1.0, 0.8, and 0.5. This means that (at least) 100%, 80%, or 50% of the cases in the event log must be replayable in the result net. We used fitness metrics based on alignments, and precision metrics based on escaping edges [4]. The fitness of the result nets is related to the selected τ thresholds, and the precision tends to decrease with increasing fitness. Both values can be balanced by selecting a suitable τ. Note that for almost all test event logs, using the same parameter settings, the same workflow nets are discovered for both event log versions. In case the discovered nets differ, the difference is minimal.

Runtime. Table 2 compares the runtime in seconds for eST^2 Miner and eST Miner for a fixed τ threshold of 1.0. As a reference, we include the number of cases, lpo and sequential trace variants for each event log, to assess the size of the partially and the totally ordered input. For most event logs, the speedup of the eST^2 Miner compared to the runtime of the eST Miner scales almost linearly to the relative difference between the log versions. For example, in the Teleclaims log, the relative difference between the traces in the log variants is 33%, almost linearly reflected in the speedup of 38%. Even in the Reviewing log, where there are almost as many lpos as totally ordered traces, the eST^2 Miner does not show a significant runtime increase compared to the eST Miner. This supports our expectation that the candidate evaluation step of the eST^2 Miner is executed in linear time for most candidate places, like in the eST Miner (although the problem is now cubic), and that the eST^2 Miner approach is comparably efficient.

6 Conclusion

There are various important reasons for using partially ordered event logs. Real-life processes are partially ordered by nature, and there are ways to obtain data on the causal relation of the events. A total order based on the temporal order is prone to fail as soon as the timestamps are unreliable, incomparable, or too coarse granular. Only partial orders allow us to directly model uncertainty, concurrency, duration, and overlap of events, which a total order cannot capture. Overcoming the total order assumption is especially relevant in fields such as healthcare, education, and logistics which exhibit highly concurrent behavior. Consequently, there is a growing need for process mining algorithms which can directly handle partially ordered input.

To bridge this gap and fully exploit the concurrency information of real-life data in process discovery, we combine two well-established and efficient algorithms, the eST Miner process discovery algorithm and the Firing LPO algorithm

[5] Online accessible at: https://data.4tu.nl/.

from Petri net theory, to introduce the eST^2 Miner. The eST^2 Miner is a process discovery algorithm which can handle both partially and totally ordered input while maintaining the same guarantees with respect to the discovered workflow nets as the established eST Miner, offering space efficiency and good runtimes even on real-life event logs. We conducted several experiments with well-known public event logs, assessing the runtime of the algorithm and the quality of the discovered process models. The results show that, for the majority of our experiments, the runtime of the eST^2 Miner on partially ordered input scales almost linearly with the level of compactification of the event log compared to the eST Miner, and that the same nets are discovered based on both event log versions.

In future work, we plan to evaluate the eST^2 Miner with respect to other new partial order-based discovery approaches, all based on the same partially ordered input, to further analyze, compare, and optimize our approach. We also intend to conduct further experiments to evaluate and analyze the quality of the discovered process models based on partially ordered event logs created from the same totally ordered event log using different concurrency oracle types.

Acknowledgments. We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research (grant no. 1191945). The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint projects AIStudyBuddy (grant no. 16DHBKI016) and Bridging AI (grant no. 16DHBKI023). The authors have no competing interests to declare that are relevant to the content of this article.

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: ICATPN. LNCS, vol. 1248, pp. 407–426. Springer (1997)
2. van der Aalst, W.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
3. van der Aalst, W.M.P.: Process Mining - Data Science in Action, 2nd edn. Springer (2016)
4. van der Aalst, W.M.P.: Process mining: a 360 degree overview. In: Process Mining Handbook, LNBIP, vol. 448, pp. 3–34. Springer (2022)
5. van der Aalst, W.M.P., Santos, L.F.R.: May I take your order? - on the interplay between time and order in process mining. In: Business Process Management Workshops. LNBIP, vol. 436, pp. 99–110. Springer (2021)
6. Armas-Cervantes, A., Dumas, M., Rosa, M.L., Maaradji, A.: Local concurrency detection in business process event logs. ACM Trans. Internet Technol. **19**(1), 16:1–16:23 (2019)
7. Bergenthum, R.: Prime miner - process discovery using prime event structures. In: ICPM, pp. 41–48. IEEE (2019)
8. Bergenthum, R.: Firing partial orders in a petri net. In: Petri Nets. LNCS, vol. 12734, pp. 399–419. Springer (2021)
9. Bergenthum, R., Mauser, S.: Folding partially ordered runs. In: Proceedings of Workshop Applications of Region Theory, vol. 2011, p. 12 (2011)

10. Bogarín, A., Cerezo, R., Romero, C.: A survey on educational process mining. WIREs Data Mining Knowl. Discov. **8**(1) (2018)
11. Bose, J.C.J.C., Mans, R.S., van der Aalst, W.M.P.: Wanna improve process mining results? In: CIDM, pp. 127–134. IEEE (2013)
12. Desel, J., Reisig, W.: Place or transition petri nets. In: Petri Nets. LNCS, vol. 1491, pp. 122–173. Springer (1996)
13. van Dongen, B.F., Van der Aalst, W.M.: Multi-phase process mining: aggregating instance graphs into EPCS and petri nets. In: PNCWB 2005 workshop, pp. 35–58 (2005)
14. Dumas, M., García-Bañuelos, L.: Process mining reloaded: Event structures as a unified representation of process models and event logs. In: Petri Nets. LNCS, vol. 9115, pp. 33–48. Springer (2015)
15. Folz-Weinstein, S., Bergenthum, R., Beecks, C.: Partially ordered event logs and concurrency oracles. In: AWPN 2024 workshop proceedings. Gesellschaft für Informatik eV (2024)
16. Folz-Weinstein, S., Bergenthum, R., Desel, J., Kovár, J.: Ilp^2 miner - process discovery for partially ordered event logs using integer linear programming. In: Petri Nets. LNCS, vol. 13929, pp. 59–76. Springer (2023)
17. ter Hofstede, A.H.M., et al.: Process-data quality: the true frontier of process mining. ACM J. Data Inf. Qual. **15**(3), 29:1–29:21 (2023)
18. IEEE Task Force on Process Mining: Process mining manifesto. In: Business Process Management Workshops (1). LNBIP, vol. 99, pp. 169–194. Springer (2011)
19. Janicki, R., Koutny, M.: Structure of concurrency. Theor. Comput. Sci. **112**(1), 5–52 (1993)
20. Leemans, S., van Zelst, S.J., Lu, X.: Partial-order-based process mining: a survey and outlook. Knowl. Inf. Syst. **65**(1), 1–29 (2023)
21. de León, H.P., Rodríguez, C., Carmona, J., Heljanko, K., Haar, S.: Unfolding-based process discovery. In: ATVA. LNCS, vol. 9364, pp. 31–47. Springer (2015)
22. Mannel, L.L., van der Aalst, W.M.P.: Finding complex process-structures by exploiting the token-game. In: Petri Nets. LNCS, vol. 11522, pp. 258–278. Springer (2019)
23. Mannel, L.L., van der Aalst, W.: Discovering process models with long-term dependencies while providing guarantees and filtering infrequent behavior patterns. Fundam. Informaticae **190**(2–4), 109–158 (2024)
24. Mannel, L.L., Bergenthum, R., van der Aalst, W.M.P.: Removing implicit places using regions for process discovery. In: ATAED@Petri Nets. CEUR Workshop Proceedings, vol. 2625, pp. 20–32. CEUR-WS.org (2020)
25. Mans, R., van der Aalst, W.M.P., Vanwersch, R.J.B.: Process Mining in Healthcare - Evaluating and Exploiting Operational Healthcare Processes. Springer Briefs in Business Process Management, Springer (2015)
26. Martin, N.: Data quality in process mining. In: Interactive Process Mining in Healthcare, pp. 53–79. Springer (2020)
27. Pegoraro, M., Uysal, M.S., van der Aalst, W.M.P.: Discovering process models from uncertain event data. In: Business Process Management Workshops. LNBIP, vol. 362, pp. 238–249. Springer (2019)
28. Pratt, V.: Modelling concurrency with partial orders> Int. J. Parallel Program. **15**(1) (1986)
29. Rennert, C., Pourbafrani, M., van der Aalst, W.M.P.: Evaluation of study plans using partial orders. In: ICPM Workshops. LNBIP, vol. 533, pp. 154–166. Springer (2024)

30. Wynn, M.T., van der Aalst, W., Verbeek, E., Stefano, B.: The IEEE XES standard for process mining: Experiences, adoption, and revision [society briefs]. IEEE Comput. Intell. Mag. **19**(1), 20–23 (2024)

Federated Stochastic Process Discovery Using Grammatical Inference

Hootan Zhian(✉), Rajkumar Buyya, and Artem Polyvyanyy

The University of Melbourne, Parkville, VIC 3010, Australia
hzhian@student.unimelb.edu.au, {rbuyya,artem.polyvyanyy}@unimelb.edu.au

Abstract. Process discovery studies algorithms for constructing process models that describe control flow of systems that generated given event logs, where an event log is a recording of executed traces of a system, with each trace captured as a sequence of executed actions. Traditional process discovery relies on an event log recorded and stored in a centralized repository. However, in distributed environments, such as cross-organizational process discovery, this centralization raises concerns about data availability, privacy, and high communication and bandwidth demands. To address these challenges, this paper introduces a novel Federated Stochastic Process Discovery (FSPD) approach. FSPD avoids centralized event logs by retaining them in decentralized silos, at organizations where they were originally recorded. Process discovery is then performed locally within each organization on its event log, and the resulting local models are shared with a central server for aggregation into a global model. Our evaluations on industrial event logs demonstrate that FSPD effectively constructs global process models while preserving organizational autonomy and privacy, providing a scalable and robust solution for process discovery in distributed settings.

Keywords: Stochastic process mining · federated process discovery · cross-silos process discovery · grammatical inference

1 Introduction

Process mining studies ways to use data generated by information systems when executing business processes to understand historical and to improve future processes [2]. Process discovery is one of the core problems in process mining. Given an event log, a collection of recorded traces, each captured as a sequence of executed actions, a process discovery algorithm aims to construct a process model that faithfully describes the traces the system that generated the event log can support [5]. A good discovered process model should describe as many traces from the input event log as possible (good recall), not describe many traces that are not in the event log (good precision), describe traces the system is likely to execute in the future (good generalization), and be as simple as possible (good simplicity) [7].

Most commercial process discovery tools construct Directly-Follows Graphs (DFGs) from event logs [3]. A DFG is a directed graph with nodes representing actions, edges capturing the "can follow" ordering constraints between the

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 76–93, 2025.
https://doi.org/10.1007/978-3-031-94571-7_5

actions, and numeric weights attached to nodes and edges specifying indicative frequencies of observing the corresponding actions and their adjacent executions in the event log. The level of detail in the constructed DFG is typically controlled by a threshold, which specifies how much information from the event log can be disregarded in the model [3]. Business analysts study the system by exploring models constructed at various thresholds and, thus, different levels of detail [15]. To support interactive adjustments of the threshold, most DFG discovery algorithms aim to meet tight runtime requirements, often operating in linear time relative to the size of the input event log.

Alkhammash et al. [6] present a genetic algorithm for stochastic process discovery (*GASPD*) that can construct DFGs that are Pareto-superior in interesting qualities over DFGs constructed by the state-of-the-art techniques proposed by academia, which construct models of similar qualities as process discovery tools from major process mining vendors [14]. *GASPD* is based on *ALERGIA*, a grammatical inference technique with linear empirical and cubic worst-case runtime complexity relative to the size of the event log [9]. However, *GASPD* searches the parameter space of the grammatical inference using a genetic strategy, which substantially increases the overall runtime requirements.

In this paper, we present an extension of *GASPD* capable of federated process discovery. *GASPD*, like other traditional process discovery algorithms, operates over a centralized event log. Federated *GASPD* works over a distributed event log in two phases. First, *GASPD* discovers models of various quality characteristics on each part of a distributed event log, where the distributed event log can be seen as a collection of local event logs, with each local event log stored on a dedicated device. Then, superior DFGs discovered from each local event log are sent to the server, where they are aggregated into a (collection of) sound [6], that is, correct, DFG(s) that describe the overall system. This distributed divide-and-conquer process discovery approach supports cross-organizational process discovery, which aims to analyze processes that span multiple organizations that may be reluctant to share raw data and instead exchange event log abstractions [4,12]. Furthermore, federated process discovery in the cross-silos setting improves data privacy and security [12,16], reduces data transfer costs [12,17], and supports scalability and efficiency [1,8,17].

Concretely, this paper contributes:

1. An efficient algorithm for merging a deterministic frequency finite automaton (DFFA) into a target DFFA, where a DFFA is an abstract model of an event log constructed by *ALERGIA* that can be translated to a DFG by a technique used by *GASPD*.
2. A discussion of formal properties of merged DFFAs (and the practical consequences of these properties), including consistency, similarity of the DFFAs involved in merging, and morphism between the merged DFFA and the input target DFFA.
3. The *FedGASPD* and *FedCGASPD* algorithms (federated versions of *GASPD*) that adapt *GASPD* to discover DFFAs using *ALERGIA* from local event logs and then merge superior discovered DFFAs into, respectively, a sin-

gle DFFA and a collection of DFFAs that can be translated to sound DFGs, thus implementing FSPD.
4. An evaluation over industrial event logs confirming that *FedGASPD* and *FedCGASPD* can discover process models from distributed event logs that have similar quality characteristics as process models constructed by *GASPD* from the corresponding centralized event logs and work substantially faster than *GASPD*, with *FedCGASPD* constructing interesting models of smaller sizes.

The remainder of the paper proceeds as follows. The next section discusses related work. Then, Sect. 3 presents preliminaries necessary to understand the subsequent discussions. Section 4 introduces *FedGASPD* and *FedCGASPD*, while Sect. 5 presents results of an evaluation of these algorithms based on their open-source implementation.[1] Finally, Sect. 6 closes the paper with a discussion of the limitations of our approach and directions for future work, and states concluding remarks.

2 Related Work

This section discusses existing works in federated process mining, divided-and-conquer strategies for process discovery, and cross-organizational process mining.

Van der Aalst [4] introduces a federated process mining approach to create a unified view of cross-organizational processes. This is achieved by mapping multiple organization-specific event logs into a single federated event log. The paper distinguishes between two types of federated process mining: organizations share filtered event data, and organizations share abstractions, such as DFGs, to maintain higher levels of confidentiality. However, it does not propose a solution for merging these abstractions, leaving this as an open problem for future research, which is addressed in this work.

Rojo et al. [17] explore federated process mining techniques that leverage distributed devices, such as smartphones, to analyze human actions and interactions (including those with other individuals). These techniques conceptualize human behavior as a process and focus on event logs collected from individual devices. The authors propose two approaches: discovering process models directly from the data of individuals and integrating event logs from multiple devices into a centralized dataset to construct process models. While the latter approach involves aggregating traces from multiple devices to a central repository, it often incurs substantial data transfer costs. In contrast, our approach discovers models locally on each device and merges them into a final model, significantly reducing communication overhead.

Khan et al. [12] present a federated approach for cross-silo process mining, utilizing a dependency graph to analyze distributed process logs while maintaining data privacy collaboratively. The authors design a protocol for federated process discovery tailored to the Heuristic Miner process discovery algorithm

[1] https://github.com/HzhianUnimelb/FederatedProcessDiscovery.

that relies on a centralized trusted server to orchestrate communication between the parties that own parts of the entire event log. Our approach can be embedded into similar protocols for federate process discovery. Rafiei and van der Aalst [16] propose an abstraction-based approach for privacy-aware federated process discovery in inter-organizational settings, utilizing directly-follows relations as abstractions of event logs. The approach highlights the importance of enabling organizations to collaborate while safeguarding sensitive information through mechanisms such as handover relations and risk-aware reveal methods. A handover occurs when the execution of a trace transitions from one organization to another. Their method constructs event log abstractions from trace fragments visible to each organization, which are then aggregated into a unified abstraction representing the entire event log. By contrast, we partition the event log into groups of complete traces—for example, those managed by subsidiaries or business units within an organization—and derive abstractions for each group prior to merging them.

Carmona et al. [8] presents two techniques for decomposing process discovery using the theory of regions. The first technique performs a local search for regions rather than globally, and constructs model components from these local regions. The second technique involves selecting groups of related events from the event log, projecting the log onto these groups, and then discovering model components from each of these projections. In both cases, the identified components are then combined into the final process model using parallel composition. Van der Aalst [1] introduces a divide-and-conquer framework for process discovery based on partitioning of actions. This framework also proceeds by splitting the actions into (possibly overlapping) groups, projecting the event log onto these groups, discovering model components from the projections, and composing the components into the resulting process model. Several strategies for decomposing event logs are discussed. For example, one strategy is based on single-entry-single-exit (SESE) fragments, which split the actions based on well-defined sub-processes with unique entry and exit points. Yan et al. [19] present a five-step framework for decomposing process discovery. First, an action relationship graph is derived from the event log, representing the dependencies between actions. This graph is then decomposed into smaller subgraphs, each representing a subset of the process. These subgraphs are used to create corresponding sublogs, which serve as input for constructing submodels. Finally, the submodels are composed to form the complete process model. An instantiation of the framework based on SESE decomposition and heuristic miner is proposed. Different from the existing techniques, we work with multiple collections of complete traces rather than their projections, discover models from these collections, each capturing aspects of the overall process based on a data subset, and, finally, merging, rather than assembling, the intermediate models.

3 Preliminaries

This section presents concepts necessary to understand the contributions discussed in the subsequent sections. A Deterministic Frequency Finite Automaton (DFFA) is a mathematical model that describes traces and their frequencies.

Definition 3.1 (Deterministic Frequency Finite Automata)
A *Deterministic Frequency Finite Automaton* (DFFA) is a tuple $(Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$, where:

- Q is a finite nonempty set of *states*;
- Λ is a finite set of *actions*, called the *alphabet*;
- $\mathbb{I} : Q \to \mathbb{N}$ is the *initial state frequency function*, with one *initial state* $q_0 \in Q$ for which it holds that $\mathbb{I}(q_0) > 0$ and for all $q \in Q$, $q \neq q_0$, it holds that $\mathbb{I}(q) = 0$;
- $\mathbb{F} : Q \to \mathbb{N}$ is the *final state frequency function*; and
- $\delta : Q \times \Lambda \times Q \to \mathbb{N}$ is the *transition frequency function*, such that $\forall q \in Q \forall a \in \Lambda \forall q' \in Q \forall q'' \in Q : \big((\delta(q, a, q') > 0 \wedge \delta(q, a, q'') > 0) \Rightarrow (q' = q'') \big)$. ⌟

The notation $\delta(q, a, q') = n$ indicates that there is a transition from state q to state q' labeled with a, occurring n times. Figure 1 shows a DFFA with states $\{p_0, \ldots, p_4\}$, initial state p_0, $\mathbb{I}(p_0) = 10$, $\mathbb{F} = \{(p_0, 1), (p_1, 4), (p_2, 1), (p_3, 2), (p_4, 2)\}$, and $\delta(p_0, a, p_1) = 4$, $\delta(p_0, b, p_1) = 5$, $\delta(p_1, a, p_2) = 3$, $\delta(p_1, b, p_3) = 2$, $\delta(p_2, a, p_4) = 2$, and $\delta(p_4, a, p_4) = 2$.

A DFFA is consistent if for each of its states the sum of frequencies of entering and leaving (or terminating at) that state is identical.

Definition 3.2 (Consistency [11])
A DFFA $(Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$ is *consistent* if and only if it holds that:

$$\forall q \in Q : \left(\mathbb{I}(q) + \sum_{q' \in Q,\, a \in \Lambda} \delta(q', a, q) \right) = \left(\mathbb{F}(q) + \sum_{q' \in Q,\, a \in \Lambda} \delta(q, a, q') \right).$$

The DFFA in Fig. 1 is consistent. The *simulation* relation associates automata that behave in a similar way, that is, one automaton can "mimic" the actions of the other.

Definition 3.3 (Similarity [18])
DFFA $A_1 = \left(Q^1, \Lambda^1, q_0^1, \mathbb{I}^1, \mathbb{F}^1, \delta^1\right)$ *simulates* DFFA $A_2 = \left(Q^2, \Lambda, q_0^2, \mathbb{I}^2, \mathbb{F}^2, \delta^2\right)$, denoted by $A_2 \preceq A_1$, if and only if there exists a relation $\mathbb{S} \subseteq Q^1 \times Q^2$ such that:

- $\left(q_0^1, q_0^2\right) \in \mathbb{S}$, and
- for every $(p, q) \in \mathbb{S}$, if there exists $a \in \Lambda^1$ and $p' \in Q^1$ such that $\delta^1(p, a, p') > 0$, then it holds that there exists $q' \in Q^2$ such that $\delta^2(q, a, q') > 0$ and $(p', q') \in \mathbb{S}$. ⌟

Our approach to federated process discovery is based on *GASPD*, an evolutionary stochastic process discovery algorithm grounded in grammatical inference [6]. *GASPD* uses *ALERGIA* to extract accurate stochastic language models of traces recorded in an input event log. *ALERGIA* operates by first constructing the prefix automaton of the event log and then iteratively reduces it through recursive merging of states that spawn similar sequences of actions into a consistent DFFA [9].

Algorithm 1 summarizes *GASPD* in pseudocode.

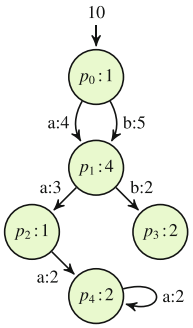


Fig. 1. DFFA A_1

Algorithm 1: *GASPD* [6]

Data: Initial population size p, number of generations m, number of parents k to generate offspring, and event log L
Result: Pareto-optimal DFGs defines by parameter triples

1 $g \leftarrow 0$;
2 $P \leftarrow POPULATION(p)$;
3 **while** $g < m$ **do**
4 $\quad F \leftarrow SELECT(P, L)$;
5 $\quad U \leftarrow CROSSOVER\text{-}MUTATION(F, k)$;
6 $\quad P \leftarrow REPLACE\text{-}ELITE(U, F, L)$;
7 $\quad g \leftarrow g + 1$;
8 **return** $SELECT(P, L)$;

GASPD uses a multi-objective genetic search to discover interesting DFGs from the input event log. It begins by generating a population P of p parameter triplets (Line 2). Each triplet defines a DFG, with one parameter determining which infrequent traces to exclude from the event log and two parameters guiding *ALERGIA* in constructing a DFFA from the remaining frequent traces. Using its native translation mechanism, *GASPD* converts these DFFAs into sound DFGs. In the selection phase (Line 4), the most "fit" population members F are chosen. Fitness is defined by Pareto optimality concerning model size (smaller models are preferred) and entropic relevance of discovered models (lower relevance values are preferred). Relevance serves as a quality criterion because it enables rapid model scoring, balances precision and recall relative to the event log, and quantifies how well the model captures the likelihood of observed traces [6]. Next, in the crossover phase (Line 5), offspring U are generated by applying crossover and mutation operations to k randomly selected parents from F. These operations aim to ensure that offspring inherit the best features of prior generations. Finally, in the replacement phase (Line 6), *GASPD* updates the population by preserving Pareto-optimal members in F and incorporating new, superior members from U.

4 Federated Process Discovery

This section presents the problem of federated (stochastic) process discovery, a technique for merging DFFAs, the *FedGASPD* algorithm that discovers one DFFA from each input event log and then merges them into the final discovered process model, and the *FedCGASPD* algorithm that refines *FedGASPD* to merge similar models to obtain a range of models of different characteristics.

4.1 Definition

Let X be a set. By $\mathcal{B}(X)$, we denote the set of all finite multisets over X, while, by X^*, we denote the set of all finite sequences over X. Let $\mathbb{X} \in X^*$ be a sequence over X. Then, by $\mathbb{X}(i)$, $i \in [1 \,..\, |\mathbb{X}|]$, we denote the element at position i in $\mathbb{X}$.

Let $\mathcal{A}$ be the universe of *actions*. Then, $\mathcal{L} = \mathcal{B}(\mathcal{A}^*)$ is the universe of *event logs*. Let $\mathcal{M}$ be the universe of *process models*. Given an event log $L \in \mathcal{L}$, the conventional *process discovery* problem studies ways to construct a process model that describes the system that generated L. A solution to the process discovery problem can be captured as a function $d : \mathcal{L} \to \mathcal{M}$. The *federated process discovery* problem studies ways to construct a process model from a list of event logs, where each event log from the list can be stored on a dedicated device at a particular organization. That is, a solution to the federated process discovery problem can be given as a function $f : \mathcal{L}^* \to \mathcal{M}$.

Let $q : \mathcal{L} \times \mathcal{M} \to [0, 1]$ be a function that measures *quality* $q(L, M)$, $L \in \mathcal{L}$, $M \in \mathcal{M}$ of model M discovered from event log L. For instance, measure q can be precision, recall, simplicity, or generalization [7]. Such a measure usually associates larger values with models that describe the system better. A good federated process discovery approach f should construct a model from multiple event logs that is comparable in quality to the model constructed from the corresponding centralized event log:

$$\forall\, \mathbb{L} \in \mathcal{L}^* : q(L, f(\mathbb{L})) \approx q(L, d(L)), \text{ where } L = \uplus_{i=1}^{|\mathbb{L}|} \mathbb{L}(i).$$

In this paper, we solve the federated process discovery problem by first constructing process models from individual event logs using the conventional approach and then merging the obtained models into the final process model. Let $m : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ be a function that merges models. Specifically, given a non-empty, finite list of event logs $\langle L_1, \ldots, L_n \rangle \in \mathcal{L}^*$, $n \in \mathbb{N}$, we first discover the sequence of models $\langle M_1, \ldots, M_n \rangle$, $M_i = d(L_i)$, $i \in [1 \,..\, n]$, using the conventional approach on each event log. Each such model can be constructed on a dedicated device at the organization that owns the data and is an abstract, and thus more confidential [4], representation of the corresponding event log. We then merge the models incrementally using function $F_m : \mathcal{M}^* \to \mathcal{M}$:

$$F_m(\langle M_1 \rangle) = M_1 \text{ and } F_m(\langle M_1, M_2, \ldots, M_n \rangle) = F(\langle m(M_1, M_2), \ldots, M_n \rangle).$$

In general, different orders and approaches to model merging can lead to different resulting models. Next, we discuss an approach to merging DFFAs.

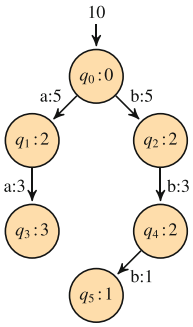


Fig. 2. DFFA A_2

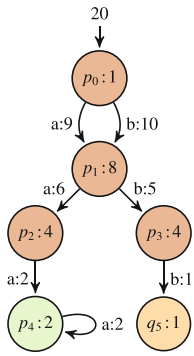


Fig. 3. DFFA A

4.2 Model Merging

We refer to the procedure for merging a DFFA A_2 into a DFFA A_1 as *MergeDFFA*. In addition to the two automata, the procedure takes as input a parameter $d \in \mathbb{N}$. The first step is to construct the (prefix of the) unfolding of A_2 truncated at the $d+1$ occurrences of its states [10]. In this prefix automaton, state and transition frequencies are derived from those in A_2 by propagating frequencies along the unfolding branches, using probabilities estimated based on the original frequencies in A_2. *MergeDFFA* then "embeds" this prefix automaton into A_1 by reusing the structure of A_1 and extending it as necessary to incorporate any remaining parts of the prefix. During this embedding process, the frequencies from A_1 and the constructed prefix of A_2 are aggregated. We denote the result of merging A_2 into A_1 using parameter d as $A_1 \triangleleft_d A_2$.

DFFA A in Fig. 3 is the result of merging A_2 in Fig. 2 into A_1 in Fig. 1 using $d = 1$, that is, $A = A_1 \triangleleft_1 A_2$. In the figure, the khaki-colored states represent states from A_1 that were reused during the embedding of the prefix unfolding of A_2 into A_1. The green state p_4 corresponds to a state of A_1 that was not visited during the embedding, while the orange state q_5 originates from A_2 and was used to extend the structure of A_1.

Since A_2 is acyclic and has no states with multiple incoming transitions, the prefix of its unfolding for any $k \in \mathbb{N}$ is structurally identical to A_2. The merged automaton A incorporates all transitions from both A_1 and A_2, and it permits termination in the same states as these two automata. Consequently, A simulates both A_1 and A_2. When embedding A_2 into A_1, the transitions from A_2 "follow" those of A_1 as closely as possible accumulating transition and state termination frequencies from both automata, effectively merging their behaviors. After executing b twice, both A_1 and A_2 reach a common merged state p_3. At this point, A_1 terminates, while A_2 is capable of performing an additional b transition. This leads to the introduction of a new state q_5, extending the behavior of A beyond A_1 to simulate the behavior of A_2.

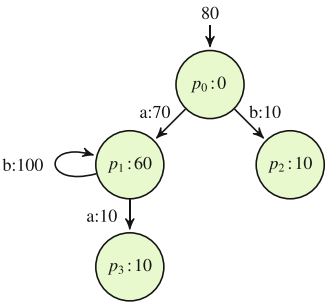


Fig. 4. DFFA A_3

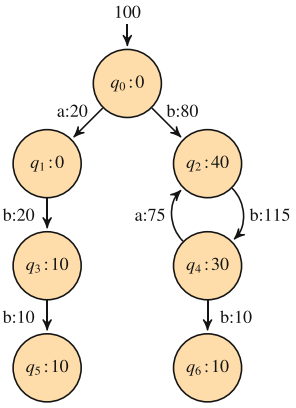


Fig. 5. DFFA A_4

Figure 6 shows the prefix of the unfolding of the DFFA A_4 from Fig. 5 computed for $d = 2$. The construction begins at the initial state of the input automaton and proceeds iteratively: at each step, a state in the current prefix is selected and extended using one of its outgoing transitions from the input automaton. Each extension introduces a fresh state to represent the target of the chosen transition, ensuring that the unfolding forms a tree. To guide the construction, only states with unexplored transitions are selected for expansion. If the target state of a transition has already appeared d times among the ancestors of the current node in the prefix, the expansion is halted; otherwise, if the input DFFA has a cycle, the prefix construction will continue forever. In the example prefix shown in Fig. 6, the state q'_4 corresponds to state q_4 in A_4. However, it is not expanded via transition a because doing so would introduce a third occurrence of state q_2 along the path from q'_4 to the root (state q_0). To ensure the consistency of the prefix automaton, the frequency of a pruned state is accumulated in the parent state (state q'_4 in the example). The frequencies of states and transitions in the prefix automaton are computed by multiplying the frequency of each concept in the input automaton by the probability of reaching its corresponding occurrence during the execution of the input automaton. This probability is estimated based on the state and transition frequencies in the input automaton.

DFFA B in Fig. 7 is the result of merging A_4 from Fig. 5 into A_3 from Fig. 4 using $d = 2$, that is, $B = A_3 \triangleleft_2 A_4$. The khaki-colored states indicate states from A_3 that were reused during the embedding of the prefix of A_4 from Fig. 6 into A_3. The green states correspond to the original states of A_3 not visited during the embedding of the prefix, while the orange states are the states of the prefix that do not "fit" into A_3.

MergeDFFA terminates after it explores the necessary pairs of states from the two input automata starting from the pair of the initial states. The procedure always terminates because the number of all possible such state pairs is finite. The runtime of the merging process is primarily determined by the size of the

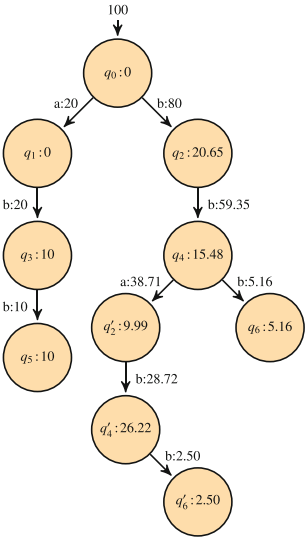


Fig. 6. Prefix of DFFA A_4 ($d = 2$)

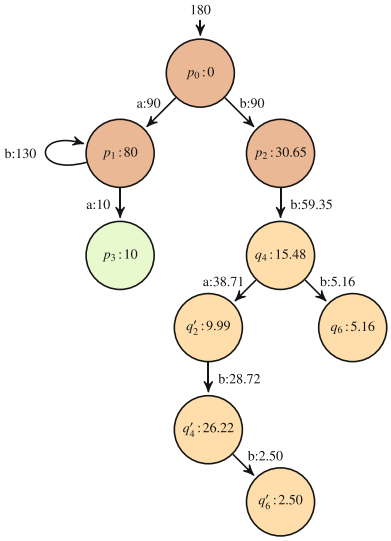


Fig. 7. DFFA B

prefix of the unfolding of DFFA A_2, which, in the worst case, can be exponential in the size of A_2. When the parameter d is set to a small value, the size of the prefix unfolding is typically only a few times larger than that of A_2. However, increasing d allows the behavior of A_2 to be captured more accurately in the resulting merged automaton.

The automaton that results from merging two consistent automata is consistent.

Lemma 4.1 (Consistency)
If A_1 and A_2 are consistent DFFAs, then DFFA A, such that $A = A_1 \triangleleft_d A_2$, $d \in \mathbb{N}$, is consistent.

Lemma 4.1 holds because the prefix automaton that *MergeDFFA* embeds into A_1 is, by construction, consistent. Furthermore, each time a state from the prefix of the unfolding is embedded into the resulting automaton, the *MergeDFFA* procedure updates both the termination frequency of the merged state and the frequencies of its outgoing transitions to preserve consistency.

The automaton produced by *MergeDFFA* is in the following simulation relations with the input automata.

Lemma 4.2 (Similarity)
If A_1 and A_2 are consistent DFFAs and $A = A_1 \triangleleft_d A_2$, $d \in \mathbb{N}$, it holds that:

- *A simulates A_1, that is, $A_1 \preceq A$; and*
- *if A_2 is acyclic, then A simulates A_2, that is, $A_2 \preceq A$.* ⌟

Lemma 4.2 holds based on the following observations. Each automaton produced by *MergeDFFA* is constructed by extending the structure of DFFA A_1, thus preserving all of its transitions. Also, *MergeDFFA* ensures that all transitions from the prefix of the unfolding of A_2 are retained in the resulting DFFA. If A_2 is acyclic, the unfolding of A_2 simulates A_2. Finally, the unfolding of an acyclic automaton is isomorphic to the prefix of the unfolding of this automaton for any truncation depth $d > 0$.

Notably, given DFFAs A_1 and A_2 and $d \in \mathbb{N}$, it is not always the case that $A_1 \triangleleft_d A_2 = A_2 \triangleleft_d A_1$; that is, the merging operation is not commutative. This fact suggests that if *MergeDFFA* is used to merge models when solving the federated process discovery problem, different orders of mergings will indeed lead to different discovered models.

4.3 Discovering Single Model

Given parameters to configure the *GASPD* algorithm and a collection of event logs, each stemming from the same process but possibly recorded by different organizations or different departments within a single organization, the *FedGASPD* algorithm solves the federated process discovery problem by constructing one model from each event log and then merging them into the final process model; refer to Algorithm 2.

Algorithm 2: *FedGASPD*

Input: Initial population size p, number of generations m, number of parents k to generate offspring, and a non-empty list of event logs $L = \langle L_1, \ldots, L_n \rangle$
Output: A DFFA discovered from L

1 $M \leftarrow \langle\rangle$;
2 **parallelfor** $i \in [1 \,..\, n]$ **do**
3 $\quad M \leftarrow M \circ \langle \mathit{SELECTDFFA}\,(\mathit{GASPD}\,(p, m, k, L_i))\rangle$;
4 **return** $F_{\mathit{MergeDFFA}}\,(M)$;

To discover a model from an input event log, we first use the *GASPD* algorithm (Algorithm 1) to identify Pareto-optimal models based on their size and entropic relevance. From the set of Pareto-optimal models, one is selected using the *SELECTDFFA* function. The selected model is then added to the list of constructed DFFAs, M, using the sequence concatenation operator "$\circ$" (Line 3). Since the above-outlined procedure for obtaining a model can be performed independently for each input event log, we execute all such procedures for all input event logs in parallel (Line 2). Various model selection policies can be implemented by different implementations of the *SELECTDFFA* function. Finally, once all models are selected, they are merged pairwise using the *MergeDFFA* procedure (Line 4); cf. Sect. 4.2 for details on the merging procedure.

The discovery of Pareto-optimal models and the selection of one optimal model for each input event log can be performed locally on devices at the respective organizations where the input event logs are stored. Once the models are selected, they can be transmitted to a central server, where the merging process takes place.

Note that *GASPD* identifies parameter triplets that trigger *ALERGIA* to construct automata that can be translated into interesting DFGs. The *SELECTDFFA* function, thus, selects one parameter triplet identified by *GASPD* and constructs the corresponding DFFA using *ALERGIA*. After merging all DFFAs into a final model, this model can be trivially translated into a probabilistic automaton using Algorithm 16.1 by de la Higuera [11], which, in turn, can be trivially converted into a DFG model following the procedure outlined in Definition 4.2 by Alkhammash et al. [6]. To ensure brevity of presentation and because the quality of DFFAs directly impacts the quality of the resulting DFGs, here, we design the core techniques to operate with DFFAs.

4.4 Discovering Multiple Models

FedGASPD solves federated process discovery in a straightforward manner: it discovers one model from each input event log and merges them. This approach gives a basic baseline for federated process discovery. However, when the discovered models exhibit diverse characteristics, merging them may result in a loss of distinctiveness and dilute their strong individual features due to the averaging effect. To address this limitation, we propose a variation of *FedGASPD* that avoids merging all models into one. Instead, it groups similar models into clusters and merges the models within each cluster, yielding multiple models that retain similar, strong characteristics of the merged models. We refer to this variant as *FedCGASPD*, which is summarized in Algorithm 3.

Algorithm 3: *FedCGASPD*

Input: Initial population size p, number of generations m, number of parents k to generate offspring, and a non-empty list of event logs $L = \langle L_1, \ldots, L_n \rangle$
Output: A list of DFFAs discovered from L

1 $M \leftarrow \langle\rangle$; $\mathbb{M} \leftarrow \langle\rangle$;
2 **parallelfor** $i \in [1 \,..\, n]$ **do**
3 $\quad M \leftarrow M \circ \langle SELECTDFFA\,(GASPD\,(p, m, k, L_i))\rangle$;
4 $C \leftarrow KMeansCluster\,(M, n)$;
5 **for** $i \in [1 \,..\, n]$ **do**
6 $\quad$ **if** $|C\,(i)| \geq 0$ **then** $\mathbb{M} \leftarrow \mathbb{M} \circ \langle F_{MergeDFFA}\,(C\,(i))\rangle$;
7 **return** $\mathbb{M}$;

After discovering models from input event logs (Lines 2 and 3 in Algorithm 3), they are clustered using the k-means algorithm [13] via the *KMeansCluster* function (Line 4). The *KMeansCluster* function takes as input the list of models,

M, and the desired number of clusters, n, and outputs a list of clusters, C, where each cluster is stored as a list of models. We specify n as the number of desired clusters and initialize n random centroids (initial cluster centers), each represented as a point in the space $[0, 1] \times [0, 1]$. If there are fewer than n meaningful clusters, the k-means algorithm identifies these clusters by iteratively refining the centroids while allowing some centroids to remain unused. For clustering, each DFFA is represented as a point in a two-dimensional space, defined by its normalized size and entropic relevance. Normalization is achieved by dividing the size and entropic relevance of each model by the respective maximum values across all models in M. Then, the *MergeDFFA* procedure is applied to each non-empty cluster. All models within a cluster are merged, and the resulting merged model is appended to the list of discovered models (Line 6).

5 Evaluation

We implemented the *GASPD*, *FedGASPD*, and *FedCGASPD* algorithms and made them publicly available.[2] Using this implementation, we evaluated the performance of these algorithms using nine real-world event logs, which we sourced from the IEEE Task Force on Process Mining (https://www.tf-pm.org/resources/logs). For experiments, we selected event logs that exhibit a wide range of characteristics, such as the number of (distinct) actions, the number of (distinct) traces, and trace length. This diversity is essential for evaluating the robustness of our algorithms in various real-world scenarios. Table 1 summarizes the characteristics of the nine event logs.

Table 1. Characteristics of the event logs used in our experiments

Event log	#Actions	#Traces	#Distinct traces	Max. trace length	Avg. trace length
BPI-2012	24	13,087	4,366	175	20.03
BPI-2013	4	7,554	1,511	123	8.67
BPI-2017	26	31,509	15,930	180	38.15
BPI-2018	34	2,861	2,498	680	61.58
BPI-2019	42	251,734	11,973	990	06.33
BPI-2020-1	34	6,449	753	27	11.18
BPI-2020-2	19	6,886	89	20	05.34
Sepsis Cases	16	1,050	846	185	14.48
nasa-cev	47	2,566	2,513	50	28.69

All experiments were executed on a computer featuring a 13th Gen Intel® Core™ i7-1355U processor running at 1.70 GHz using 16.0 GB of RAM (15.7 GB of effective memory). The aim of the experiments is twofold. Firstly, we study the feasibility of using federated process discovery in industrial settings. Secondly, we

[2] https://github.com/HzhianUnimelb/FederatedProcessDiscovery.

Table 2. Execution times of *GASPD*, *FedGASPD*, and *FedCGASPD* algorithms

Event log	Execution time (in seconds)						
	GASPD	*FedGASPD*			*FedCGASPD*		
		2 nodes	4 nodes	8 nodes	2 nodes	4 nodes	8 nodes
BPI-2012	416	178	83	61	166	81	58
BPI-2013	84	35	26	11	43	22	17
BPI-2017	3,120	1,459	514	342	1,401	499	367
BPI-2018	2,898	1,325	165	76	1,317	173	78
BPI-2019	2,288	2,216	471	251	2,224	480	260
BPI-2020-1	1402	685	331	229	676	342	236
BPI-2020-2	175	94	57	46	92	53	42
Sepsis Cases	421	234	64	31	237	60	36
nasa-cev	3,820	575	329	136	568	332	142

compare the quality of the models generated by *FedGASPD* and *FedCGASPD* to those produced using the centralized *GASPD* method. For each event log, *GASPD*, *FedGASPD*, and *FedCGASPD* were initialized the initial population size, p, and the number of generations, m, to 50. To construct human-readable models, we controlled the parameter responsible for the exclusion of infrequent traces. Consequently, we constructed models of at most 1,000 nodes and arcs, with many models substantially smaller than this target size.

To simulate cross-organizational environments, each event log was split randomly among a requested number of computational nodes, ensuring that each node has approximately the same portion of the event log traces. We implemented the *SELECTDFFA* function from Algorithms 2 and 3, by selecting one DFFA from a Pareto-frontier of DFFAs that minimizes this objective function: $(1 - size^*) + (1 - relevance^*)/2$, where $size^*$ and $relevance^*$ are the normalized size and entropic relevance of the model, with normalization performed as described in Sect. 4.4. This way, we select a model that balances the two competing objectives equally.

Table 2 presents the execution times of *GASPD* and its federated alternatives. Note that *FedGASPD* and *FedCGASPD* are substantially faster than *GASPD*. For example, for the nasa-cev event log, the federated discovery techniques are up to 26 times faster than *GASPD* and, at a minimum, 5 times faster, refer to BPI-2013 event log. This observation can be explained by the fact that *FedGASPD* and *FedCGASPD* perform computationally demanding *GASPD* genetics on many smaller event logs in parallel and then apply a computationally efficient merging of the resulting models.

Figure 8 shows the characteristics of the Pareto-optimal models discovered from the nine event logs by the three algorithms for the different numbers of computation nodes, while Tables 3 and 4 summarize the ranges of the sizes and entropic relevance values of the discovered models. These results further support our contention that federated approaches yield useful new models compared to the centralized solution. When comparing the two federated approaches, *Fed-*

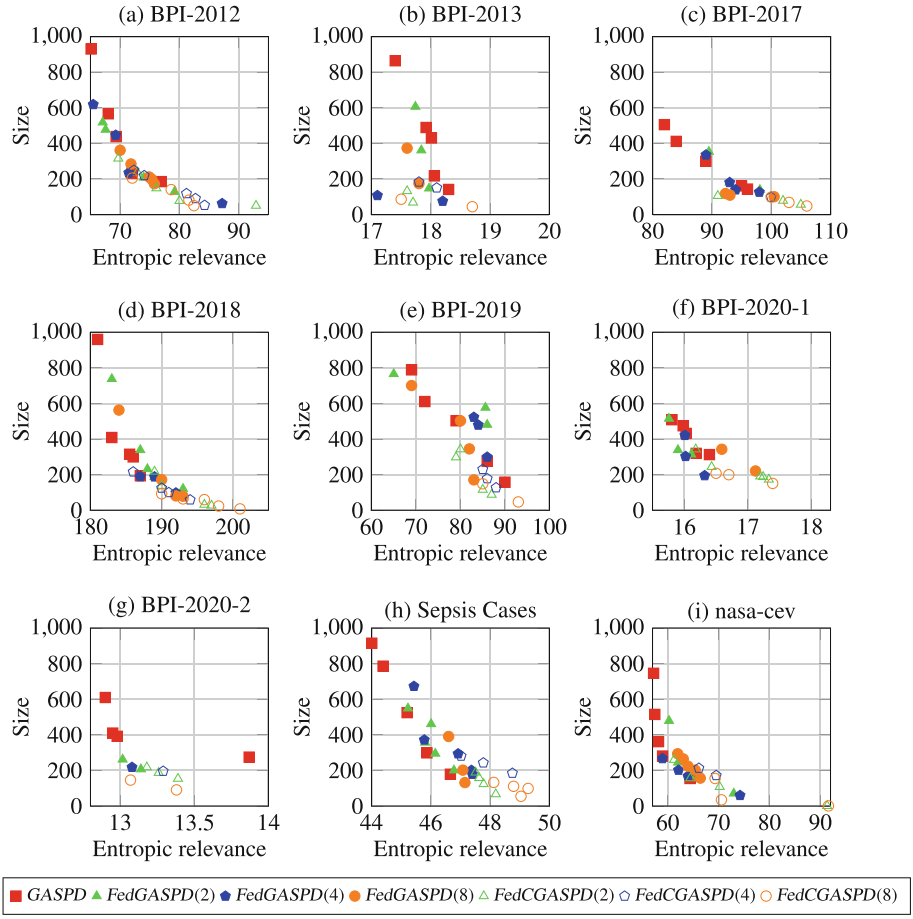


Fig. 8. Size and entropic relevance of DFFAs discovered by *GASPD*, *FedGASPD*, and *FedCGASPD*; smaller sizes and entropic relevance values signify better models. The numbers in the parenthesis signify the numbers of computation nodes.

CGASPD results more often in smaller models than *FedGASPD*, as it tends to merge models of similar characteristics and, hence, preserve the distinctive sizes of the various groups of similar models. *FedGASPD*, however, tends to discover models of lower entropic relevance, hence describing the traces and their frequencies from the input event logs more accurately. This observation confirms the trade-off between the size and quality of the discovered models [14] and suggests that the two proposed methods prioritize different aspects of this trade-off.

We conclude that *FedGASPD* and *FedCGASPD* discover models of comparable characteristics as *GASPD*. Table 5 presents Pareto dominance counts for the various techniques. Note that the centralized solution, *GASPD*, does not

consistently dominate across different datasets. Specifically, it delivers a good share of Pareto-optimal solutions for BPI-2017, BPI-2020-2, Sepsis Cases, and nasa-cev event logs, but not for other event logs, where *FedCGASPD* discovers the lion's share of Pareto-optimal models.

Table 3. Size of DFFAs discovered by *GASPD*, *FedGASPD*, and *FedCGASPD*

Event Log	Size												
	GASPD		*FedGASPD*						*FedCGASPD*					
			2 nodes		4 nodes		8 nodes		2 nodes		4 nodes		8 nodes	
	min	max	min	max	min	max	min	max	min	max	min	max	min	max
BPI-2012	185	930	128	856	173	361	52	249	62	446	50	315	11	205
BPI-2013	486	760	143	715	174	373	130	291	107	311	67	218	47	190
BPI-2017	143	460	166	507	100	184	98	121	125	7500	56	154	48	121
BPI-2018	196	403	120	339	81	282	59	125	98	318	27	218	8	66
BPI-2019	158	808	481	766	171	503	127	230	229	632	116	572	47	218
BPI-2020-1	313	509	314	515	221	465	187	369	195	421	173	321	151	235
BPI-2020-2	275	629	234	459	247	362	196	283	213	345	153	266	90	241
Sepsis Cases	179	786	201	549	179	673	132	391	132	189	149	314	43	133
nasa-cev	155	746	71	514	60	576	157	294	1	260	172	241	1	155

Table 4. Relevance of DFFAs discovered by *GASPD*, *FedGASPD*, and *FedCGASPD*

Event Log	Entropic relevance													
	GASPD		*FedGASPD*						*FedCGASPD*					
			2 nodes		4 nodes		8 nodes		2 nodes		4 nodes		8 nodes	
	min	max	min	max	min	max	min	max	min	max	min	max	min	max
BPI-2012	65.11	76.89	64.84	79.13	70.00	75.17	72.32	84.23	65.47	87.19	72.91	92.90	72.05	98.00
BPI-2013	17.40	18.10	17.72	18.25	17.65	18.46	17.82	18.71	16.33	18.29	17.70	18.83	17.51	20.77
BPI-2017	82.01	93.34	89.79	95.42	92.63	103.04	100.88	105.45	89.69	100.18	93.88	106.57	101.31	110.33
BPI-2018	185.26	187.42	187.77	193.08	191.87	194.16	190.44	197.13	189.25	193.34	192.33	198.58	193.01	205.07
BPI-2019	61.01	87.27	64.77	87.39	69.51	88.59	70.65	89.66	76.40	88.69	81.56	92.22	83.39	94.73
BPI-2020-1	15.80	16.39	15.75	16.30	16.59	17.13	16.30	17.61	16.00	16.75	16.18	17.33	16.70	18.35
BPI-2020-2	12.95	13.78	13.07	13.47	13.09	13.85	13.40	13.53	13.06	13.25	13.17	13.44	13.40	13.75
Sepsis Cases	44.01	46.66	45.23	46.67	45.24	47.45	46.6	47.51	47.45	48.32	47.01	48.85	48.12	50.03
nasa-cev	57.21	64.37	60.25	72.98	58.91	74.21	62.95	66.39	61.23	91.95	66.11	69.52	67.91	91.59

Table 5. Pareto dominance counts

Event log	*GASPD*	*FedGASPD*			*FedCGASPD*			Total score
		2 nodes	4 nodes	8 nodes	2 nodes	4 nodes	8 nodes	
BPI-2012	2 (10.52%)	**4 (21.05%)**	2 (10.52%)	0 (00.00%)	3 (15.78%)	0 (00.00%)	**4 (21.05%)**	19 (100.00%)
BPI-2013	0 (00.00%)	0 (00.00%)	1 (25.00%)	0 (00.00%)	1 (25.00%)	0 (00.00%)	**2 (50.00%)**	04 (100.00%)
BPI-2017	**3 (30.00%)**	0 (00.00%)	0 (00.00%)	0 (00.00%)	**3 (30.00%)**	1 (10.00%)	**3 (30.00%)**	10 (100.00%)
BPI-2018	3 (15.00%)	1 (05.00%)	3 (15.00%)	3 (15.00%)	3 (15.00%)	3 (15.00%)	**4 (20.00%)**	20 (100.00%)
BPI-2019	2 (20.00%)	1 (10.00%)	0 (00.00%)	2 (20.00%)	**3 (30.00%)**	0 (00.00%)	2 (20.00%)	10 (100.00%)
BPI-2020-1	1 (10.00%)	2 (20.00%)	2 (20.00%)	0 (00.00%)	**3 (30.00%)**	1 (10.00%)	1 (10.00%)	10 (100.00%)
BPI-2020-2	**4 (57.14%)**	1 (14.28%)	0 (00.00%)	0 (00.00%)	0 (00.00%)	0 (00.00%)	2 (28.58%)	07 (100.00%)
Sepsis Cases	**5 (38.46%)**	2 (15.38%)	1 (07.69%)	1 (07.69%)	2 (15.38%)	0 (00.00%)	2 (15.38%)	13 (100.00%)
nasa-cev	**4 (28.57%)**	1 (07.14%)	3 (21.42%)	0 (00.00%)	3 (21.42%)	0 (00.00%)	3 (21.42%)	14 (100.00%)

6 Conclusions

This paper presents the *FedGASPD* and *FedCGASPD* algorithms for stochastic process discovery in distributed environments. These algorithms are built upon the robust foundation of *GASPD*, a stochastic process discovery algorithm that operates over centralized event logs. The new algorithms discover process models from several local event logs, possibly scattered across different organizations, and aim to preserve the autonomy and privacy of each party and to decrease data communication requirements and the overall model discovery time. Our experiments with real-world event logs demonstrate the effectiveness of *FedGASPD* and *FedCGASPD*, providing scalable alternatives for process discovery in distributed environments. The algorithms discover DFFAs that can be translated, using the native translation of *GASPD*, into sound DFGs.

Although the presented results suggest that the models constructed by *FedGASPD* and *FedCGASPD* are comparable in quality to those discovered by *GASPD*, several avenues exist to further enhance the findings. First, a systematic exploration of the impact of different orders of merging DFFAs discovered from local event logs on the quality of the constructed global DFFAs and, consequently, DFGs, could provide valuable insights. Similarly, investigating clustering techniques that improve the performance of *FedCGASPD* may lead to significant advancements. Additionally, optimizing the discovered models for other quality criteria than size and relevance could further refine their applicability. Finally, exploring alternative policies for selecting superior local models represents an interesting direction for future research.

References

1. van der Aalst, W.: Decomposing Petri nets for process mining: a generic approach. Distrib. Parallel Databases **31**(4), 471–507 (2013)
2. van der Aalst, W.: Process Mining-Data Science in Action, 2nd edn. Springer, Heidelberg (2016)
3. van der Aalst, W.: A practitioner's guide to process mining: Limitations of the directly-follows graph. Procedia Comput. Sci. **164**, 321–328 (2019)

4. van der Aalst, W.M.P.: Federated process mining: exploiting event data across organizational boundaries. In: SMDS, pp. 1–7. IEEE (2021)
5. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
6. Alkhammash, H., Polyvyanyy, A., Moffat, A.: Stochastic directly-follows process discovery using grammatical inference. In: CAiSE. LNCS, vol. 14663, pp. 87–103. Springer, Cham (2024)
7. Buijs, J., van Dongen, B.F., van der Aalst, W.: Quality dimensions in process discovery: the importance of fitness, precision, generalization and simplicity. Int. J. Coop. Inf. Syst. **23**(01), 1440001 (2014)
8. Carmona, J., Cortadella, J., Kishinevsky, M.: Divide-and-conquer strategies for process mining. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 327–343. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03848-8_22
9. Carrasco, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In: Carrasco, R.C., Oncina, J. (eds.) ICGI 1994. LNCS, vol. 862, pp. 139–152. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58473-0_144
10. Esparza, J., Heljanko, K.: Unfoldings: A Partial-Order Approach to Model Checking. EATCS Monographs in Theoretical Computer Science. Springer, Cham (2008)
11. de la Higuera, C.: Grammatical Inference. Cambridge University Press, Cambridge (2010)
12. Khan, A., Ghose, A., Dam, H.: Cross-silo process mining with federated learning. In: Hacid, H., Kao, O., Mecella, M., Moha, N., Paik, H. (eds.) ICSOC 2021. LNCS, vol. 13121, pp. 612–626. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91431-8_38
13. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press (1967)
14. Polyvyanyy, A., Moffat, A., Garcia-Banuelos, L.: An entropic relevance measure for stochastic conformance checking in process mining. In: ICPM, pp. 97–104. IEEE (2020)
15. Polyvyanyy, A., Smirnov, S., Weske, M.: Process model abstraction: a slider approach. In: EDOC, pp. 325–331. IEEE Computer Society (2008)
16. Rafiei, M., van der Aalst, W.M.P.: An abstraction-based approach for privacy-aware federated process mining. IEEE Access **11**, 33697–33714 (2023). ISSN 2169-3536
17. Rojo, J., Garcia-Alonso, J., Berrocal, J., Hernández, J., Murillo, J.M., Canal, C.: SOWCompact: a federated process mining method for social workflows. Inf. Sci. **595**, 18–37 (2022)
18. Sangiorgi, D.: On the origins of bisimulation and coinduction. ACM Trans. Program. Lang. Syst. **31**(4), 15:1–15:41 (2009)
19. Yan, Z., et al.: Decomposed and parallel process discovery: a framework and application. Future Gener. Comput. Syst. **98**, 392–405 (2019)

Object-Centric Causal Nets

Lukas Liss(✉), Caspar Mensing, and Wil M. P. van der Aalst

RWTH Aachen University, Aachen, Germany
{liss,wvdaalst}@pads.rwth-aachen.de, caspar.mensing@rwth-aachen.de

Abstract. Traditional process mining techniques are constrained by the use of a single case identifier. To address this, object-centric process mining was developed. Current object-centric process models like object-centric Petri nets are restricted by representational biases, for example concerning the concrete distribution of objects to their succeeding activities (abstracted by places) or optionally involved object types (abstracted using variable arcs). This paper presents four key contributions to object-centric process discovery, to overcome the mentioned representational biases. First, we define object-centric causal nets, allowing for the modeling of complex process behavior. Second, we introduce a baseline algorithm for mining these nets from object-centric event logs. Third, we provide a publicly available implementation. Fourth, we evaluated the implementation quantitative and qualitatively, demonstrating the algorithm's applicability across diverse event logs.

Keywords: Object-Centric Process Mining · Process Model · Process Discovery

1 Introduction

The real-world consist of complex processes comprising multiple subprocesses that operate on various objects from different types [8]. To handle the growing complexity, information systems are used to support these processes. They track the real process behavior of all the digitally traceable objects involved in the process. The goal of process discovery in information system management is to understand the real process and, based on that, improve the information system and the process itself [4]. To do so, it is essential to represent the real process as accurately as possible. It has been shown that an object-centric process perspective, which models the behavior of multiple interacting objects of a process at once, results in more accurate results because it is free of problems like divergence, convergence, and deficiency, compared to traditional process discovery [19]. Current object-centric process discovery approaches use design-oriented models like object-centric Petri net [21], which comes with a certain representational bias, which can lead to imprecise and undesirable process models, as demonstrated in the running example of an order handling process.

The running example process is depicted as an object-centric event log [2] in Table 1. This type of information can be extracted from information systems

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 94–110, 2025.
https://doi.org/10.1007/978-3-031-94571-7_6

Table 1. Excerpt of the object-centric event log of the running example.

ID	activity	time	order	container	box
1	a (check order)	1	o1		
2	b (register order)	2	o1		
3	b (register order)	3	o2		
4	c (get container)	4		c1	
5	s (send)	5	o1,o2	c1	
6	r (receive feedback)	6	o1,o2		
7	i (investigate)	7	o1		
8	n (no investigate)	8	o2		
9	b (register order)	9	o3		
10	d (get box)	10			b1
11	s (send)	11	o3		b1
12	r (receive feedback)	12	o3		
13	i (investigate)	13	o3		

to understand the process and then improve the information system. In the example process, orders can be checked first (see order *o1*) or directly registered (see order *o2*). Then they are *send* either using a container when multiple orders are shipped together (see e5) or in a box when only one order is sent (see e11). In the end, after receiving feedback for the orders sent together, the company investigates only one of the orders and does not investigate the others (see e7, e8, and e13). All three described process characteristics require artificial constructs or can not be modeled correctly using an object-centric Petri net, as one can see in the object-centric Petri net for the running example in Fig. 1.

Object-centric Petri nets suffer from the following three representation biases: (1) They require artificial constructs like silent transitions (represented as black boxes) and places, which both have no correspondence in the event data [20]. (2) Object-centric Petri nets without duplicated labels (the only ones that can be discovered) overgeneralize optional object types by using variable arcs. This can be seen in the *send* activity, which is connected with variable arcs to the container and the box. This allows this activity to use multiple containers and boxes for the same *send* event, which is not represented in the process. Also, it is no longer modeled that individual orders are always sent via boxes due to the generalizing behavior of variable arcs. (3) The distribution of objects to parallel following events is not clear in object-centric Petri nets. In the example, precisely one of the orders that received feedback together is investigated, and the others are not. The Petri net also allows to investigate all, some or none.

To overcome these representational biases, we propose a new object-centric process model: object-centric causal nets. First, we define the syntax and semantics of object-centric causal nets. Second, we propose a discovery approach to mine them from event logs. Third, we provide a publicly accessible implementation of the miner. Finally, we perform a qualitative and quantitative evaluation using our implementation and publicly accessible event logs.

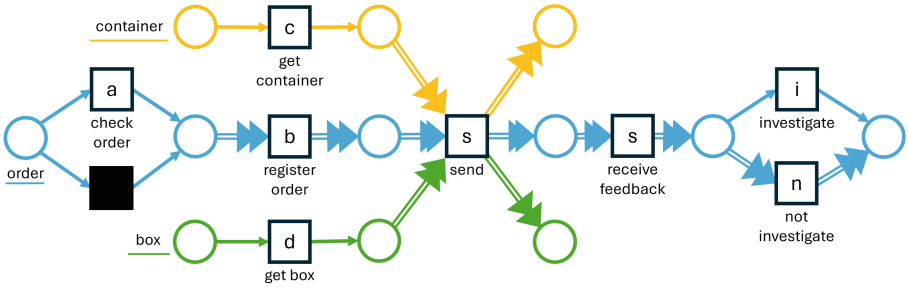


Fig. 1. Object-centric Petri net for the running example used to illustrate three limitations induced by the representational bias of object-centric Petri nets.

2 Related Work

Our approach belongs to the field of process model discovery [1] and is therefore part of the continuous improvement life cycle of information systems which is the overall goal of information system engineering research [24]. This involves a process-aware information system, where the differences between the real process and the designed process, as supported by the information system, can be discovered in a data-driven fashion [18,25]. Differences between the intended and real process behavior can then be used to improve the information system. Process discovery [23] is used in the information system engineering life-cycle to uncover the real process. The object-centric process discovery approach presented in this paper aims to improve the discovery of the actual system process, thereby improving the information system life cycle.

Many process models and also multiple discovery algorithms have been proposed over the years. Where traditional process discovery assumes a single case identifier, object-centric process discovery allows for multiple interacting objects in a process [8]. Thus making it possible to mine system behavior instead of isolated processes as conceptualized in [9]. It has been shown that object-centric approaches can improve the quality of the discovered models because they do not suffer from problems like divergence, convergence, and deficiency [19]. Thus, the research interest recently shifted towards object-centric process analysis, but most models and algorithms do not yet support object-centric processes.

In traditional process mining, there are model notations and miners, for example, for directly follows graphs, Petri nets [15], process trees [12], and causal nets [20]. Each model has unique representational biases, making it suited for specific tasks and less suited for others. Some of them have been generalized to the object-centric setting, like directly follows graphs [14], Petri nets [21], process trees [22], and DCR graphs [6], which orchestrate the lifecycle of object types and support instance spawning and one-to-many relations between objects. There exist also new discoverable object-centric exclusive process models like TOTeM models [13]. Other models like PHILharmonicFlows [11], the MERODE approach [16], or object-centric Petri nets with identifiers [10] also try to offer

different representational biases, but can not be discovered from event data. Current discoverable object-centric process models still have certain representational biases, so we propose object-centric causal nets to overcome them. More concretely, we want a model that does not use artificial constructs for skipping behavior, can clearly model optional object type involvement for activities, and explicitly models object distributions over following activities. The proposed object-centric causal net generalizes the general model idea of traditional causal nets [20] to object-centric processes and builds upon causal discovery algorithms like the flexible heuristic miner [26].

3 Preliminaries

Object-centric process mining deals with events that operate on a variety of objects of different types. Events are activities that happen at a timestamp for a number of objects of different types. $\mathbb{U}_{event}$ is the universe of event identifiers. The universe $\mathbb{U}_{act}$ contains all visible activities. $\mathbb{U}_{typ}$ is the universe of all object types. The universe of objects is $\mathbb{U}_{obj}$. Each object has exactly one type associated with it $\pi_{type} : \mathbb{U}_{obj} \rightarrow \mathbb{U}_{typ}$. $\mathbb{U}_{time}$ is the universe of all timestamps. This is the most used subset of the object-centric event log format OCEL 2.0 [2].

Definition 1 (Event Log). *Event log* $L = (E, O, OT, \pi_{act}, \pi_{obj}, \pi_{time})$ *with:*

- $E \subseteq \mathbb{U}_{event}$ *is a set of events,* $O \subseteq \mathbb{U}_{obj}$ *is a set of objects,*
- $OT = \{\pi_{type}(o) | o \in O\}$ *is a set of object types,*
- $\pi_{act} : E \rightarrow \mathbb{U}_{act}$ *maps each event to an activity,*
- $\pi_{obj} : E \rightarrow \mathcal{P}(\mathbb{U}_{obj}) \setminus \{\emptyset\}$ *maps each event to at least one object,*
- $\pi_{time} : E \rightarrow \mathbb{U}_{time}$ *maps each event to a timestamp*

4 Object-Centric Causal Nets

Object-centric causal nets describe the causal relations between the activities of a process and enriches this information with split and join constructs both for the workflow and for the object distribution perspective. The basis of an object-centric causal net is an object-centric dependency graph which captures the dependency relations between activities for multiple object types. The object-centric dependency graph is a directed graph with colored edges. The nodes represent activities, and the edges represent causal dependencies for the object type represented by the color.

Definition 2 (Object-Centric Dependency Graph). *An object-centric dependency graph is a tuple* $OCDG = (A_A, A_{\triangleright}, A_{\square}, OT, D)$ *where:*

- $A = A_A \cup A_{\triangleright} \cup A_{\square}$ *is the set of activities with visible activities* $A_A \subseteq \mathbb{U}_{act}$*, type-specific start activities* $A_{\triangleright} = \{\triangleright_{ot} \mid ot \in OT\}$*, and type-specific end activities* $A_{\square} = \{\square_{ot} \mid ot \in OT\}$*, such that* $A_{\triangleright} \cap \mathbb{U}_{act} = \emptyset$ *and* $A_{\square} \cap \mathbb{U}_{act} = \emptyset$*,*

- $OT \subseteq \mathbb{U}_{ot}$ *is a finite set of object types;*
- $D \subseteq (A_A \cup A_{\triangleright}) \times OT \times (A_A \cup A_{\square})$ *is the type-specific dependency relation, depicting the causal dependencies between two activities;*

such that all activities in the type-specific sub-graph $(A_{ot}, \{ot\}, D_{ot})$ *with edges* $D_{ot} = \{(a, ot', a') \in D \mid ot' = ot\} \subseteq D$ *and nodes* $A_{ot} = \{a \mid (a, ot, a') \in D_{ot} \vee (a', ot, a) \in D_{ot}\} \subseteq A$ *are on a path from one start activity* $\triangleright_{ot} \in A_{\triangleright}$ *to the corresponding end activity* $\square_{ot} \in A_{\square}$.

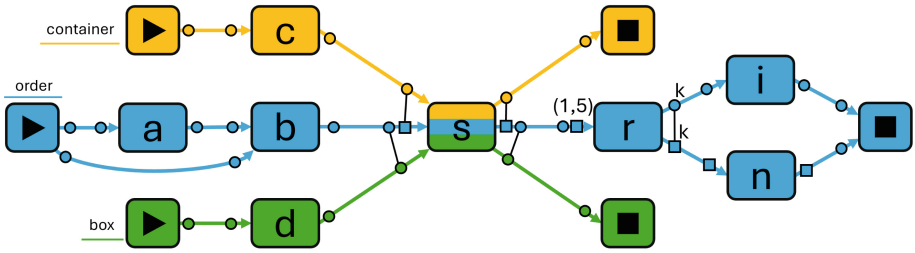


Fig. 2. Object-centric causal net for the running example that describes the example process correctly. Note that all markers are assumed to have unique keys except marked otherwise (here two markers have the key k)

Figure 2 shows the object-centric causal net for the running example, which has an object-centric dependency graph as a basis. There, for example, one can see that for the *order* activity r ($receive feedback$) is causally dependent from activity s ($send$). The object-centric dependency graph describes which activities are causally related to each other, but it does not describe three essential types of workflow information. (1) The workflow splits and joins: The logical connections between possible preceding or succeeding activities. For example, whether all causal successors must always follow or only one of them can follow. (2) The cardinality of involved objects: So whether single or multiple objects participate in the activity, and if multiple participate, how many exactly. (3) The distribution of objects of the same type over multiple succeeding activities: So whether objects of the same type that participated together in an event share the same workflow afterward or are split up. To model these workflow perspectives, we enhance the object-centric dependency graph with groups of markers.

An individual marker is connected to an activity and an object type and has a minimal and maximal cardinality and a key.

Definition 3 (Marker). *Given an object-centric dependency graph* $OCDG = (A_A, A_{\triangleright}, A_{\square}, OT, D)$*, a marker is a tuple* $m = (a, ot, c, k) = (a, ot, (c_{min}, c_{max}), k)$ *where:*

- $a \in A \cup A_{\triangleright} \cup A_{\square}$ *is an activity;*
- $ot \in OT$ *is an object type;*
- $c_{min}, c_{max} \in \mathbb{N}_{>0}$ *are cardinality constrains with* $c_{min} \leq c_{max}$*;*

- $k \in \mathbb{U}_{key}$ *is a key from the universe of keys* $\mathbb{U}_{key}$.

The set of potential markers for OCDG is given by $M_{OCDG} = \{(a, ot, c, k) \in A \times OT \times (\mathbb{N}_{>0} \times \mathbb{N}_{>0}) \times \mathbb{U}_{keys}\}$.

Markers are represented as dots (capacity of 1) and squares (multiple objects) and can be annotated with a specific capacity range like the marker in front of r (*receive*) in Fig. 2. It models that at most five orders are received together. Only non unique keys are shown, like the key k for the two markers after r.

An object-centric causal net consists of an object-centric dependency graph annotated with input and output groups of markers for each activity.

Definition 4 (Object-Centric Causal Net). *An object-centric causal net is a tuple* $OCCN = (OCDG, I, O) = ((A_A, A_\triangleright, A_\square, OT, D), I, O)$ *where:*

- $OCDG = (A_A, A_\triangleright, A_\square, OT, D)$ *is an object-centric dependency graph;*
- $I \in A \rightarrow \mathcal{P}(\mathcal{P}(M_{OCDG}))$ *defines the set of possible groups of input markers per activity; and*
- $O \in A \rightarrow \mathcal{P}(\mathcal{P}(M_{OCDG}))$ *defines the set of possible groups of output markers per activity; such that*
- $\forall a \in A : \forall ms \in I(a) : \forall (a', ot, c, k) \in ms : (a', ot, a) \in D$ *(an input marker of an activity must stem from causally preceding activities);*
- $\forall a \in A : \forall ms \in O(a) : \forall (a', ot, c, k) \in ms : (a, ot, a') \in D$ *(an output marker of an activity must only connect to causally succeeding activities);*
- $\forall a \in A_\triangleright : I(a) = \{\emptyset\}$ *(no input marker groups for start activities);*
- $\forall a \in A_\square : O(a) = \{\emptyset\}$ *(no output marker groups for end activities); and*
- $\forall a, a' \in A, ot \in OT : |\{(a', ot, c, k) \in I(a)\}| \leq 1 \wedge |\{(a', ot, c, k) \in O(a)\}| \leq 1$ *(at most one marker per arc in input and output marker groups);*

Figure 2 shows the object-centric causal net for the running example. Activity s has two input marker groups. One models that one container can be sent with multiple orders: $\{(c, container, (1, 1), k_i), (b, order, (1, *), k_j)\} \in I(s)$. It is visualized as the connected yellow dot and blue square in front of s. The other input marker group of s models that a box is sent with one order: $\{(d, box, (1, 1), k_l), (b, order, (1, 1), k_m)\} \in I(s)$.

The semantics of an object-centric causal net are defined by a state notion, and bindings (representing events) that can change the state. The state is defined as a multiset of obligations. An obligation like $(a, o1, b)$ describes that order $o1$ had performed activity a and thus has to perform activity b in the future.

Definition 5 (State of an OCCN). *Given an object-centric causal net* $OCCN = ((A_A, A_\triangleright, A_\square, OT, D), I, O)$*, the state space of* $OCCN$ *is* $S_{OCCN} = \mathbb{B}(A \times \mathbb{U}_{obj} \times A)$ *such that* $s \in S_{OCCN}$ *is a state, i.e., a multiset of pending obligations between causally related activities:* $\forall (a, ot, a') \in S_{OCCN} : (a, ot, a') \in D$.

For example, the state $s1 = [(a, o1, b), (d, b1, s)]$ describes that b must happen for order $o1$ (because a happened before) and s must happen for box

$b1$ (because d happened before). Bindings can change states. A binding consists of an activity a, the consumed obligations C and the produced obligations P. We use the shorthand notation $a \xrightarrow[P]{C}$ for bindings. For example, $s1 = [(b, o1, s), (d, b1, s)]\ s \xrightarrow[(o1,r),(b1,\square_{box})]{(b,o1),(d,b1)} [(s, o1, r), (s, b1, \square_{box})] = s2$ notes that a binding for activity s transformed the state $s1$ by consuming one obligation for order $o1$ (from b) and one for box $b1$ (from d) and creating new obligations for both objects, resulting in state $s2$. This describes the state change by an event with activity s and objects $o1$ and $b1$. Bindings must match the object-centric causal net. All consumed (and produced) obligations must be mapped to one input (output) marker group. The object assigned to a marker must match its type, and the number of objects assigned per marker must be within its cardinality constraints. Markers with the same key can never share assigned objects. This models an object workflow split. For non-starting and non-ending activities, the objects used in the input and output binding must be the same.

Definition 6 (Binding). *Given an object-centric causal net* $OCCN = ((A_A, A_\triangleright, A_\square, T, D), I, O)$*, the tuple* $b = (a, C, P) \in B = A \times \mathcal{P}(A \times \mathbb{U}_{obj}) \times \mathcal{P}(\mathbb{U}_{obj})$ *is a binding iff there exist marker mappings* $\phi_C \in A \times \mathbb{U}_{obj} \to M_{OCDG}$*,* $\phi_P \in A \times \mathbb{U}_{obj} \to M_{OCDG}$ *such that:*

- $\exists mg \in I(a) : range(\phi_C) = mg$ *(one input marker group used)*
- $\forall (a', o) \in C : \exists c', c'' \in \mathbb{N}_{>0}, k \in \mathbb{U}_{keys} : \phi_C(a', o) = (a', \pi_{type}(o), (c', c''), k)$ *(mapping matches in activity and object type)*
- $\forall (a', ot, c, k) \in M_{OCDG} : |\{o \in \mathbb{U}_{obj} | \phi_C(a', o) = (a', to, c, k)\}| \in c$ *(cardinality within contains)*
- $\forall (a', ot', c', k), (a'', ot'', c'', k) \in M_{OCDG} : \{o \in \mathbb{U}_{obj} | \phi_C(a', o) = (a', ot', c', k) \wedge \phi_C(a'', o) = (a'', ot'', c'', k)\} = \emptyset$ *(exclusive choice of objects for same keys) The conditions above must also hold for* ϕ_P *in an analog way.*
- $a \in A_A \Rightarrow \{o | \exists a' \in A : (a', o) \in C\} = \{o | \exists a' \in A : (o, a') \in P\}$ *(for non starting or ending activities the objects in* C *and* P *must be the same)*

A binding sequence $\sigma \in B^*$ *is a sequence of activity bindings.* $\langle\rangle$ *denotes the empty binding sequence. Function* $\psi(\sigma)$ *with* $\psi \in B^* \to S_{OCCN}$ *defines the state of an* $OCCN$ *after executing a binding sequence* σ*.* ψ *is defined inductively:* $\psi(\langle\rangle) = []$ *and*

$$\psi(\sigma \oplus (a, C, P)) = \left(\psi(\sigma) \setminus \biguplus_{(a',o) \in C} (a', o, a)\right) \uplus \left(\biguplus_{(o,a') \in P} (a, o, a')\right)$$

The language of an object-centric causal net is given by its valid binding sequences. A binding sequence is valid if, when applied to the empty state, it results in the empty state and only consumes existing obligations. Since each binding relates to an event with a certain activity and related objects, each valid bind sequence relates to a valid execution of the process according to the model.

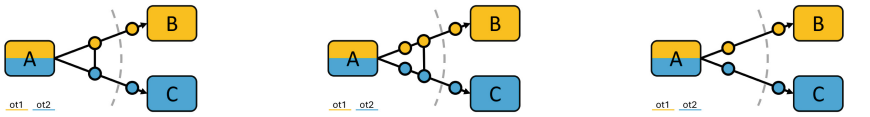
Definition 7 (Valid Binding Sequence). *Given an object-centric causal net* $OCCN = ((A_A, A_\triangleright, A_\square, T, D), I, O)$*, the binding sequence* $\sigma = \langle (a_1, C_1, P_1), (a_2, C_2, P_2), \ldots, (a_n, C_n, P_n) \rangle \in B^*$ *is valid iff*

- $\psi(\sigma) = []$; *and*
- *for any non-empty prefix* $\sigma' = \langle (a_1, C_1, P_1), \ldots, (a_l, C_l, P_l) \rangle$ $(1 \leq l \leq n)$: $\biguplus_{(a',o) \in C_l} (a', o, a) \leq \psi(\sigma'')$ *with* $\sigma'' = \langle (a_1, C_1, P_1), \ldots, (a_{l-1}, C_{l-1}, P_{l-1}) \rangle$.

$V(OCCN)$ *is the set of all valid sequences of* $OCCN$.

We semantically restrict the process space of $OCCN$ to $V(OCCN)$. Any invalid sequence is therefore not part of the process described by the model.

The syntax and semantics of object-centric causal nets allow precise modeling of the workflow, the object cardinality, and the object distribution. The workflow modeling is based on the same *AND, OR, and XOR-splits* that traditional causal nets use to describe processes. However, they can involve multiple objects and object types. Figure 3 shows the workflow splits for two object types.



(a) activity-**AND**-split (b) activity-**OR**-split (c) activity-**XOR**-split

Fig. 3. The workflow pattern splits object-centric causal nets can model. The dashed line only serves to separate the input- and output markers visually. More details on workflow splits and joins are described for traditional causal nets [20].

The concrete object cardinality for the activities is modeled by the (c_{min}, c_{max}) values of the markers. Figure 4 shows the cardinality relation between activities that can be modeled. For simplicity, we often visualize only whether the capacity is exactly one (with a circle) or multiple (0 or more).

(a) one-to-one (b) one-to-many (c) many-to-one (d) many-to-many

Fig. 4. Relation between the occurrences of A to the occurrences of B depending on whether the markers between A and B have a capacity of 1 object (represented as a circle) or multiple objects (represented as a square).

Object-centric causal net can precisely describe the object distribution to succeeding activities (see the splits in Fig. 5) and, similarly, the merging of objects from preceding activities. The keys of markers describe this. If makers share the same key, any valid binding is required to distribute the objects involved in the event over the markers with the same key without duplicates. This models that objects of this type have to make an exclusive choice if the makers have the same

keys. For markers with unique keys, the objects are free to be assigned to one or multiple markers. Lets assume for example an event $A_{(o1,o2)}$ with objects $o1$ and $o2$ happened. The model in Fig. 5b would require each object $o1$ and $o2$ to do either B or C because the markers have the same key k. The model in Fig. 5a also allows any of the objects to do both B and C.

(a) free object distribution (b) object choice

Fig. 5. If markers share keys, objects need to exclusively choose one of the markers. Markers without indicated keys are assumed to have unique keys. Objects do not need to choose between them but can be assigned to multiple markers.

Combining all these modeling capabilities, one can correctly model the desired behavior of the running example process and overcome the representational biases of object-centric Petri nets pointed out in Sect. 1. The object-centric causal net in Fig. 2 models the running example. First, the skipping behavior for activity a (*checkorder*) is represented without the need for silent transitions in the object-centric causal net. Second, the input and output marker groups for s (*send*) clearly show that *containers* and *boxes* are never used together and that whenever *boxes* are involved, only one *order* is sent, whereas multiple *orders* can be sent with *containers*. Third, the markers with the same key in the output marker group for r (*receivefeedback*) ensure that only one order is investigated and the rest are not investigated. This shows that object-centric causal nets can overcome the mentioned representational biases of object-centric Petri nets.

5 Mining Object-Centric Causal Nets

In this section, we describe the object-centric causal net miner. The algorithm's structure is based on the frequent heuristic miner [26] used to discover traditional causal nets [20]. As defined in Definition 4, an object-centric causal net consists of an object-centric dependency graph enriched with groups of markers. Therefore, the discovery algorithm for object-centric causal nets first discovers an object-centric dependency graph and then discovers marker groups for that graph. Figure 6 shows an overview of the algorithm. The described algorithm is implemented in Python and publicly accessible on GitHub[1].

[1] https://github.com/LukasLiss/OCCN-Miner.

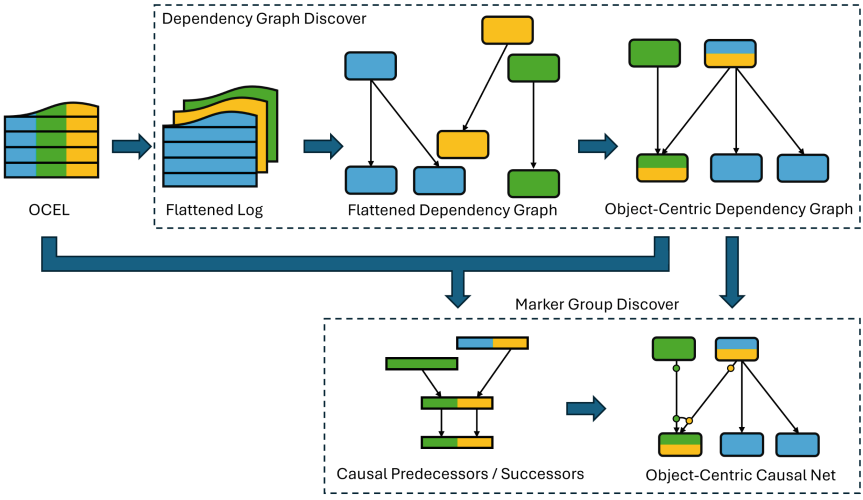


Fig. 6. Overview of the algorithm to mine object-centric causal nets.

5.1 Object-Centric Dependency Graph Discovery

The first step is discovering the object-centric dependency graph. As an input, this step uses an object-centric event log. Since the dependencies are object type-specific, they can be discovered independently for each object type. So, the object-centric event log is then flattened, as described by van der Aalst and Berti [21]. This step creates one traditional event log per object type containing the traces of the objects of that type. For the example log in Table 1, the flattened container log consists of one trace for container $c1$: $\langle c, s \rangle$.

Given the traditional event log, the traditional dependency graph discovery algorithm [26] is used to discover the dependency graphs for each object type. This comes with three parameters DT, L_1T, and L_2T that are described in [26]. Additionally, each object type-specific dependency graph is extended with a start (and end) node connected to all start (and end) activities. The object-centric graph is constructed by merging all nodes with the same activity label. This creates the object-centric dependency graph which is the foundation of Fig. 2.

5.2 Marker Group Discovery

The second part of the object-centric causal net discovery is the discovery of the input and output marker groups. The input is the object-centric event log and the discovered object-centric dependency graph. This part consists of two steps. First, each event's causal predecessors and successor from the object-centric event log are determined. Second, input and output marker groups are discovered by detecting frequent patterns in the set of causal predecessors and

successors of events of a certain activity. The marker groups are added to the graph. In the following, we will look at the *s (send)* events from the example log in Table 1 and how input marker groups are created for them. In the tool, users can filter for frequent marker groups to focus on the most dominant behavior.

Causal Predecessor and Successors Heuristics. The first step uses the object-centric event log and the object-centric dependency graph to mine for each event in the log, its causal predecessors, and causal successors. The causal predecessors are those events that causally resulted in the execution of the given event. The causal successors are those events that causally result from the execution of the given event. Since the event log does not clearly state which events causally infer each other, a heuristic must be used to select suitable candidates. The heuristic uses both the dependency relations from the dependency graph and the temporal ordering present in the event log. This problem also exists for traditional causal net mining, and two approaches, which can be generalized to the object-centric setting, have been proposed to solve the issue.

One heuristic is window-based: Based on a user-defined window size, one limits the search for causal predecessors and successors to events within the window size of the given event. All dependent events within the window that share objects are considered to be causal predecessors or successors. For the *s (send)* event *e5* in Table 1, all *c*, *b*, and *d* events with a shared object within the window are predecessors because *s* causally depends on them based on the dependency graph. So with a large window size, the events *e2*, *e3*, and *e4* are causal predecessors, but with a window size of 1, only the closest event *e4* remains.

Another heuristic is based on event ordering: This heuristic assumes that a given event is a causal result of those events before the given event that have a dependency to the given event, and there is no other event between them that they also have a dependency to. So, a given event has all those events as its causal predecessors that have the given event as the closest following dependent event. For the causal successors this means that a given event has all those events as its causal successors that have the given event as the closest event before them that has a dependency relation to them. This does not require user input and allows finding predecessors and successors outside of the window size. For example, for the *send* event *e5*, the events *e2*, *e3*, and *e4* are causal predecessors because they share objects, and no events that also share the same object are in between them. So *e1* is not because *e2* happens in between and also contains object *o1*.

The published implementation uses the second heuristic of the ordering of events. But regardless of the used heuristic, the result of this step is a mapping of events $e \in \mathbb{U}_{events}$ to their causal successors $CS(e) \subset \mathbb{U}_{event}$ and predecessors $CP(e) \subset \mathbb{U}_{event}$. For the following, we assume $CS(e)$ and $CP(e)$ given.

Marker Group Algorithm. The second step creates input and output marker groups for each event and its causal predecessors and successors. Listing 1.1 shows the algorithm to create an input marker group for a given event and its predecessors. It creates a marker for each activity in the causal predecessors

and selects the cardinality based on the number of objects of a certain type involved in the causal predecessor and the given event. Finally, markers that should have similar keys are identified by checking whether two markers from different activities but with the same object type distribute objects among each other. The objects are distributed between multiple markers of the same type if they do not share any object, then they receive the same key to show that the objects are distributed here.

For example, the *s (send)* event *e5* has predecessors with the following activities: $[b, b, c]$. This creates a marker group that connects b and c. The marker for s has a maximum capacity of 2. If more *s (send)* events have predecessors that match this marker group, no new groups are added. But for the *s (send)* event *e11* with successors with activities $[b, d]$, a new marker group is needed because d is not part of the existing one. So a new marker group that connects b and d is added. Both markers have a capacity of precisely one. These two marker groups are the input marker groups for the *send* activity in the example in Fig. 2.

Algorithm 1.1: Marker computation for a given event.

```
input: Event e, Causal Predecessors CP(e)
output: Input marker group mg (contributed by event e)
begin
  mg ← []
  foreach a ∈ {π_act(e')|e' ∈ CP(e)}
    foreach t ∈ {π_type(e')|e' ∈ CP(e) ∧ π_act(e') = a}
      c_min ← min{|π_obj(e')_↓t ∩ π_obj(e)_↓t| |e' ∈ CP(e) ∧ π_act(e) = a}
      c_max ← max{|π_obj(e')_↓t ∩ π_obj(e)_↓t| |e' ∈ CP(e) ∧ π_act(e) = a}
      mg ← mg ∪ (a, t, (c_min, c_max), k) with unused k ∈ U_keys
      end
  end

  # find partition of keys for object distribution
  while ∃(a', t, (c'_min, c'_max), k') ∈ mg ∧ (a'', t, (c''_min, c''_max), k'') ∈ mg
   such that a' ≠ a'' ∧ k' ≠ k'' ∧ π_obj(e')_↓t ∩ π_obj(e'')_↓t ≠ ∅
   for e' ∈ CP(e) ∧ π_act(e') ∧ π_act(e') = a'
   and e'' ∈ CP(e) ∧ π_act(e') ∧ π_act(e'') = a''
      # unify keys
      (a', t, (c'_min, c'_max), k') ← (a', t, (c'_min, c'_max), k'')
  end
  return mg
end
```

One computes all the input marker groups using the algorithm in Listing 1.1. Similarly, one computes the outgoing marker groups using causal successors. Multiple input marker groups can then be merged when they have the same structure (including the same distribution of keys but abstracting from concrete keys since they are unique). The capacity minimum and maximum are also adjusted based on which marker groups are merged. But markers with a capacity of 1 are not merged with those with a higher capacity to distinguish whether multiple or single objects can be used. The final object-centric causal net, is created by adding all marker groups to the object-centric dependency graph.

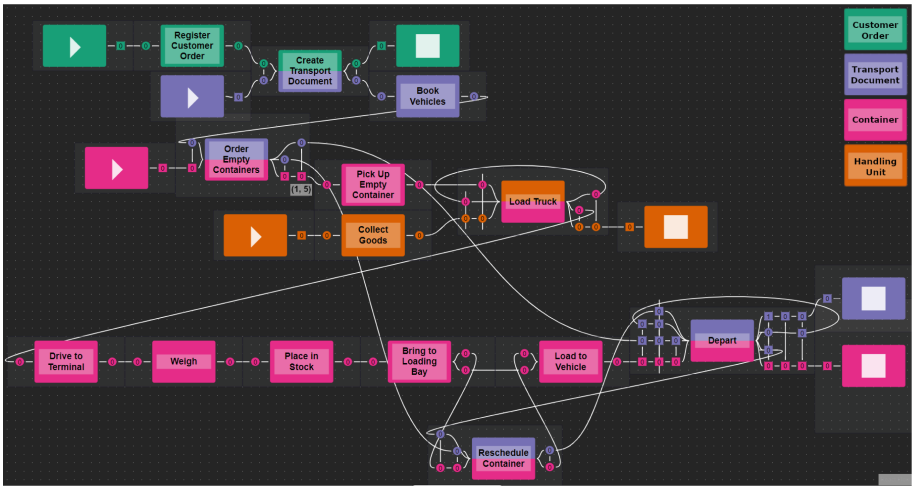


Fig. 7. The object-centric causal net mined, filtered, and visualized with our implementation on the *Logistics* log. The legend shows the object types.

6 Evaluation

In this section, we present the qualitative and quantitative evaluation results. We used 10 publicly available OCEL 2.0 [2] object-centric event logs.

Qualitative Evaluation. To evaluate object-centric causal nets and their discovery algorithm qualitatively, we mined object-centric causal nets for the 10 logs and compared them with the discovered object-centric Petri net discovered from the same log. In the following, we focus on the filtered *Logistics* log results, but the other results are publicly accessible in the GitHub repository. We filtered the *Logistics* log for four object types *Customer Orders*, *Transport Document*, *Containers*, and *Handling Unit*. The resulting object-centric causal net can be seen in Fig. 7 and the object-centric Petri net in Fig. 8.

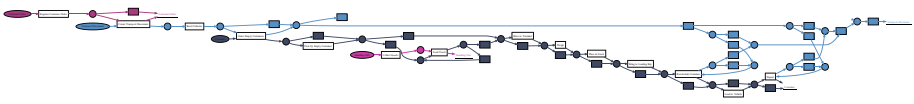


Fig. 8. Object-Centric Petri net [21] for the *Logistics* log mined with pm4py [3]. For better readability see original at https://www.ocel-standard.org/event-logs/simulations/logistics/.

The overall workflow behaviour is very similar between the causal net and the Petri net. Both show 14 activities. The Petri net contains additionally 25 places and 26 silent transitions. The causal net contains 8 additional start and end activites. Both describe similar workflow patterns like the sequence of *Create Transport Document* and *Book Vehicle*, or the loop of the *Load Truck* event for the *Container object type*. This shows that the discovered behavior is consistent with already existing approaches but has a different representational bias. For example, after *Order Empty Containers*, the *Transport Document* can either go to *Depart* or directly or to *Reschedule Container* first. For this optional skipping, one needs an artificial silent transition in a Petri net.

The object-centric causal net describes behavior aspects not modeled in the Petri net. For example, it shows concrete cardinalities for the output marker of *Order Empty Containers* which models that at maximum 5 containers are ordered together matching the log. This information can not be represented in an object-centric Petri net. The object-centric causal net in Fig. 7 also shows an object XOR split, which would not be explicitly representable in an object-centric Petri net. There can be multiple *Transport Documents* per *Depart* activity. Each can either end its life cycle or be part of another *Depart* event. The qualitative evaluation shows that the behavior discovered using the presented algorithm is consistent with existing approaches in the workflow dimension and can discover additional aspects due to different representational biases.

Quantitative Evaluation. This quantitative evaluation investigates the runtime of our implementation by using 10 publicly accessible event logs. The experiments were performed on a 12th Gen Intel i7-12650H processor with 2.3 GHz and 16 GB of RAM. For the dependency graph discovery parameters [26] we used the following settings: $DT = 0.5$, $L_1T = 0.5$, and $L_2T = 0.5$. The computation times were measured three times and averaged to reduce the impact of noise. The evaluation shows that the algorithm, with an average overall run time of 125 s on the used event logs, is usable on consumer hardware. Figure 9 shows the relation between the overall runtime and the number of events and the number of objects. As the correlation coefficient is 0.92 for the number of events and 0.34

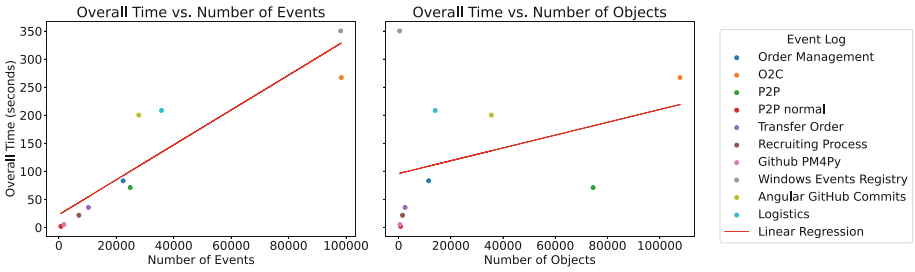


Fig. 9. The overall runtime of the object-centric causal net discovery using our implementation for variate of object-centric event logs.

for the number of objects, the runtime is mainly influenced by the number of events in the event log. This is reasonable since the algorithm must discover the potential marker groups for each event in the event log. The linear regression line indicates a linear increase in the runtime with the size of the object-centric event log. Since, there are no algorithms to discover object-centric causal nets yet, we can not compare the performance with other implementations. Instead, we publish the given implementation as a baseline for future improvements.

7 Conclusion

In this paper, we introduce object-centric causal nets and make four contributions to the field of object-centric process discovery. First, we defined the syntax and semantics of object-centric causal nets, a model that can overcome some of the representational biases of current models like object-centric Petri nets. Second, we presented an algorithm to discover them from object-centric event data. This two-phase approach creates an object-centric dependency graph first by merging traditional dependency graphs. In the second phase, this graph is populated with input and output marker groups, resulting in an object-centric causal net. This phase uses a heuristic to identify which events with dependent activities causally proceed and succeed an event. Then the causal predecessors (successors) result in input (output) marker groups. Third, we published our algorithm implementation together with a visualization tool for object-centric causal nets that allows the user to filter for the most frequent input and output marker groups. Finally, we performed a qualitative and quantitative evaluation, showing that object-centric causal nets can represent behavior that can not be discovered and modeled using object-centric Petri nets.

This work motivates future research on object-centric causal nets for conformance checking [5], enhancement [7] and prediction [17].

Acknowledgments. The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint project Bridging AI (grant no. 16DHBKI023).

References

1. Augusto, A., et al.: Automated discovery of process models from event logs: review and benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019)
2. Berti, A., et al.: OCEL (object-centric event log) 2.0 specification. CoRR, abs/2403.01975 (2024)
3. Berti, A., van Zelst, S.J., Schuster, D.: Pm4py: a process mining library for python. Softw. Impacts **17**, 100556 (2023)
4. Böhmann, T., Leimeister, J.M., Möslein, K.M.: Service systems engineering - a field for future information systems research. Bus. Inf. Syst. Eng. **6**(2), 73–79 (2014)

5. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer, Cham (2018)
6. Christfort, A.K.F., Rivkin, A., Fahland, D., Hildebrandt, T.T., Slaats, T.: Discovery of object-centric declarative models. In: 2024 6th International Conference on Process Mining (ICPM), pp. 121–128. IEEE (2024)
7. de Leoni, M.: Foundations of process enhancement. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. LNBIP, vol. 448, pp. 243–273. Springer, Cham (2022)
8. Fahland, D.: Process mining over multiple behavioral dimensions with event knowledge graphs. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. LNBIP, vol. 448, pp. 274–319. Springer, Cham (2022)
9. Fettke, P., Reisig, W.: Systems mining with heraklit: the next step. In: Di Ciccio, C., Dijkman, R., del Río Ortega, A., Rinderle-Ma, S. (eds.) Business Process Management Forum, pp. 89–104. Springer, Cham (2022)
10. Gianola, A., Montali, M., Winkler, S.: Object-centric conformance alignments with synchronization. In: Guizzardi, G., Santoro, F.M., Mouratidis, H., Soffer, P. (eds.) CAiSE 2024. LNCS, vol. 14663, pp. 3–19. Springer, Cham (2024)
11. Künzle, V., Reichert, M.: Philharmonicflows: towards a framework for object-aware process management. J. Softw. Maintenance Res. Pract. **23**(4), 205–244 (2011)
12. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Cham (2013)
13. Liss, L., Adams, J.N., van der Aalst, W.M.P.: Totem: temporal object type model for object-centric process mining. In: Marrella, A., Resinas, M., Jans, M., Rosemann, M. (eds.) BPM 2024 . LNBIP, vol. 526, pp. 107–123. Springer, Cham (2024)
14. Park, G., Adams, J.N., van der Aalst, W.M.P.: Conformance checking and performance analysis using object-centric directly-follows graphs. In: Marrella, A., Resinas, M., Jans, M., Rosemann, M. (eds.) BPM 2024 Forum. LNBIP, vol. 526, pp. 179–196. Springer, Cham (2024)
15. Reisig, W.: Petri Nets: An Introduction. EATCS Monographs on Theoretical Computer Science, vol. 4. Springer, Cham (1985)
16. Snoeck, M.: Enterprise Information Systems Engineering - The MERODE Approach. The Enterprise Engineering Series. Springer, Cham (2014)
17. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30
18. van der Aalst, W.: Process-aware information systems: lessons to be learned from process mining. Trans. Petri Nets Other Model. Concurr. **2**, 1–26 (2009)
19. Aalst, W.: Object-centric process mining: dealing with divergence and convergence in event data. In: Ölveczky, P.C., Salaün, G. (eds.) SEFM 2019. LNCS, vol. 11724, pp. 3–25. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30446-1_1
20. van der Aalst, W., Adriansyah, A., van Dongen, B.: Causal nets: a modeling language tailored towards process discovery. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 28–42. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23217-6_3
21. van der Aalst, W., Berti, A.: Discovering object-centric petri nets. Fundam. Informaticae **175**(1–4), 1–40 (2020)
22. van Detten, J.N., Schumacher, P., Leemans, S.J.J.: Discovering compact, live and identifier-sound object-centric process models. In: 6th International Conference on

Process Mining, ICPM 2024, Kgs. Lyngby, Denmark, 14–18 October 2024, pp. 113–120. IEEE (2024)
23. van Dongen, B.F., de Medeiros, A.K.A., Wen, L.: Process mining: overview and outlook of petri net discovery algorithms. Trans. Petri Nets Other Model. Concurr. **2**, 225–242 (2009)
24. Veit, D., et al.: Business models - an information systems research agenda. Bus. Inf. Syst. Eng. **6**(1), 45–53 (2014)
25. Weber, B., Reichert, M., Rinderle-Ma, S., Wild, W.: Providing integrated life cycle support in process-aware information systems. Int. J. Cooper. Inf. Syst. **18**(1), 115–165 (2009)
26. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible heuristics miner (FHM). In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, Part of the IEEE Symposium Series on Computational Intelligence 2011, 11–15 April 2011, Paris, France, pp. 310–317. IEEE (2011)

System Architecture and Privacy

Building FAIR-Compliant Lakehouses with FLAMI

João P. C. Castro[1,2(✉)], Gabriel F. X. Vasconcelos[1], Genoveva Vargas-Solar[3], and Cristina D. Aguiar[1]

[1] Department of Computer Science, University of São Paulo (USP), São Carlos, Brazil
jpcarvalhocastro@ufmg.br, gabriel.vasconcelos@usp.br, cdac@icmc.usp.br
[2] Information Technology Board, Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil
[3] CNRS, Univ. Lyon, INSA Lyon, UCBL, LIRIS, UMR5205, Lyon, France
genoveva.vargas-solar@cnrs.fr

Abstract. This paper introduces FLAMI, a software reference architecture for building big data-sharing repositories that adhere to the FAIR (Findability, Accessibility, Interoperability, and Reusability) principles. Inspired by the data lakehouse concept, FLAMI can extract, load, and transform large volumes of data and metadata from heterogeneous data providers into a unified storage solution. It integrates with existing infrastructures of external repositories, allowing data stored outside the lakehouse to be fetched and processed within its infrastructure. FLAMI also introduces a set of guidelines for its implementation. We validate our proposal through a case study that instantiates FLAMI to the context of a real-world seismology dataset, exploiting analytical queries that encompass the needs of geoscientists. We also employ a framework to assess FLAMI's FAIR compliance, achieving 100% in its conceptual version and over 73% in the seismology instantiation.

Keywords: FAIR Principles · Open Science · SRA · Data Lakehouse

1 Introduction

Open Science has emerged to encourage researchers to share their findings, removing barriers to global scientific collaboration. These findings can include the data generated by their research, the used methodologies, and the resulting scientific papers [28]. Building data-sharing repositories is key to advancing Open Science, promoting transparency and increasing the impact of scientific discoveries. Given the inherent heterogeneity of content, data providers, processing mechanisms, and protocols to manage data and metadata, these repositories must ensure findability, accessibility, interoperability, and reusability.

The FAIR (Findable, Accessible, Interoperable, Reusable) principles offer a standardized framework to achieve these guarantees [38]. They ensure data

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 113–129, 2025.
https://doi.org/10.1007/978-3-031-94571-7_7

and metadata are findable through unique identifiers and rich descriptions and accessible via standard protocols that comply with legal and ethical constraints. Additionally, the FAIR Principles enable interoperability by utilizing widely recognized languages and vocabularies and allow reusability through detailed data provenance and adherence to established standards and licensing practices.

Achieving the standardization outlined by FAIR is challenging due to the gap between their high-level definitions and the practical development of a data-sharing repository. Another challenge refers to the different profiles of data providers, which can impose a diverse range of requirements. Some can prioritize data ownership, sharing only metadata while maintaining control over their data in an external repository [8]. Others value simplicity, relinquishing control of data and metadata to third-party repositories for simplified management [9].

Adopting a Software Reference Architecture (SRA) can address these challenges, guiding data engineers in developing a FAIR-compliant repository. An SRA is a high-level architecture that is context- and technology-independent [30]. It allows development teams to build systems by outlining the application's structure, components, interactions, and data flow, ensuring consistency and maintainability across projects [26]. An SRA can also support the interaction of different stakeholder profiles with the architecture, provided this is specified during its conception. We believe that it can bridge the existing silo between the FAIR Principles and their implementation in a big data-sharing repository.

Despite the advantages of a FAIR-compliant SRA, solutions available in the literature are scarce [10]. Some approaches lack design considerations for addressing FAIR Principles or fail to achieve full compliance. Others are context-specific and therefore unable to adhere to the concept of an SRA. There are also solutions that do not provide support for data analytics. Additionally, many existing SRAs are not designed to accommodate diverse data provider profiles. Instead, they tend to prioritize either data ownership or simplicity, often neglecting a more holistic approach to FAIR compliance.

This paper introduces FLAMI (FAIR Lakehouse Architecture for Multiple Infrastructures), an SRA designed to enable the development of data-sharing repositories aligned with the FAIR Principles. Rather than proposing an ad-hoc solution for a specific data-sharing repository, we present a general architecture inspired by the lakehouse paradigm [34]. This allows FLAMI to support data analytics across diverse repository contexts, leveraging components such as data lakes [33] and data warehouses [21]. FLAMI also integrates with existing infrastructures of external repositories, allowing data stored outside the lakehouse to be fetched and processed within its infrastructure. This enables FLAMI to support both data ownership and simplicity, addressing the requirements of different profiles of data providers. The contributions of our work include:

- FLAMI, a FAIR-compliant SRA with a modular design tailored for constructing data-sharing repositories across diverse applications and domains.
- A set of comprehensive guidelines for leveraging FLAMI to develop FAIR-compliant data-sharing repositories.

- Validation of FLAMI by building a FAIR-compliant data-sharing repository for a real-world seismology dataset in collaboration with geosciences experts. This includes designing and implementing a series of queries to evaluate the repository's ability to retrieve data and metadata.
- An assessment protocol to evaluate the FAIR compliance of FLAMI and its geosciences instantiation, using the FAIREST assessment framework [15].

The remainder of this paper is organized as follows. Section 2 proposes FLAMI, assesses its FAIR compliance, and details its instantiation guidelines. Section 3 details the case study that instantiates FLAMI to the context of a real-world seismology dataset. Section 4 reviews related work. Finally, Sect. 5 concludes the paper and discusses future work.

2 The FLAMI Architecture

This section introduces FLAMI, an SRA for building FAIR-compliant data repositories inspired by the data lakehouse concept. Figure 1 depicts its layers, with solid borders for components, dashed borders for processes, arrows for data flow, and padlocks to indicate end-to-end encryption [32].

2.1 Users and Interaction

Data Providers and Consumers. Data providers can follow one of two profiles: (i) those who favor simplicity and performance, storing their source data directly in the lakehouse infrastructure; and (ii) those who prioritize data ownership and flexibility to implement their own security policies, storing their source data in their own external infrastructure. Metadata from both cases is extracted by FLAMI, which integrates and stores it in the lakehouse. Data consumers query and retrieve data and metadata through the Repository Interaction Layer, with different access levels depending on their role.

Repository Interaction Layer. Acts as the gateway for data providers and consumers to interact with FLAMI. It includes interfaces for uploading and retrieving data and metadata according to user roles and permissions. It provides a search engine that should index the metadata and can provide data and metadata retrieval functions to ensure findability.

2.2 Data and Metadata Storage

External Storage Layer. Hosts external repositories containing research data and metadata. These repositories can be heterogeneous and geographically distributed. They must support real-time metadata streaming to the Repository Interaction Layer and provide access interfaces for the Data Retrieval Layer.

Repository Storage Layer. Consists of several components for storing data and metadata. The Data Lake stores two copies of research data objects: anonymized

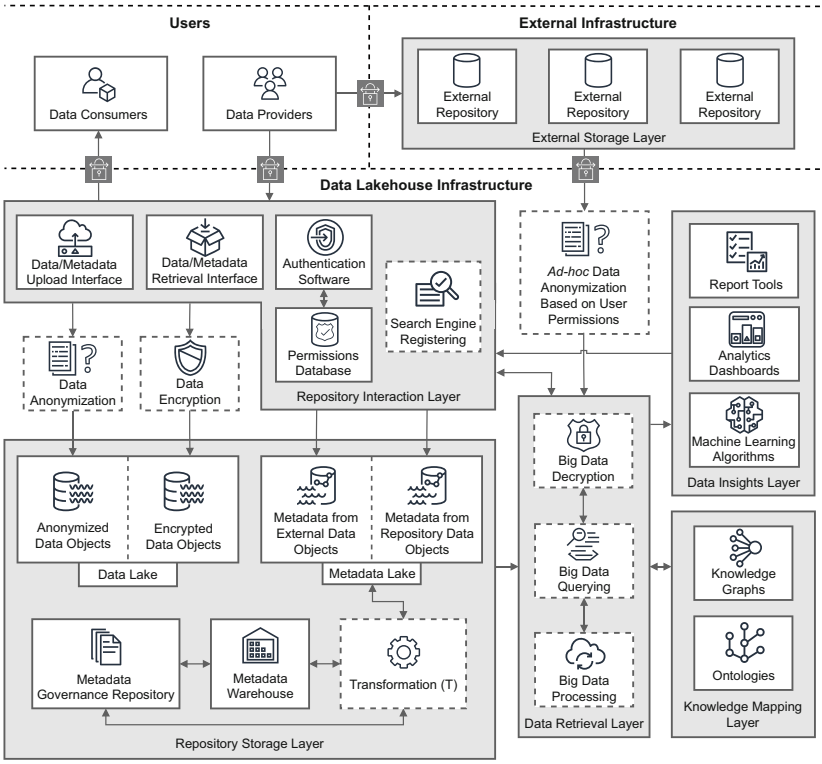


Fig. 1. The FLAMI architecture.

and encrypted. We perform this duplication to improve query execution, allowing data consumers to instantly access anonymized or non-anonymized data based on their roles. Furthermore, the Metadata Lake stores raw metadata for real-time querying, with separate files for metadata from external data objects in the External Storage Layer and repository data objects in the Data Lake. This metadata is gradually transformed, cleaned, and integrated before being stored in the Metadata Warehouse. Intermediate levels of these transformations can be stored in the Metadata Lake, following the premise of a lakehouse. The Metadata Warehouse, implemented using the Generic Metadata Warehouse Model [8], stores the final transformed metadata. This model is based on a star schema [21], consisting of multiple dimensions to store descriptive metadata along with a fact table to link these dimensions and store their associated quantitative measures. By leveraging its integrated, subject-oriented, historical, and non-volatile attributes [19], the Metadata Warehouse keeps metadata alive even in the absence of its associated data objects while also enhancing batch metadata querying. Finally, a Metadata Governance Repository tracks data provenance and meta-metadata.

2.3 Data Search and Exploitation

Data Retrieval Layer. Handles requests for metadata and source data. Metadata is retrieved directly from the Metadata Warehouse and Metadata Lake. For source data, the layer checks if the data is stored internally or externally. Then, based on the consumer's permissions, it fetches internal data from the Data Lake in its anonymized or non-anonymized version, or applies ad-hoc anonymization for external data. The results can either be delivered directly to the data consumer or processed in the Data Insights Layer. During query execution, a request can be made to the Knowledge Mapping Layer to translate data consumer's requests to fit the data model of the Repository Storage Layer.

Knowledge Mapping Layer. This layer contains associations between generic data models and the data models implemented in the other layers, which can be created using ontologies [35] or knowledge graphs [16]. They can enable the recovery of data and metadata based on standards well-known to the data consumers of specific research fields. They can also allow the investigation of different relationships between the data and metadata, feeding the Data Insights and the Data Retrieval Layers with novel content to improve their analytics capabilities.

Data Insights Layer. Enables big data analytics within FLAMI by leveraging content from the Repository Storage, External Storage, and Knowledge Mapping Layers. It provides descriptive, predictive, and prescriptive analyses [25] through public and private dashboards, report tools for repository management, and machine learning algorithms for advanced insights. Access to these insights is controlled via the Repository Interaction Layer to ensure security.

2.4 FAIR Compliance Evaluation of FLAMI

We use the FAIREST assessment framework [15] to evaluate the FAIR compliance of FLAMI. FAIREST provides categorical and quantitative metrics for this assessment. Details on how to comply to each metric and to calculate compliance scores are available in [15]. Here, we discuss how FLAMI addresses the metrics in the Findability, Accessibility, Interoperability, and Reusability categories.

Findability. The Repository Storage Layer satisfies FCS1, FCS2, FCS3, and FCS4 because it includes a Metadata Warehouse that stores globally unique *identifiers* for each data object, such as Digital Object Identifiers (DOIs) or their combination with local unique identifiers. These identifiers can also be derived from *external identifiers* (FCS3) in the External Storage Layer and, together with other metadata, can point to the *exact location of the source data* (FCS4) in the Repository Storage and External Storage Layers. Furthermore, the Data Retrieval Layer enables *metadata* (FCA1) and *content* (FCA2) search, supporting *advanced search features* (FCA3) through complex queries and analytics for the Data Insights Layer.

Accessibility. The Repository Interaction Layer supports customizable *GUIs* (ACL1) and *APIs* (ACL2) for data and metadata retrieval. The Repository

Storage Layer guarantees *long-term preservation* (ACA1), ensuring the recovery of data and metadata even if deleted from their sources. *Availability* (ACA2) is rated "high" since the FLAMI's unified lakehouse infrastructure is deployed on big data environments (e.g., cloud providers) by definition. This setup enforces the replication of each component, prioritizing availability and avoiding the impact of crashes and service interruptions. Finally, the Repository Interaction Layer provides all specified levels of *access control* (AC3). For instance, the public can access an *open* dashboard; a repository manager can use *closed* self-service analytics, and data providers can interact *on request* with data objects.

Interoperability. Metrics IC1 and IC2 are addressed by the Repository Storage and Knowledge Mapping Layers, which use *standard data formats* and can map metadata to *recognized models* (e.g., Dublin Core, MARC 21). *Persistent identifiers* (IC3) and *custom metadata* (IC4) are achieved when FLAMI assigns identifiers to each data object and provides the extensibility of the Metadata Warehouse generic model, respectively. *Linking of metadata and content* (IC5) occurs via ontologies and knowledge graphs in the Knowledge Mapping Layer and relationships of stored data and metadata in the Repository Storage Layer. Further, the Repository Interaction Layer guarantees *upload* (ICI1) and *download* (ICI2) of metadata and content, as well as a *custom submission process* (ICI3) by supporting GUIs and APIs for data access with different roles and permissions. Finally, the External Storage Layer meets *data federation* (ICI4) by integrating external repositories.

Reusability. License support for content depositing (RCD1) and *access* (RCA1) are rated "high" because the Metadata Warehouse allows assigning different licenses to each data object and FLAMI guarantees the availability of licensing attributes, including license descriptions and links. Furthermore, the Data Retrieval and Knowledge Mapping Layers fulfil *content support* (RCS1) by processing and mapping data and metadata in various formats.

Discussion. Based on the aforedescribed metrics, we applied the FAIREST's formulae[1] to calculate FLAMI's FAIR compliance. From these formulae, we obtained a score for each category of the FAIR Principles as an absolute value. Then, to calculate the compliance percentage for each category, we divided the obtained score by the category's maximum possible score. The compliance percentage regarding all categories of the FAIR Principles was determined by averaging the percentages from each category. This approach ensures that each category contributes equally to the final compliance percentage, regardless of the number of metrics they contain. The result demonstrated that FLAMI is 100% FAIR-compliant according to the FAIREST framework.

2.5 Instantiation Guidelines

We propose a set of instantiation guidelines for implementing FAIR-compliant data-sharing repositories using FLAMI. These guidelines allow a data engineer

[1] https://doi.org/10.5281/zenodo.5282929.

to decide which components and functions to adopt and the desired level of FAIR compliance to tackle. Each guideline concerns a specific FLAMI layer, detailing the implementation of its components and their interactions with other components and layers.

Guideline 1: Implementing the External Storage Layer. Data providers can implement external repositories using technologies tailored to their data characteristics (model, format) and required data management guarantees (SQL, NoSQL, distributed file systems). They can be deployed in different target architectures on the cloud, the fog, or the edge. These repositories must interact with the Repository Interaction Layer to upload their metadata. The interfaces and interactions will vary depending on the type of storage support (look-up commands for distributed file systems, SQL queries or ad-hoc retrieval expressions for NoSQL systems). Metadata extraction can imply processing data using parallel solutions like Spark or stream processing solutions like Kafka, depending on the data production rate. The external repositories should also integrate with the Data Retrieval Layer to enable data retrieval and querying.

Guideline 2: Implementing the Repository Interaction Layer. The first step in this layer is to develop the data/metadata upload and retrieval interfaces, which can be provided as APIs or GUIs. Authentication mechanisms should be integrated, using either a permissions database for credential management or SSL/TLS protocols for certificate-based validation [13]. To ensure findability, metadata should be made searchable through domain-specific engines or generic platforms like Google Dataset Search. Data engineers are responsible for implementing consolidation strategies to notify incoming metadata and data to the Repository Storage Layer. Metadata should remain in its raw format while data is duplicated, with one copy anonymized and the other encrypted.

Guideline 3: Implementing the Repository Storage Layer. To choose the most suitable technologies to implement the storage components in this layer, a data engineer should understand the component's purpose and the expected characteristics of the data and metadata it stores. The metadata lake stores metadata in its raw format. The same is true for data stored in the data lake. Thus, technologies that do not require a predefined schema are more suitable, such as NoSQL databases (e.g., MongoDB) or distributed file storages (e.g., HDFS), with the latter being preferred for handling larger datasets. FLAMI requires all metadata in the metadata lake to be transformed to a metadata warehouse generic model [8] to ensure FAIR compliance. These transformations can be implemented using Pandas for smaller datasets and Spark for larger ones, with intermediate results stored in the metadata lake. To implement the metadata warehouse, data engineers can employ relational databases like PostgreSQL for smaller datasets or distributed file storages (e.g., HDFS) for larger datasets. The same technologies can also be used to implement the metadata governance repository. The data engineer needs to ensure that the chosen technologies can integrate with the data processing framework used in the Data Retrieval Layer to enable data and metadata querying.

Guideline 4: Implementing the Data Retrieval Layer. This layer should implement querying pipelines to locate data and metadata in the External Storage Layer or the Repository Storage Layer, leveraging big data frameworks like Spark if necessary. For data querying requests to the External Storage Layer, end-to-end encryption [32] must be implemented to ensure secure communication with external repositories. Additionally, ad-hoc data anonymization [3], tailored to data consumer permissions, should be applied to sensitive data. Data stored in the Repository Storage Layer, having already been anonymized, can either be queried in its anonymized form or decrypted to reveal sensitive information, contingent upon the data consumer's access permissions. Alternatively, encryption-based analytics [18] can enable querying of encrypted data without decryption, ensuring security and functionality.

Guideline 5: Implementing the Knowledge Mapping Layer. This layer comprises components that utilize tailored models, such as ontologies or knowledge graphs, to represent knowledge and semantic data. The choice of model depends on the nature of the requests being processed. Ontology-based models often leverage reasoning mechanisms implemented with frameworks like RDF triples, enabling inferencing and semantic enrichment. Alternatively, property graphs can be employed for knowledge representation, facilitating querying through graph query languages such as GQL or Cypher. This data can then be stored in a graph database such as GraphDB or Virtuoso, ensuring integration with the Data Retrieval and Data Insights layers.

Guideline 6: Implementing the Data Insights Layer. This layer consists of components that enable the creation of dashboards for generating reports and analytics, providing valuable insights into processed data and metadata. It interacts with the Data Retrieval Layer to extract the necessary data and metadata for analysis. Tools such as Microsoft Power BI, Tableau, and Metabase are well-suited for implementing analytics dashboards. For machine learning, libraries such as TensorFlow and Spark MLlib can be employed. Customized reports tools can be generated using query engines like SparkSQL. Data engineers should guarantee that the chosen technologies are compatible with the Data Retrieval Layer.

3 Case Study: Building an Earth Sciences Lakehouse

We validate FLAMI and our instantiation guidelines by building an Earth Sciences lakehouse using a real-world seismology dataset. This dataset comprises Seismic Analysis Code (SAC) files, which document seismic activity recorded across stations in the northeast region of Brazil. Seismic measurements are captured by seismographs and validated and processed by various agents before inclusion in an official seismic bulletin [14]. For this case study, we focus on the initial collection phase of the SAC files, before any processing by data collectors or analysts. The dataset consists of 13,900 files, totaling 167GB of data. Each seismic event generates three SAC files corresponding to the three measurement axes (north-south, east-west, and up-down). Each SAC file contains an array of

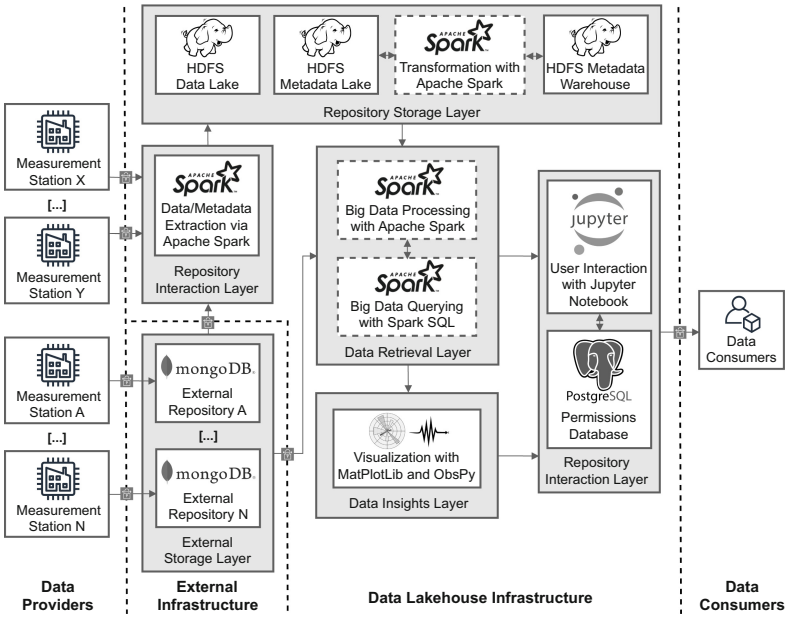


Fig. 2. Instantiation of FLAMI to the context of earth sciences.

integer values representing the seismic measurements, accompanied by metadata such as station, channel, and number of data points.

3.1 FLAMI's Instantiation

The instantiation of FLAMI is illustrated in Fig. 2, with only the components necessary to fulfill the requirements of the case study being deployed. In this setup, each measurement station functions as an independent data provider. Specifically, nine stations utilize external repositories to store their SAC files, while ten rely on the Lakehouse Infrastructure for storage. As the instantiation focuses on managing data during the initial phase of the seismic bulletin generation process (i.e., before any user interaction), the primary data consumers are research personnel involved in this stage, such as data collectors and analysts.

We implemented each repository within the External Storage Layer as a separate MongoDB instance, adhering to Guideline 1, and used Spark to enable measurement stations to upload SAC files and their associated metadata to the lakehouse, following Guideline 2. Additionally, the Repository Interaction Layer includes a Jupyter Notebook interface, allowing data consumers to issue queries and visualize results. Authentication mechanisms to verify identities and permissions are managed through a PostgreSQL database.

For the Repository Storage Layer, we employed the Hadoop Distributed File System (HDFS) to implement the Data Lake, Metadata Lake, and Metadata

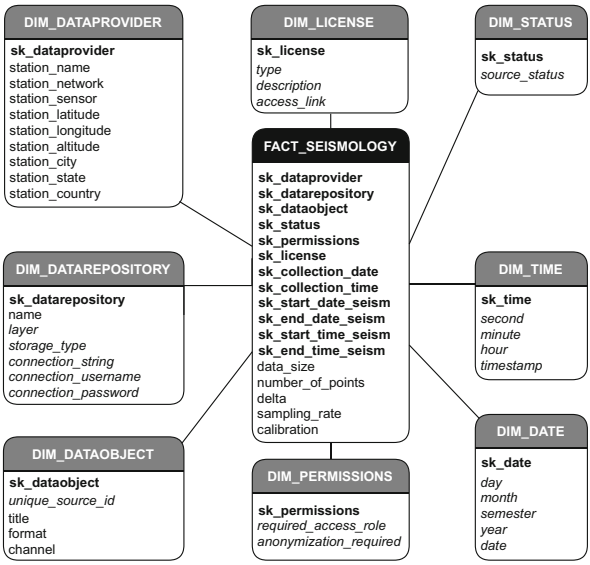


Fig. 3. The Metadata Warehouse Star Schema.

Warehouse, in line with Guideline 3. The Data Lake stores raw SAC files data providers upload without external repositories. As these files do not contain sensitive information, no anonymization or encryption measures were required. The Metadata Lake stores metadata in the Parquet file format, with Spark handling the processing and transformation of this metadata before storing it in the Metadata Warehouse.

Metadata Warehouse. Figure 3 depicts the star schema of the implemented metadata warehouse. Each entry of the fact table FACT_SEISMOLOGY represents storing a SAC file, considering the data size, number of points in the measurement, delta, sampling rate, and calibration. The dimension tables are: (i) DIM_DATAPROVIDER, holding station information (e.g., location); (ii) DIM_DATAREPOSITORY, containing information about each repository (e.g., connection details); (iii) DIM_DATAOBJECT, detailing the SAC file (e.g., unique source ID); (iv) DIM_DATE and DIM_TIME, storing both measurement and file insertion timestamps; (v) DIM_PERMISSIONS, defining access permissions and anonymization; (vi) DIM_LICENSE, holding licensing details; and (vii) DIM_STATUS, indicating whether the SAC file still exists in its source. Each table is implemented as a Parquet file. Bold attributes represent primary keys, while italic attributes indicate metadata essential to comply with the FAIR Principles. A data engineer can adjust the schema to add or remove content, provided these essential attributes are maintained.

In the Data Retrieval Layer, components are designed to implement algorithms for retrieving, processing, and analyzing data and metadata in alignment with Guidelines 4 and 6. For this case study, these algorithms were developed

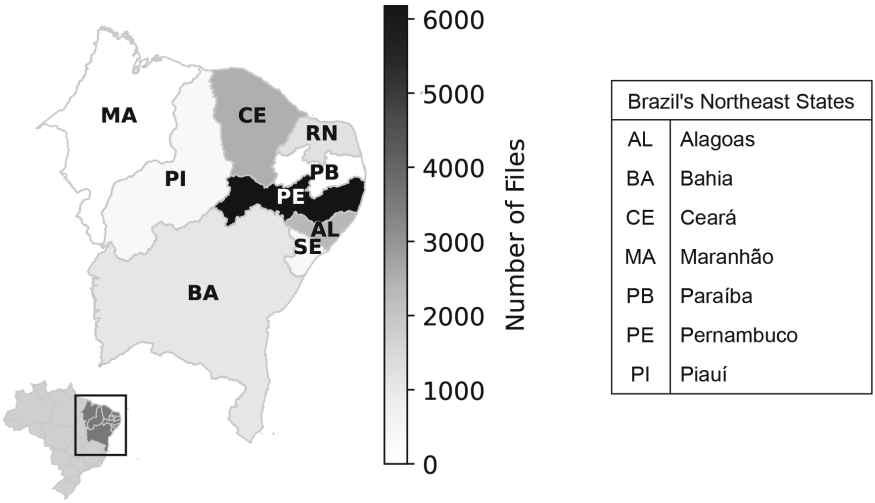


Fig. 4. Query Q1 (total number of files generated by the data providers of each state).

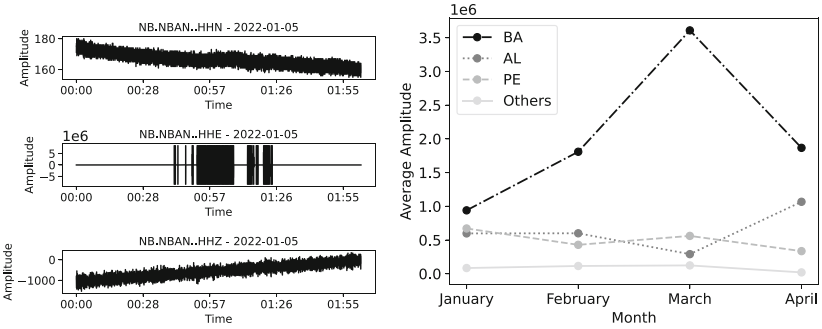
using Spark's Resilient Distributed Datasets (RDDs) and SparkSQL. In the Data Insights Layer, components provide visualization features using the ObsPy and MatPlotLib libraries, enabling a comprehensive and detailed exploration of both data and metadata. Lastly, the Knowledge Mapping Layer, corresponding to Guideline 5, was not instantiated, as the limited scope of data consumer requests did not warrant its implementation.

3.2 Analytical Queries

We designed a series of analytical queries to illustrate typical data consumer requests and demonstrate the interaction between the layers of FLAMI. These queries were inspired by those commonly posed by geoscientists during the initial stages of preparing a seismic event bulletin [14]. The queries were implemented in Spark, using both its RDD interface and SparkSQL.

Query Q1. Which regions produce the most data? This query retrieves *metadata* from the Metadata Warehouse in the Repository Storage Layer to identify the number of files generated by data providers in each state. By analyzing this information, the query estimates the distribution of seismic data across different regions. The results provide insights into which areas contribute with more data, denoting either higher seismic activity or increased monitoring efforts. They can also assist in resource allocation among stations.

Figure 4 depicts a choropleth map containing Q1's results. We show the map of Brazil on the bottom left and highlight the states that comprise its northeast region. The visualization reveals that Pernambuco (PE) generates the highest number of files (6,180), followed by Ceará (CE) with 2,508 files and Alagoas (AL) with 2,338 files. In contrast, Bahia (BA) and Piauí (PI) generate fewer files, with



(a) Q2's results. **(b)** Q3's results.

Fig. 5. Queries Q2 (all channels of the seismic reading with the highest amplitude in AL) and Q3 (average amplitude in each state per month). Amplitude is obtained from the source data, while states/months are extracted from the Metadata Warehouse.

1,050 and 312, respectively. This distribution indicates higher seismic activity or more intensive monitoring efforts in PE and CE. Additionally, it highlights underrepresented areas, such as PI, which may warrant increased monitoring efforts and resource allocation.

Query Q2. Which are the channels of the reading of the highest amplitude in AL? This query is implemented through the following actions: (i) query the metadata warehouse to retrieve connection details to a data provider; (ii) retrieve SAC files from the data providers located in Alagoas (AL); (iii) calculate their amplitude; (iii) identify the SAC file with the highest amplitude; and (iv) retrieve other SAC files for the remaining channels. FLAMI performed these actions in parallel using Spark's RDD interface and SparkSQL.

The results of Query Q2 are visualized in Fig. 5a using ObsPy. Geoscientists can observe that the E channel exhibits a significantly higher amplitude (in the millions) compared to the Z channel (in the thousands) and the N channel (in the hundreds). These insights enable them to evaluate whether the readings correspond to a seismic event. This query aids in analyzing specific seismic measurements, helping determine if the data represents an earthquake or another type of seismic activity.

Query Q3. What is the average amplitude in each state over the months of 2022? This query involves retrieving metadata from the Repository Storage Layer and source data from data providers, following these steps: (i) querying the metadata warehouse to obtain connection details for the data sources, along with the state and month for each entry in the fact table; (ii) retrieve the corresponding SAC files from the data providers; (iii) calculate the average amplitude of the seismic measurements, aggregated by month and state. FLAMI performed these steps in parallel leveraging both SparkSQL and Spark's RDD interface.

The results of Query Q3 are illustrated in Fig. 5b. Bahia (BA) had the highest average amplitude in January (939,956) and peaked in March (3,608,805),

suggesting significant seismic activity. Alagoas (AL) exhibited moderate values, starting at 597,287 in January and increasing to 1,066,790 in April. Pernambuco (PE) showed a decline in amplitude, ranging from 670,916 in January to 335,157 in April. Other states consistently reported lower amplitudes, beginning at 84,269 in January and decreasing to 19,624 in April. These findings underscore BA's prominent seismic activity and reveal varying trends across states, offering valuable insights for prioritizing regional monitoring efforts. The results also help geoscientists to identify seasonal patterns or regional variations in seismic activity, as well as to detect potential outliers in the analyzed period.

3.3 FAIR Compliance Evaluation: Seismology Instantiation

The FAIR compliance assessment conducted in Sect. 2.4 showed that FLAMI is 100% FAIR-compliant according to the FAIREST framework [15]. However, since not all layers and components are instantiated in our case study, some framework metrics cannot be fully satisfied. Details on how to comply to each metric and to calculate compliance scores are available in [15]. Here, we detail which metrics FLAMI's seismology instantiation is unable to fulfill, as follows.

Regarding Findability, FAIREST clearly specifies that metric FCS2 can only be fulfilled if the repository supports the generation of a DOI for each research artifact. In our instantiation, this metric is not satisfied due to the high cost associated with assigning a DOI to every SAC file. As a result, Findability compliance is reduced to 71.4%. In Accessibility, metric ACA3 can only be marked as "open" if it makes research artifacts open by default. Similarly, it can only assume the "on-request" value if the repository offers an on-request feature to the public. Therefore, ACA3 assumes the "closed" value in our instantiation because data access is limited to analysts in the early stages of seismic bulletin generation. This results in a 66.7% score for Accessibility.

Regarding Interoperability, metric IC1 demands that standard metadata formats, such as Dublin Core and MARC 21, are interpretable by the repository. Metric IC5 states that semantic links between data and metadata must be created and published. These metrics are not satisfied in our case study due to our decision of not instantiating the Knowledge Mapping Layer. Thus, the Interoperability compliance is reduced to 55.6%. Finally, the instantiation achieves 100% in the Reusability category, boosting its overall FAIR compliance to 73.4%.

Our assessment shows that full FAIR compliance may not always be achievable, since the compliance score is influenced by the repository's context-specific constraints. For instance, certain contexts may require restricting data consumer profiles, leading to a closed repository and directly impacting the accessibility assessment. Hence, data engineers must understand these context-specific constraints when determining the target FAIR compliance score for a repository.

4 Related Work

We categorize the studies reviewed in the literature into groups based on a timeline that reflects the evolution from adopting SRAs to achieving alignment

with the FAIR Principles. The studies were selected from our systematic review detailed in [10], supplemented with recent and relevant solutions to ensure comprehensive coverage of the state of the art.

General Purpose Big Data SRAs (Group 1). Historically, studies in Group 1 do not consider the FAIR Principles in their design. These studies distinguish themselves because of the introduction of the traditional data warehouse architecture [11]. This architecture has been augmented with parallel and distributed data processing frameworks [6] and data lakes [33] to deal with the volume, variety, and velocity properties of big data [12]. Furthermore, the data lakehouse architecture [34] stores data in object storage solutions using standard formats, integrated with a metadata layer to manage transactions and versioning [1]. Some solutions provide specialized layers for data extraction, loading, and transformation (ELT) aimed at big data analytics, such as Kappa [23], Lambda [22], Liquid [17], Solid [27], Bolster [29], and NeoMycelia [2]. FLAMI is inspired by the data lakehouse concept and incorporates components for ELT.

Implementation of FAIR-Driven Repositories (Group 2). Studies in Group 2 propose solutions incorporating the FAIR Principles to integrate data and metadata in specific contexts, such as earthquakes [31], biomedical research [4], oncology [20], COVID-19 [5], scientific metadata [36], digital twin integration with research data management [24], and astronomy [39]. However, most of these studies lack a detailed explanation of how their solutions fulfill each FAIR principle. Only a few, such as those in [20,31], provide partial insights into their FAIR compliance. In contrast, our work offers a clear justification for why FLAMI is a FAIR-compliant SRA. Additionally, we assess its compliance using the FAIREST framework [15]. A key distinction of our approach is the introduction of detailed guidelines for using FLAMI to develop FAIR-compliant repositories, a discussion absent from studies in Group 2.

FAIR-Compliant Big Data SRAs (Group 3). FAIR-compliant SRAs have gained attention in the literature. Among them, BigFAIR [8] and CloudFAIR [9] are FAIR-compliant SRAs we previously proposed. BigFAIR employs a hybrid approach, combining a local source data infrastructure with a centralized metadata repository. This design prioritizes data ownership, enhances flexibility, and ensures compliance with data protection regulations. Building on BigFAIR, CloudFAIR integrates data and metadata management into a unified cloud infrastructure, improving performance and simplifying interactions for data providers.

These experiences with BigFAIR and CloudFAIR inspired the development of FLAMI, designed to support different profiles of data providers. FLAMI achieves flexibility and data ownership by integrating with existing external infrastructures while ensuring performance and simplicity through a unified lakehouse infrastructure for source data storage. By bridging these two data provider profiles, FLAMI simplifies the specification of FAIR-compliant repositories.

Another FAIR-compliant SRA in the literature is GADDS [37], a cloud-based platform focusing on metadata quality control via blockchain-based cryp-

tographic storage and decentralized validation. However, unlike our work, BigFAIR, CloudFAIR, and GADDS do not assess FAIR compliance using a formal assessment framework or provide instantiation guidelines. Additionally, GADDS compromises findability, limiting its overall utility. Furthermore, GADDS compromises findability due to its adoption of a blockchain network.

5 Conclusions and Future Work

This paper presented FLAMI, a software reference architecture designed for developing FAIR-compliant data-sharing repositories. Inspired by the data lakehouse concept, FLAMI integrates with external infrastructures, enabling the querying and processing of internal and external data while accommodating diverse data provider profiles. We also proposed six guidelines to enable the implementation of FAIR-compliant repositories using FLAMI. These guidelines were demonstrated through a geosciences case study, where FLAMI was instantiated for the context of seismology. Finally, we validated FLAMI's compliance to the FAIR Principles, achieving 100% in its conceptual version and over 73% in the seismology instantiation.

We are currently working on describing the systematic process used to design FLAMI. This process includes validating correctness, support for instantiation, and utility by interviewing real-world data engineers and geoscientists. We are also conducting performance evaluations to compare FLAMI with other solutions in the literature. Another future work is to develop interface definitions for the Repository Interaction Layer, enabling a higher degree of standardization across different FLAMI instantiations. We also plan to perform other case studies whose scope allows us to instantiate enough components of FLAMI to reach 100% FAIR compliance, such as those in the Knowledge Mapping Layer. For instance, we plan to explore real-world jellyfish monitoring data [7], showcasing FLAMI's capacity to deal with data and metadata variety. Finally, we will explore different contexts that allow the instantiation of other components and processes, such as machine learning algorithms and a metadata governance repository.

Acknowledgments. This work was supported by the São Paulo Research Foundation (FAPESP), the Brazilian Federal Research Agency (CNPq), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brasil (CAPES), Finance Code 001. The work is partially associated with the project LETITIA funded by the Lyon Federation of Informatics 2023–2025 and the project RESUME funded by Edital 36/2023 Pró-Defesa V. João P. C. Castro was also supported by the Institutional Development Program for UFMG Employees (PRODIS).

References

1. Armbrust, M., Ghodsi, A., Xin, R., Zaharia, M.: Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In: Proceedings of CIDR, pp. 28–35 (2021)

2. Ataei, P., Litchfield, A.: NeoMycelia: a software reference architecture for big data systems. In: Proceedings of APSEC, pp. 452–462 (2021)
3. Bazai, S.U., Jang-Jaccard, J., Alavizadeh, H.: Scalable, high-performance, and generalized subtree data anonymization approach for Apache Spark. Electronics **10**(5) (2021)
4. Begoli, E., Goethert, I., Knight, K.: A lakehouse architecture for the management and analysis of heterogeneous data for biomedical research and mega-biobanks. In: IEEE International Conference on Big Data (Big Data), pp. 4643–4651 (2021)
5. Borges, V., de Oliveira, N.Q., Rodrigues, H., Campos, M., Lopes, G.: A platform to generate FAIR data for COVID-19 clinical research in Brazil. In: Proceedings of ICEIS, pp. 218–225 (2022)
6. Brito, J.J., Mosqueiro, T., Ciferri, R.R., Ciferri, C.: Faster cloud star joins with reduced disk spill and network communication. Procedia Comput. Sci. **80**, 74–85 (2016)
7. Carneiro, A., Nascimento, L., Noernberg, M., Hara, C., Pozo, A.: Social media image classification for jellyfish monitoring. Aquat. Ecol. **58**(1), 3–15 (2024)
8. Castro, J.P.C., Romero, L.M., Carniel, A.C., Aguiar, C.D.: FAIR principles and big data: a software reference architecture for open science. In: Proceedings of ICEIS, pp. 27–38 (2022)
9. Castro, J.P.C., Romero, L.M., Carniel, A.C., Aguiar, C.D.: Open Science in the cloud: the CloudFAIR architecture for FAIR-compliant repositories. In: Proceedings of ADBIS, pp. 56–66 (2022)
10. Castro, J.P.C., Aguiar, C.D.: Big data architectures for FAIR-compliant repositories: a systematic review. In: Proceedings of SBBD (2023)
11. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. **26**(1), 65–74 (1997)
12. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Mob. Netw. Appl. **19**(2), 171–209 (2014)
13. Chien, H.Y.: Dynamic public key certificates with forward secrecy. Electronics **10**(16) (2021)
14. Costa, U.S., Espinosa-Oviedo, J.A., Musicante, M.A., Vargas-Solar, G., Zechinelli-Martini, J.L.: Using provenance in data analytics for seismology: challenges and directions. In: New Trends in Database and Information Systems. ADBIS 2022, pp. 311–322 (2022)
15. D'Aquin, M., Kirstein, F., Oliveira, D., Schimmler, S., Urbanek, S.: FAIREST: a framework for assessing research repositories. Data Intell. **5**(1), 202–241 (2023)
16. Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. In: Martin, M., Cuquet, M., Folmer, E. (eds.) Proceedings of Posters and Demos Track of SEMANTiCS. CEUR Workshop Proceedings, vol. 1695 (2016)
17. Fernandez, R.C., et al.: Liquid: unifying nearline and offline big data integration. In: Proceedings of CIDR (2015)
18. Gershinsky, G.: Efficient analytics on encrypted data. In: Proceedings of the 11th ACM International Systems and Storage Conference, p. 121 (2018)
19. Inmon, W.H.: Building the Data Warehouse. Wiley, New York (2005)
20. Jha, A.K., et al.: Implementation of big imaging data pipeline adhering to FAIR principles for federated machine learning in oncology. IEEE Trans. Radiat. Plasma Med. Sci. **6**(2), 207–213 (2022)
21. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. Wiley, New York (2011)

22. Kiran, M., Murphy, P., Monga, I., Dugan, J., Baveja, S.S.: Lambda architecture for cost-effective batch and speed big data processing. In: Proceedings of IEEE Big Data, pp. 2785–2792 (2015)
23. Kreps, J.: Questioning the Lambda architecture (2014). https://www.oreilly.com/radar/questioning-the-lambda-architecture/
24. Lehmann, J., Schorz, S., Rache, A., Häußermann, T., Rädle, M., Reichwald, J.: Establishing reliable research data management by integrating measurement devices utilizing intelligent digital twins. Sensors **23**(1), 468 (2023)
25. Lepenioti, K., Bousdekis, A., Apostolou, D., Mentzas, G.: Prescriptive analytics: literature review and research challenges. Int. J. Inf. Manag. **50**, 57–70 (2020)
26. Martinez-Fernandez, S., Medeiros Dos Santos, P.S., Ayala, C.P., Franch, X., Travassos, G.H.: Aggregating empirical evidence about the benefits and drawbacks of software reference architectures. In: 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 1–10 (2015)
27. Martínez-Prieto, M.A., Cuesta, C.E., Arias, M., Fernández, J.D.: The solid architecture for real-time management of big semantic data. Futur. Gener. Comput. Syst. **47**, 62–79 (2015)
28. Medeiros, C.B., et al.: IAP input into the UNESCO Open Science Recommendation (2020). https://www.interacademies.org/sites/default/files/2020-07/Open_Science_0.pdf
29. Nadal, S., et al.: A software reference architecture for semantic-aware big data systems. Inf. Softw. Technol. **90**, 75–92 (2017)
30. Nakagawa, E.Y., Antonino, P.O., Becker, M.: Reference architecture and product line architecture: a subtle but critical difference. In: Proceedings of ECSA, pp. 207–211 (2011)
31. Pana, G.T., Ivanoaica, T., Raportaru, M.C., Baran, V., Nicolin, A.: Towards the implementation of FAIR principles on an earthquake analysis platform. In: Proceedings of RoEduNet, pp. 1–4 (2021)
32. Puthal, D., Nepal, S., Ranjan, R., Chen, J.: A synchronized shared key generation method for maintaining end-to-end security of big data streams. In: Proceedings of HICSS, pp. 6011–6020 (2017)
33. Sawadogo, P., Darmont, J.: On data lake architectures and metadata management. J. Intell. Inf. Syst. **56**(1), 97–120 (2021)
34. Schneider, J., Gröger, C., Lutsch, A., Schwarz, H., Mitschang, B.: The lakehouse: state of the art on concepts and technologies. SN Comput. Sci. **5**(449), 1–39 (2024)
35. Staab, S., Studer, R., Schnurr, H.P., Sure, Y.: Knowledge processes and ontologies. IEEE Intell. Syst. **16**(1), 26–34 (2001)
36. Toulet, A., et al.: ISSA: generic pipeline, knowledge model and visualization tools to help scientists search and make sense of a scientific archive. In: Proceedings of ISWC, pp. 660–677 (2022)
37. Vazquez, P., Hirayama-Shoji, K., Novik, S., Krauss, S., Rayner, S.: Globally accessible distributed data sharing (GADDS): a decentralized FAIR platform to facilitate data sharing in the life sciences. Bioinformatics **38**, 3812–3817 (2022)
38. Wilkinson, M.D., et al.: The FAIR guiding principles for scientific data management and stewardship. Sci. Data **3**(1), 1–9 (2016)
39. Zonca, A., Wagner, R., Borrill, J.: Cheap and FAIR: a serverless research data repository for the next generation cosmic microwave background experiment. In: Practice and Experience in Advanced Research Computing 2024: Human Powered Computing, pp. 1–4 (2024)

Advanced System Integration: Analyzing OpenAPI Chunking for Retrieval-Augmented Generation

Robin D. Pesl[1(✉)], Jerin G. Mathew[2], Massimo Mecella[2], and Marco Aiello[1]

[1] University of Stuttgart, Stuttgart, Germany
{robin.pesl,marco.aiello}@iaas.uni-stuttgart.de
[2] Sapienza, Università di Roma, Rome, Italy
{mathew,mecella}@diag.uniroma1.it

Abstract. Integrating multiple (sub-)systems is essential to create advanced Information Systems (ISs). Difficulties mainly arise when integrating dynamic environments across the IS lifecycle, e.g., services not yet existent at design time. A traditional approach is a registry that provides the API documentation of the systems' endpoints. Large Language Models (LLMs) have shown to be capable of automatically creating system integrations (e.g., as service composition) based on this documentation but require concise input due to input token limitations, especially regarding comprehensive API descriptions. Currently, it is unknown how best to preprocess these API descriptions. Within this work, we (i) analyze the usage of Retrieval Augmented Generation (RAG) for endpoint discovery and the chunking, i.e., preprocessing, of state-of-practice OpenAPIs to reduce the input token length while preserving the most relevant information. To further reduce the input token length for the composition prompt and improve endpoint retrieval, we propose (ii) a Discovery Agent that only receives a summary of the most relevant endpoints and retrieves specification details on demand. We evaluate RAG for endpoint discovery using the RestBench benchmark, first, for the different chunking possibilities and parameters measuring the endpoint retrieval recall, precision, and F1 score. Then, we assess the Discovery Agent using the same test set. With our prototype, we demonstrate how to successfully employ RAG for endpoint discovery to reduce the token count. While revealing high values for recall, precision, and F1, further research is necessary to retrieve all requisite endpoints. Our experiments show that for preprocessing, LLM-based and format-specific approaches outperform naïve chunking methods. Relying on an agent further enhances these results as the agent splits the tasks into multiple fine granular subtasks, improving the overall RAG performance in the token count, precision, and F1 score.

Keywords: Retrieval augmented generation · Large language models · OpenAPI · Endpoint discovery · RestBench

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 130–148, 2025.
https://doi.org/10.1007/978-3-031-94571-7_8

1 Introduction

OpenAPI is the state-of-practice for describing interfaces for integrating Information Systems (ISs). It contains formal elements like paths and natural language constituents like descriptions. For integrating these systems automatically, automated service composition using Large Language Models (LLMs) has been proposed [34–36]. These approaches exploit the capabilities of LLMs to process formal and natural language input, combining them with the inherent nature of automated service composition of decoupling and independent lifecycle management. While prohibiting any manual modeling effort by relying on already broadly available OpenAPIs, the approaches face the challenge of limited input token length [36]. This bounds the quantity and extent of the input service description. Even for proprietary models with a large input token context, e.g., OpenAI's GPT4 with a context size of 128,000 tokens [29], an economic constraint emerges as these models are paid by input and output token count. Therefore, a smaller prompt length is beneficial to (i) insert further service documentation and (ii) reduce costs for proprietary models.

To address these challenges, Retrieval Augmented Generation (RAG) [22] has emerged as a promising resort. In this approach, the external information is collected in a database, typically structured as a set of documents or document chunks. The primary goal is retrieving only a small subset of the most relevant documents or document chunks, which is then inserted into the prompt [22]. How to optimally apply RAG for endpoint discovery in IS is open to investigation, leading to the following research questions:

RQ1. How best to preprocess, i.e., chunk, OpenAPIs for RAG endpoint discovery?
RQ2. Can LLM agents be employed to reduce token count further and improve retrieval performance?

To answer RQ1, we develop an *OpenAPI RAG* system that takes as input service descriptions. We apply different token-based and LLM-based chunking strategies to split the documentation and evaluate them based on retrieval quality. The token-based strategies process the document using a classical parser and then split the parts, e.g., endpoints, into equal-sized chunks. The LLM-based strategies let an LLM create a description, i.e., a summary or a question, for each endpoint and then use these descriptions for similarity matching. We employ mainstream open-source and proprietary embedding models for similarity matching, which can create an embedding vector for an input. The similarity between two inputs can then be determined by comparing their embedding vectors using, e.g., the cosine similarity. We evaluate the OpenAPI RAG and the different chunking strategies by relying on the already available RestBench benchmark [42] for LLM agents, measuring recall, precision, and F1 score for each chunking strategy. The benchmark consists of the OpenAPI descriptions of Spotify and TMDB and corresponding queries, each with a set of endpoints as the sample solution.

To address RQ2, we propose an LLM agent called *Discovery Agent*. As LLM agents allow the usage of external tools, we first investigate using one tool that simply inputs the results of the RAG to the prompt. Then, we experiment with using two tools: the first tool filters and enters the LLM endpoint summaries to the prompt using RAG, while the second tool allows the retrieval of the endpoint details on demand. We resort to the same RestBench benchmark for evaluation and measure recall, precision, F1 score, and additional token count. As the chunking strategy, we rely on the LLM-based summary strategy with OpenAI's `text-embedding-3-large` embedding model [30].

The remainder of the paper is structured as follows. First, we provide an overview of related works regarding service discovery and LLMs in Sect. 2. Then, we present how to use RAG for endpoint discovery and the OpenAPI chunking strategies in Sect. 3. We evaluate and discuss the RAG and the different chunking strategies in Sect. 4. Finally, we conclude with Sect. 5.

2 Related Work

Regarding endpoint discovery, we provide a brief overview of the essential concepts of the various service discovery approaches. Additionally, we provide relevant insights into LLMs and the novel approach of integrating LLMs with tools, known as LLM agents, and how they relate to our approach.

2.1 Service Discovery

Service discovery describes the process of revealing relevant services that satisfy given requirements. This includes the service capabilities, addresses, functionality, usually called endpoints, and descriptions [21]. An example would be a vehicle wanting to interact with roadside units modeled as a service to book a parking spot. The service discovery discloses the service to the vehicle and filters them for the parking spot booking.

The most common service discovery implementation is a service registry, which collects information about available services and offers search facilities. This service registry is usually backed by a component residing at the middleware or application level [21]. It is characterized by the syntax used to describe the services and their invocation and the expressive power of the available query language. The typical integration model is a pull model where service consumers search for the required services. Less is a push model as used in the UPnP protocol, where service providers regularly advertise their services [39].

In the early days of XML-based services, the infrastructure for service discovery was the Universal Description, Discovery, and Integration (UDDI) specification [11]. UDDI had a global incarnation called the UDDI Business Registry (UBR), intended to offer an Internet-wide repository of available web services and promoted by IBM, Microsoft, and SAP. Unfortunately, UBR never gained widespread adoption and was short-lived (2000–2006). Significant research in the early days focused on enhancing service discovery on UDDI, improving search

capabilities, and creating federated registries, e.g., [5,8,16]. Alternatively, WS-Discovery is a multicast protocol that finds web services on a local network.

Nowadays, OpenAPI is the de facto standard for describing services. While not offering a discovery protocol and mechanism, given its popularity, OpenAPI would also benefit from discovery [41]. So, additional infrastructure for discovery has been proposed, such as centralized repositories (SwaggerHub or Apiary), service registry integration (Consul, Eureka), API Gateways (Kong, Apigee), or Kubernetes annotations (Ambassador).

Populating registries of services requires effort from service providers, which often hinders the success of such approaches, especially if the service provider is expected to provide extensive additional information beyond the service endpoints. This additional effort has often been the reason for the failure of some of these technologies, most notably UBR. Approaches confined to specific applications, domains, or enterprises have been more successful, e.g., Eureka. Developed by Netflix as part of its microservices architecture [43], Eureka helps clients find service instances described by host IP, port, health indicator URL, and home page. Developers can add additional data to the registry for extra use cases.

While classical incarnations like UDDI used to be comprehensive, they required extensive modeling, e.g., as semantic annotations. Hence, our approach relies on already broadly available state-of-practice OpenAPI specification and their integration with RAG.

2.2 Information System Engineering

The challenge of service discovery has been around in Information System Engineering (ISE) for at least two decades and has been continually addressed through various approaches [3,4,6,7,13,18,31,45,51]. It is highly relevant to ISE as it enables steering system design, development, and management towards supporting dynamic changes, reducing manual effort, preventing human-induced errors, and increasing scalability.

Early approaches handled service discovery as a phase of requirement engineering [13,51]. These proposals employed techniques based on pattern languages [13], controlled natural language [51], or ontologies [7,31], and eventually advanced to formal models like the Semantic Web Service Ontology (WSMO) [45]. Their application has been examined across various domains, including Smart Cities [6], the Social Internet of Things (SIoT) [3], and cloud computing [18]. Recent implementations also support prevalent OpenAPI specifications [4].

Although formal model-based approaches yield correct results, they often fall short because requirements are usually expressed using natural language. By employing LLMs and RAG, our approach overcomes this shortcoming by inherently supporting natural language queries and system documentation. In line with previous ISE approaches, our approach eases system integration by reducing manual effort and integration costs and allows agile responses to changing business needs. It supports prevalent OpenAPI documentation and natural language

requirements without additional modeling, which goes beyond the capabilities of the classical approaches.

2.3 Large Language Models

LLMs represent one of the recent advancement in the Natural Language Processing (NLP) and machine learning field [1,2,19]. Often containing billions of parameters, these models are trained on extensive text corpora to generate and manipulate human-like text [37]. They are primarily based on an encoder-decoder architecture called Transformers [44], which has been further refined to improve text generation tasks using decoder-only models such as GPT [38]. Usually, the input is a natural language task called prompt, which first needs to be translated to a sequence of input tokens. The model processes this prompt and returns an output token sequence, which can then be translated back to a natural language answer. As these models can, in general, capture intricate linguistic nuances and semantic contexts, they can be applied to a wide range of tasks, e.g., in software engineering [15]. LLMs can be used to create integration based on endpoint documentation automatically [34–36]. Yet, these face strict input token limitations, e.g., 128,000 tokens for current OpenAI models [29,36]. With this paper, we analyze how RAG can be used to preprocess API documentation to mitigate this issue.

Another approach is encoder-only models such as BERT [12], often referred to as embedding models. They allow condensing the contextual meaning of a text into a dense vector, termed embedding. Using similarity metrics such as dot product, cosine similarity, or Euclidean distance allows for assessing the similarity of two input texts. Embedding models are usually used for the similarity search in RAG systems [10], which we also do in our implementation.

2.4 LLM Agents

LLMs have shown remarkable capabilities in solving complex tasks by decomposing them in a step-by-step fashion [46] or by exploring multiple solution paths simultaneously [49]. Typically, these plans are generated iteratively by using the history of the previously generated steps to guide the generation of the next step. Additionally, recent studies have shown the potential of providing LLMs access to external tools to boost their reasoning capabilities and add further knowledge. This approach consists of prompting the LLM to interact with external tools to solve tasks, thus offloading computations from the LLM to specialized functions. Notable examples of such tools include web browsers [26], calculators [9], and Python interpreters [17]. Tool interactions are usually implemented as placeholders in the generated text, which are resolved using a previously specified set of Python functions, i.e., the developer only has to create a Python function to integrate the tool into the LLM interaction.

The LLM agent paradigm [25,28,48] combines the concepts of (i) external tool usage, (ii) the planning capabilities of LLMs, and adds a shared (iii) memory, to solve complex tasks. Given an input task, an LLM agent uses its reasoning

capabilities to decompose the task into a set of simpler subtasks. For each subtask, the LLM finds and interacts with the set of tools to solve the subtask. Then, based on the outcome of the current task and the history of previously executed subtasks, the LLM agent generates a new subtask and repeats the steps mentioned above or terminates if the original task is solved. To instruct the processing, the outcome of the tool invocations and the history of the subtasks are stored in the memory, typically consisting in the LLM agent's own context. Within this work, we apply the LLM agent paradigm to create the Discovery Agent as an LLM agent for endpoint discovery.

A critical challenge for LLM agents is the accessibility to a set of common APIs and tasks for their evaluation, e.g., tested using benchmarks like API Bank [23] or RestBench [42]. API Bank is a benchmark to evaluate the tool use of an LLM consisting of a set of APIs exposed through a search engine. Unfortunately, the available code of the benchmark is incomplete. The RestBench benchmark contains a collection of tasks and endpoints expressed using the OpenAPI specification of Spotify and TMDB [42]. As the currently most extensive available benchmark, we employ RestBench to validate our results.

OpenAPIs within LLM agents have been used in RestGPT [42] and Chain of Tools [40]. The former combines multiple LLM agents to solve complex tasks by interacting with a set of tools exposed using the OpenAPI specification. The latter solves an input query by framing the problem as a code generation task and interacts with the set of tools to generate Python code to solve the query. In contrast, our Discovery Agent does not directly interact with the endpoints stated in the OpenAPIs. Instead, it filters and returns matching endpoints that can be used for subsequent processing.

Even when considering the similarity to the LLM agent tool selection, selecting a set of tools from a larger pool to solve a specific problem remains relatively underexplored [50]. Existing research primarily focuses on the a priori selection of human-curated tools [32], heuristic-based methods for tool selection [24], choosing the relevant tool by scoring each query against every tool using a similarity metric between user queries and API names [33], and embedding-based semantic retrieval using a combination of different vector databases [50]. With our work, we contribute the analysis of preprocessing OpenAPIs into this corpus.

3 Solution Design

We first introduce the general architecture to employ RAG for endpoint discovery. As state-of-practice for service documentation, we then investigate how to chunk OpenAPIs as preprocessing for RAG.

3.1 RAG for Endpoint Discovery

RAG comprises a preprocessing step ahead of the answer generation of an LLM to enrich the prompt with additional data. Therefore, a retrieval component performs a semantic search based on some knowledge sources. Usually, the semantic

search is done by embedding similarity, and the data from the knowledge sources is reduced to small chunks to allow fine-grained information retrieval [22].

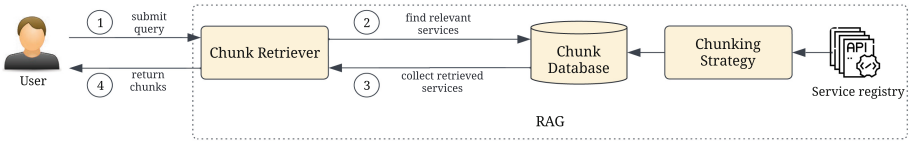


Fig. 1. RAG for Endpoint Discovery

Figure 1 depicts the application of RAG for endpoint discovery, i.e., the OpenAPI RAG. Initially, the chunking strategy determines how the chunks are created from the OpenAPIs, i.e., how many chunks are created and what they contain. Each chunk has an embedding as metadata for similarity search in addition to its content. The chunking strategy specifies which data is used as input to the embedding model to create the embedding. This input does not have to match the chunk content, e.g., it can be a summary instead of the entire content. The chunks are finally stored in the chunk database.

For retrieval, the user submits in ① a natural language query q to the chunk retriever, which converts q into the embedding e using the same embedding model as for the chunk creation. In ②, the chunk retriever queries the chunk database using e. The chunk database compares e using a similarity metric with the embeddings of the service chunks contained in the database. The results are the top k most similar chunks according to the metric, which are then returned to the chunk retriever in ③. Finally, in ④, the chunk retriever forwards the retrieved results to the user, who can add them to their prompt either manually or automatically through integration into their tooling.

The benefit of employing RAG is the insertion of only the gist of the available information, which allows picking more and only the most relevant information for the fix LLM context size. A drawback is that, based on the retrieval algorithm, not all relevant information may be retrieved. Further, fixing k reveals the advantage of controlling the result size. An alternative would be to return all chunks about a certain similarity threshold, introducing the question about the optimal cutoff.

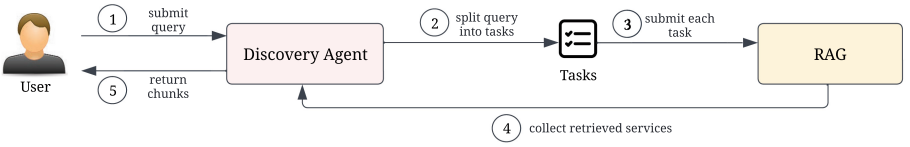


Fig. 2. Overview of the Discovery Agent Approach for Endpoint Discovery

Figure 2 shows how the Discovery Agent extends on the RAG from Fig. 1 shown in yellow hued. Instead of passing q to the RAG, the user submits it in ① to the Discovery Agent, which then iteratively decomposes q into a set of fine-grained tasks in ②. Breaking down the query into smaller, more manageable tasks can potentially fill the gap between the coarse semantics of the query and the specificities in the services documentation. In ③, the Discovery Agent submits each task to the RAG to retrieve the set of relevant chunks to solve the current task specifically. Finally, in ④, the Discovery Agent collects the retrieval results of each individual task, filters them, and repeats ② if q needs further processing or returns the results to the user in ⑤.

3.2 OpenAPI Chunking Strategies

A critical step in the RAG workflow is creating the chunks for the chunk database. Embedding models typically have a limited input token size, and real-world service registries can contain tens of thousands of services, each containing multiple potentially lengthy endpoints due to detailed descriptions or extensive input and output schemas. So, a single service might not fit into the context size of the embedding model or even exceed the limit of the LLM that further processes the output of the RAG system. In addition, service documentation can also feature additional metadata that, while valuable for understanding service details, is not necessarily relevant for composing services to solve a query.

Table 1. Implemented Chunking Strategies

Category	Splitting	Refinement	Meta-Parameters
Token-based	No split	Token chunking	m (model), s (chunk size), l (overlap)
	Endpoint split	Token chunking	m (model), s (chunk size), l (overlap)
	Endpoint split	Remove examples	m (model)
	Endpoint split	Relevant fields	m (model)
	JSON split	Token chunking	m (model), s (chunk size), l (overlap)
LLM-based	Endpoint split	Query	m (model)
	Endpoint split	Summary	m (model)

To determine advantageous chunking strategies, we employ the seven different chunking strategies presented in Table 1. Input is always an OpenAPI specification, and output is a list of chunks. The chunking strategies can be categorized into *token-based* and *LLM-based* strategies. Each strategy consists of a *splitting* method, which dissects the OpenAPI specification into a list of intermediate chunks, and another *refinement* step, which converts the intermediate chunks to the final list of chunks. In addition, there is the meta-parameter for the used embedding model m. For the token chunking refinement step, there is also the

chunk size s in tokens and their overlap l, i.e., how many tokens two consecutive chunks share, in tokens.

For the token-based approaches, we consider three main splitting methods. The *no split* method returns a single intermediate chunk for each OpenAPI containing the whole specification. The *endpoint split* divides the OpenAPI into one chunk per endpoint. The *JSON split* is a built-in LlamaIndex[1] splitting strategy tailored to files in the JSON format. This strategy parses the JSON file and traverses it using depth-first search, collecting leaf nodes, i.e., key-value pair where the value is a primitive type, e.g., strings, numbers, etc. During this traversal, the parser concatenates keys and values into single lines of text to create a comprehensive textual representation of each leaf node.

For the refinement, we implemented *token chunking*, *remove example*, and *relevant field*. The *token chunking* splits each intermediate chunk into a list of fixed-size chunks of s tokens respecting an overlap of l tokens with the previous node. The *remove example* removes the `requestBody` and recursively all `examples` fields for each endpoint as these are typically lengthy but contribute little information. The *relevant field* extracts representative fields, i.e., service title, service description, endpoint verb, endpoint path, and endpoint description, which contribute much information but few tokens.

For the LLM-based processing strategies, we apply the endpoint split and a *summary* (similar to [27]) and *query* approach for refinement. In the *summary* approach, we prompt an LLM to generate a summary for each OpenAPI endpoint. For the *query* approach, we instruct the LLM to generate a possible query matching the OpenAPI endpoint, as this might be closer to a possible input query than the summary. For both approaches, we only consider the LLM output for the embedding creation. The chunk content remains the original OpenAPI endpoint information. The no split and JSON split splitting methods can only be used with token chunking since all other refinement strategies rely on exactly one endpoint as an intermediate chunk.

4 Evaluation

To evaluate the OpenAPI RAG and the Discovery Agent, we implement it as a fully operational prototype. Then, we employ the RestBench [42] benchmark to validate it in a real-world setting.

4.1 Implementation

We implement the OpenAPI RAG and Discovery Agent approaches as open-source prototypes based on the LlamaIndex library (see Footnote 1). For the prototypes, we rely solely on OpenAPIs as the state-of-practice for service descriptions. All sources and results are available online.[2]

[1] https://github.com/run-llama/llama_index.
[2] https://doi.org/10.18419/darus-4605.

For the OpenAPI RAG, we focus on the components presented in Fig. 1. At the first start, the system loads the OpenAPIs and applies a chunking strategy to create chunks and their embeddings for their later retrieval. The chunks contain thereby the information from the OpenAPIs, e.g., a whole endpoint or a part of it. A chunk embedding does not necessarily have to match the chunk's content; for example, the content can be the endpoint, and the embedding is created using a natural language summary of the endpoint. Thus, the matching is performed based on the embedding, and the result returned is the chunk's content, which can include additional information not required for the matching process. As the service database, we use FAISS, which allows the storage and the similarity search of chunks [14]. We use a so-called QueryEngine from LlamaIndex for the chunk retriever, which allows us to query a chunk database based on textual input.

We realize the Discovery Agent from Fig. 2 using a LlamaIndex OpenAIAgent, which implements the LLM agent pattern for OpenAI's LLMs. An OpenAIAgent takes a list of tools, i.e., Python functions with a name and a description as parameters, and interacts with these using the OpenAI API. We implement two strategies for the tools. In the first strategy (*Query*), we use the OpenAPI RAG as input for a LlamaIndex QueryEngineTool, which allows the agent to interact with the RAG on demand. This has the advantage of being a simple, straightforward implementation but may increase the token count as the results of the RAG are fed into the chat history, which is transferred to the LLM for the reasoning on this data. The second strategy (*Summary*) uses a RAG with chunks of the endpoint's verb, path, and summary as contents and for their embeddings. We create the summary by instructing an LLM to create it based on the endpoint information, i.e., as in the summary chunking strategy. This should reduce the token count, as the chunks are much smaller, as not all endpoint details are returned and processed. To account for the same function as the first approach with the OpenAPI RAG and provide all information, we introduce a second tool, which takes the endpoint verb and path as input parameters and returns the whole endpoint information. The complete data is only inserted into the history for indispensable endpoints.

To enable measuring the retrieved endpoints, we attach the endpoint information, i.e., verb and path, to each chunk as metadata. For the endpoint split splitting strategies, we take the information from the endpoint. For the other strategies, we first attach a list of all endpoints to the nodes before splitting and then filter on the endpoint paths in the final chunks after splitting. So, for each chunk, we know to which endpoint or endpoints it relates to.

4.2 Dataset and Metrics

We evaluate our approach using the RestBench benchmark, covering the Spotify and TMDB OpenAPI specifications [42]. With 40 endpoints for Spotify and 54 for TMDB, this benchmark is much more complex than usual Service-Oriented Computing (SOC) case studies containing usually just three to seven

endpoints [35]. Nevertheless, a holistic benchmark covering various domains is still missing (see Sect. 2).

RestBench contains 57 queries for Spotify and 100 for TMDB. For each of these queries, a solution set of endpoints is given, i.e., one to four endpoints that must be called to fulfill the query. For example, one query is "Who directed the top-1 rated movie?" The solution contains the endpoints "GET /movie/top_-rated" and "GET /movie/{movie_id}/credits."

As embedding models, we employ OpenAI's `text-embedding-3-large` [30] as one of the currently leading proprietary models. As open-source models, we utilize BAAI/`bge-small-en-v1.5` [47], which is relatively small while still producing reasonable results, allowing the model to be executed on commonly available hardware like laptops, and Nvidia's `NV-Embed-v1` [20] as one of the leading open-source models. For the LLM, we use OpenAI's `gpt-4o-2024-05-13`.

We evaluate the quality of the retrieved information in terms of *accuracy* and the token count of the returned result. We measure the accuracy using standard information retrieval metrics, namely recall $= \frac{\text{TP}}{\text{TP+FN}}$, precision $= \frac{\text{TP}}{\text{TP+FP}}$, and $\text{F1} = \frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{recall+precision}}$.

To correctly solve the query q, a service discovery approach's recall should ideally be maximal. However, this could result in retrieving several irrelevant services, turning into a drop in precision. The F1 score represents a balance between recall and precision.

4.3 RAG

Table 2 shows the RestBench results for the OpenAPI RAG on the Spotify API. In recall, the JSON split method performs exceptionally well, especially with a high chunk size s, as this approach densely packs the information from the JSON into the chunks by removing all formatting. For precision and F1, the endpoint splitting approaches perform best because each chunk corresponds to precisely one endpoint. Differences between the models are minor, except that the `bge-small-en-v1.5` performs worse for the no split approach. We also tested $s = 2048$ and $s = 4096$, which are not reported here for space reasons. We show $s = 1024$ because it is the default chunk size of LlamaIndex and $s = 8191$ because it is the maximum input token count for the OpenAI model. It is worth mentioning that with an increasing chunk size, the token size of the returned result also increases. Generally, a higher recall seems to correlate with a higher token count, e.g., no splitting with $s = 1024$, and $l = 0$ has 4717 tokens output on average. In contrast, the JSON split has 10056 with the same parameters, but this needs further analysis. Due to length limitations, we cannot show the token count comparison and other values for top k here. We also tested top $k = 5$ and top $k = 20$. Recall increases with a higher top k, but precision drops, which relates to retrieving more chunks, revealing some relevant endpoints, and adding unrelated endpoints. In Table 2, this can be seen with a high recall but relatively low precision. Additional data is available in the complementary material (see footnote 2).

Table 2. Results for the OpenAPI RAG for top $k = 10$ with the Spotify API. The overlap is in tokens. O represents OpenAI's `text-embedding-3-large`. S and N represent `bge-small-en-v1.5` and `NV-Embed-v1`, respectively. TC is for token chunking, RE for remove examples, and RF for relevant fields. Recall, precision, and F1 are in percent. The highest values per column are marked in bold.

Category	Splitting	Refinement	s	l	Recall			Precision			F1		
					O	S	N	O	S	N	O	S	N
Character	No	TC	1024	0	67	40	62	16	12	**19**	26	18	29
	No	TC	1024	50	68	49	64	16	13	**19**	26	20	29
	No	TC	8191	0	88	71	91	7	7	7	12	13	13
	No	TC	8191	50	89	66	91	7	7	7	12	13	13
	Endpoint	TC	1024	0	70	75	76	**19**	**19**	**20**	**29**	**31**	**31**
	Endpoint	TC	1024	50	71	74	76	**19**	**19**	**20**	**29**	**30**	**32**
	Endpoint	TC	8191	0	73	75	76	**19**	**19**	**20**	**29**	**30**	**31**
	Endpoint	TC	8191	50	73	75	76	**19**	**19**	**20**	**30**	**30**	**31**
	JSON	TC	1024	0	81	84	85	9	8	10	17	15	19
	JSON	TC	1024	50	77	87	85	10	9	10	18	17	19
	JSON	TC	8191	0	**97**	**95**	**100**	5	5	5	10	10	10
	JSON	TC	8191	50	**97**	**95**	**100**	5	5	5	10	9	10
	Endpoint	RE	N/A	N/A	72	75	73	**18**	**19**	**19**	**29**	**30**	30
	Endpoint	RF	N/A	N/A	71	71	73	**18**	**18**	**19**	**29**	29	30
LLM	Endpoint	Query	N/A	N/A	71	57	58	**18**	15	15	29	23	24
	Endpoint	Summary	N/A	N/A	72	74	67	**18**	**19**	17	**29**	**30**	27

Table 3 presents the OpenAPI RAG RestBench results for the TMDB API. The TMDB OpenAPI is more complex in length and extent than the Spotify OpenAPI. In this case, the endpoint split-based approaches performs best in precision and F1. The no split approaches achieve high values in precision due to their low value of true positives.

Overall, the endpoint split tends to outperform no splitting. The JSON splitting benefits Spotify as the endpoints are already very dense, i.e., the endpoints do not contain examples, and schemas are only referenced. Therefore, many endpoints can be condensed into one chunk. This approach performs much worse for the lengthier endpoints in the TMDB API. The summary refinement outperforms the query refinement, leading to the Discovery Agent.

4.4 Discovery Agent

We present the RestBench results of the Discovery Agent in Table 4. For accuracy, we measure recall, precision, and F1 equally to the OpenAPI RAG experiments. For the token count, we measure the actual tokens sent from the agent to the LLM from the agent as *prompt*, the tokens received as *completion*, and their

Table 3. Results for the RAG for top $k = 10$ with the TMDB API. Schema as in Table 2.

Category	Splitting	Refinement	s	l	Recall			Precision			F1		
					O	S	N	O	S	N	O	S	N
Character	No	TC	1024	0	1	14	7	33	17	**38**	2	15	11
	No	TC	1024	50	4	13	7	**47**	**20**	**38**	7	16	11
	No	TC	8191	0	30	17	15	19	5	9	23	8	11
	No	TC	8191	50	30	16	16	18	5	7	22	7	10
	Endpoint	TC	1024	0	40	40	46	20	15	18	27	21	**26**
	Endpoint	TC	1024	50	40	40	44	21	15	17	27	22	**25**
	Endpoint	TC	8191	0	66	47	59	19	12	14	**29**	19	23
	Endpoint	TC	8191	50	66	51	58	19	13	14	**30**	21	22
	JSON	TC	1024	0	44	44	46	18	12	16	26	19	24
	JSON	TC	1024	50	48	42	41	19	11	15	27	18	22
	JSON	TC	8191	0	61	60	50	8	8	6	14	14	11
	JSON	TC	8191	50	57	54	54	8	7	7	14	12	12
	Endpoint	RE	N/A	N/A	**75**	52	**71**	17	12	16	28	19	**26**
	Endpoint	RF	N/A	N/A	58	48	57	13	11	13	21	17	21
LLM	Endpoint	Query	N/A	N/A	56	**65**	46	13	15	10	20	**24**	17
	Endpoint	Summary	N/A	N/A	69	59	65	16	13	15	**29**	22	24

sum as *total*. For the RAG approach, we accumulate the tokens of the retrieved chunks. We set top $k = 10$ and use OpenAI's `text-embedding-3-large` as the embedding model. Spotify and TMDB are the two test sets from the RestGPT benchmark. RAG are the results for the summary chunking strategy from the Tables 2 and 3. Query is the standard LlamaIndex QueryEngineTool to retrieve data from a RAG system. The summary is our approach with a QueryEngineTool for summaries and a details-on-demand fetcher. Accuracy values are in percent. #Token is the number of tokens per query averaged over all queries in the test set. The best value per row is marked in bold.

The results show that both agent approaches improve precision and F1 but reduce recall. The Query approach increases the tokens in the prompt. Contrarily, the Summary approach significantly outperforms the RAG in the total token count as shown in Table 4 by 19% and 62% and the query approach by 88% and 93% fewer tokens for Spotify and TMDB, respectively. The completion token count is by a magnitude smaller than the prompt token count for the agent approaches, which is relevant as completion tokens are usually more expensive than prompt tokens. No LLM is invoked in the RAG approach, so the completion tokens are zero.

Table 4. Results of the Discovery Agent experiments.

		Spotify			TMDB		
		RAG	Query	Summary	RAG	Query	Summary
Accuracy	Recall	**71.92**	63.70	66.44	**69.33**	43.11	46.67
	Precision	18.42	67.39	**70.29**	15.60	45.97	**50.97**
	F1	29.32	65.49	**68.30**	25.47	44.50	**48.72**
#Token	Prompt	4233.65	8606.87	**3125.21**	41001.46	65699.75	**4544.57**
	Completion	**0.00**	262.30	256.26	**0.00**	242.65	231.73
	Total	4233.65	8869.18	**3411.47**	41001.46	65942.40	**4776.30**

4.5 Discussion

We demonstrated the effectiveness of the OpenAPI RAG and the Discovery Agent using our implementation. They are able to retrieve large portions of relevant data while not revealing all relevant information in all cases.

To address RQ1, we implemented the OpenAPI RAG to apply RAG for endpoint discovery with seven chunking strategies and numerous parameter combinations. We showed its effectiveness using the RestBench benchmark. Overall, our prototype demonstrates how to adequately reduce the token size to fit into the LLM context size while maintaining most of the relevant information. Regarding chunking strategies, endpoint split-based chunking strategies achieve favorable accuracies. Limitations are primarily that the RAG results may not contain all relevant information and the low precision due to the retrieval of exactly k chunks. Additional research is needed to improve the retrieval performance further and prove the results in a generalized setting across multiple domains.

For RQ2, we introduced the Discovery Agent, which transfers the LLM agent pattern to endpoint discovery. Especially using the Summary approach, the Discovery Agent showed strong improvement over the OpenAPI RAG in terms of precision, F1, and token count. Further research is needed to improve the decline in recall due to the processing through the LLM.

Limitations. While we rely on the research benchmark RestBench for our results, which covers two extensive OpenAPIs, queries, and ground truth, it is still limited to these two services. OpenAPI RAG systems in practice may operate on much larger datasets. For the data processing, we rely on standard RAG implementations like LlamaIndex, which are already designed to operate on large amounts of data. The performance evaluation, especially in larger real-world scenarios, remains open for future research.

The applicability of the OpenAPI RAG depends on the availability of service documentation. We try to mitigate this issue by relying on widely adopted OpenAPI specifications, but this might not be valid for all domains. A solution to consider is automatically generating service documentation using an LLM.

Another factor influencing the discovery is the quality of the OpenAPIs. The discovery may fail if no descriptions, meaningful naming, or erroneous information is given. This is not an issue of the approach, as a human developer would face the same problem, but it highlights the importance of high-quality documentation.

In addition to the presented chunking strategies, additional and more advanced strategies, e.g., CRAFT [50], could be added to the OpenAPI RAG. These could improve retrieval performance by combining multiple strategies or by creating a custom chunking strategy for a specific kind of service documentation.

Another advancement could also be creating a custom embedding model tailored explicitly to service descriptions and service description chunks. This model may also be trained for one specific chunking strategy or intended use case. Additionally, the RAG output may be trimmed to boost precision. This could be done by, e.g., employing a similarity threshold.

The presented Discovery Agent could be further improved to handle whole service compositions. In this case, the agent would be extended by an additional component for the service composition. The user would only submit their task to the agent to retrieve the executable service composition solving this task.

Besides the capabilities of the RAG system, resource consumption is a major issue in LLM-based systems. The OpenAPI RAG only uses embedding models. These are much more efficient than LLMs, resulting in costs in fractions of a cent per query. In contrast, the Discovery Agent requires invoking an LLM, which requires significantly more resources and may result in substantial API fees or operational costs. Further work is needed to reduce this resource footprint.

Implications for Information System Engineering. The recall, precision, and token count metrics not only provide a clear performance overview but also contribute to the applicability in ISE. When employed in practice, service discovery with RAG reduces integration effort, costs, and time, leading to strategic benefits over manual integration. Higher recall and precision reinforce these benefits by reducing the manual effort needed to make the solution operational.

Additional automation improves scalability by reducing reliance on an inflexible workforce. These improvements streamline IT workflows and increase resilience by promoting rapid adaptation. They also make the overall product more agile, simplify updates to integrated systems, and accelerate the adoption of changing business needs. Thus, higher performance metrics enable automation by reducing the need for manual corrections.

Additional research should compare worktime reduction to operational costs once the system is implemented. This can be accomplished by performing a user study that measures the time required to complete tasks manually versus using the OpenAPI RAG while also assessing monetary and environmental costs. For example, an automotive vendor wants to integrate on-road services, e.g., parking spot booking. The OpenAPI RAG offers the developer available and relevant services, eliminating manual search.

5 Concluding Remarks

The service discovery challenge has been around for a long time in SOC to integrate different ISs. With the application of automated LLM-based service composition approaches, the LLM input context limitations have become prominent, as the entire service documentation often does not fit into the input context, necessitating the preselection of relevant information. To address this issue, we proposed an OpenAPI RAG, which facilitates semantic search based on state-of-practice OpenAPIs and reduces the input token size. Further, we show an advanced integration through a Discovery Agent, which can retrieve service details on demand to reduce the input token count further. Our evaluation based on the RestBench benchmark shows that our approach is viable and performing. Limitations are especially in the restriction of RestBench to two services of the entertainment domain. We will address this in an extended version of this work. Further improvements are in optimizing the implementation and extending the agent for additional tasks, e.g., whole service compositions. We leave this for future work.

Acknowledgments. This work was partially funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) project Software-Defined Car (SofDCar) (19S21002). The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the GSaME.

Disclosure of Interests. The authors Pesl and Aiello are listed as inventors of a patent [34], which covers automated automotive service compositions using LLMs.

References

1. Achiam, J., et al.: GPT-4 technical report (2023). https://arxiv.org/abs/2303.08774
2. AI@Meta: Llama 3 model card (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md
3. Aljubairy, A., Zhang, W.E., Sheng, Q.Z., Alhazmi, A.: SIoTPredict: a framework for predicting relationships in the social internet of things. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) CAiSE 2020. LNCS, vol. 12127, pp. 101–116. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49435-3_7
4. Apostolakis, I., Mainas, N., Petrakis, E.G.: Simple querying service for openAPI descriptions with semantic extensions. Inf. Syst. **117**, 102241 (2023). https://doi.org/10.1016/j.is.2023.102241
5. Baresi, L., Miraz, M.: A distributed approach for the federation of heterogeneous registries. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 240–251. Springer, Heidelberg (2006). https://doi.org/10.1007/11948148_20
6. Ben-Sassi, N., Dang, X.-T., Fähndrich, J., Görür, O.-C., Kuster, C., Sivrikaya, F.: Service discovery and composition in smart cities. In: Mendling, J., Mouratidis, H. (eds.) CAiSE 2018. LNBIP, vol. 317, pp. 39–48. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92901-9_4

7. Bianchini, D., et al.: Ontology-based methodology for e-service discovery. Inf. Syst. **31**(4), 361–380 (2006). https://doi.org/10.1016/j.is.2005.02.010, the Semantic Web and Web Services
8. Bohn, H., Golatowski, F., Timmermann, D.: Dynamic device and service discovery extensions for WS-BPEL. In: ICSSSM 2008, pp. 1–6. IEEE (2008). https://doi.org/10.1109/ICSSSM.2008.4598557
9. Cobbe, K., et al.: Training verifiers to solve math word problems (2021). https://arxiv.org/abs/2110.14168
10. Cuconasu, F., et al.: The power of noise: redefining retrieval for RAG systems. In: SIGIR, vol. 47, pp. 719–729 (2024). https://doi.org/10.1145/3626772.3657834
11. Curbera, F., et al.: Unraveling the web services web: an introduction to SOAP, WSDL, and UDDI. IEEE Internet Comput. **6**(2), 86–93 (2002). https://doi.org/10.1109/4236.991449
12. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT 2019, pp. 4171–4186 (2019)
13. Dourdas, N., Zhu, X., Maiden, N., Jones, S., Zachos, K.: Discovering remote software services that satisfy requirements: patterns for query reformulation. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 239–254. Springer, Heidelberg (2006). https://doi.org/10.1007/11767138_17
14. Douze, M., et al.: The Faiss library (2024). https://arxiv.org/abs/2401.08281
15. Fan, A., et al.: Large language models for software engineering: survey and open problems (2023). https://arxiv.org/abs/2310.03533
16. Fikouras, I., Freiter, E.: Service discovery and orchestration for distributed service repositories. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, pp. 59–74. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-24593-3_5
17. Gao, L., et al.: Pal: program-aided language models. In: International Conference on Machine Learning, pp. 10764–10799. PMLR (2023)
18. Jerbi, I., et al.: Request relaxation based-on provider constraints for a capability-based NaaS services discovery. In: Indulska, M., Reinhartz-Berger, I., Cetina, C., Pastor, O. (eds.) CAiSE 2023. LNCS, vol. 13901, pp. 611–627. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-34560-9_36
19. Kim, M., Stennett, T., Shah, D., Sinha, S., Orso, A.: Leveraging large language models to improve REST API testing. In: ICSE, vol. 44, pp. 37–41 (2024). https://doi.org/10.1145/3639476.3639769
20. Lee, C., et al.: NV-embed: improved techniques for training LLMs as generalist embedding models (2024). https://arxiv.org/abs/2405.17428
21. Lemos, A.L., Daniel, F., Benatallah, B.: Web service composition: a survey of techniques and tools. ACM Comput. Surv. **48**(3) (2015). https://doi.org/10.1145/2831270
22. Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: NeurIPS, vol. 33, pp. 9459–9474. Curran Associates (2020)
23. Li, M., et al.: API-bank: a comprehensive benchmark for tool-augmented LLMs. In: EMNLP. Association for Computational Linguistics (2023). https://doi.org/10.18653/v1/2023.emnlp-main.187
24. Liang, Y., et al.: Taskmatrix.AI: completing tasks by connecting foundation models with millions of APIs. Intell. Comput. **3**, 0063 (2024). https://doi.org/10.34133/icomputing.0063
25. Mialon, G., et al.: Augmented language models: a survey (2023). https://arxiv.org/abs/2302.07842

26. Nakano, R., et al.: WebGPT: browser-assisted question-answering with human feedback (2021). https://arxiv.org/abs/2112.09332
27. Nogueira, R., Yang, W., Lin, J., Cho, K.: Document expansion by query prediction (2019). https://arxiv.org/abs/1904.08375
28. OpenAI: Function calling and other API updates (2024). https://openai.com/index/function-calling-and-other-api-updates/. Accessed 18 Jul 2024
29. OpenAI: GPT-4 Turbo in the OpenAI API (2024). https://help.openai.com/en/articles/8555510-gpt-4-turbo-in-the-openai-api. Accessed 19 Nov 2024
30. OpenAI: New embedding models and API updates (2024). https://openai.com/blog/new-embedding-models-and-api-updates. Accessed 18 Jul 2024
31. Oriol, X., Teniente, E.: An ontology-based framework for describing discoverable data services. In: Krogstie, J., Reijers, H.A. (eds.) CAiSE 2018. LNCS, vol. 10816, pp. 220–235. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91563-0_14
32. Parisi, A., Zhao, Y., Fiedel, N.: TALM: tool augmented language models (2022). https://arxiv.org/abs/2205.12255
33. Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: large language model connected with massive APIs (2023). https://arxiv.org/abs/2305.15334
34. Pesl, R.D., Klein, K., Aiello, M.: Verfahren zur Nutzung von unbekannten neuen Systemdiensten in einer Fahrzeuganwendung (2024). Patent DE102024108126A1
35. Pesl, R.D., Stötzner, M., Georgievski, I., Aiello, M.: Uncovering LLMs for service-composition: challenges and opportunities. In: ICSOC 2023 WS. Springer, Cham (2024). https://doi.org/10.1007/978-981-97-0989-2_4
36. Pesl, R.D., et al.: Compositio Prompto: an architecture to employ large language models in automated service computing. In: ICSOC 2024. Springer, Cham (2024)
37. Radford, A., et al.: Better language models and their implications. OpenAI blog **1**(2) (2019). https://openai.com/index/better-language-models/. Accessed 28 Nov 2024
38. Radford, A., et al.: Improving language understanding by generative pre-training (2018)
39. Santana, J.M.S., Petrova, M., Mahonen, P.: UPnP service discovery for heterogeneous networks. In: IEEE PIMRC, vol. 17, pp. 1–5. IEEE (2006)
40. Shi, Z., et al.: Chain of tools: large language model is an automatic multi-tool learner (2024). https://arxiv.org/abs/2405.16533
41. Soki, A.T., Siqueira, F.: Discovery of RESTful Web Services Based on the OpenAPI 3.0 Standard with Semantic Annotations. In: AINA, pp. 22–34. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-57853-3_3
42. Song, Y., et al.: RestGPT: connecting large language models with real-world applications via restful APIs (2023). https://arxiv.org/abs/2306.06624
43. Thönes, J.: Microservices. IEEE Softw. **32**(1), 116–116 (2015). https://doi.org/10.1109/MS.2015.11
44. Vaswani, A., et al.: Attention is all you need. In: NeurIPS, vol. 30 (2017)
45. Wang, H.H., et al.: A formal model of the semantic web service ontology (WSMO). Inf. Syst. **37**(1), 33–60 (2012). https://doi.org/10.1016/j.is.2011.07.003
46. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models. NeurIPS **35**, 24824–24837 (2022)
47. Xiao, S., Liu, Z., Zhang, P., Muennighoff, N.: C-pack: packaged resources to advance general Chinese embedding (2023). https://arxiv.org/abs/2309.07597
48. Yao, S., et al.: React: synergizing reasoning and acting in language models (2023). https://arxiv.org/abs/2210.03629

49. Yao, S., et al.: Tree of thoughts: deliberate problem solving with large language models. In: NeurIPS, vol. 36 (2024)
50. Yuan, L., et al.: CRAFT: customizing LLMs by creating and retrieving from specialized toolsets (2024). https://arxiv.org/abs/2309.17428
51. Zachos, K., Maiden, N., Zhu, X., Jones, S.: Discovering web services to specify more complete system requirements. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 142–157. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72988-4_11

Conformance-Checking

Managing and Anticipating Out-of-Order Events in Online Compliance Monitoring

Silvano Colombo Tosatto[1], Hannah Burke[1], Nick R. T. P. van Beest[1(✉)], and Heerko Groefsema[2]

[1] Data61, CSIRO, Brisbane, Australia
{silvano.colombotosatto,hannah.burke,
nick.vanbeest}@data61.csiro.au
[2] University of Groningen, Groningen, The Netherlands
h.groefsema@rug.nl

Abstract. When monitoring compliance of event streams at runtime, it is usually assumed that the events are received in the order in which they are produced. However, in reality, event data is typically transmitted from distributed sources and may be received out-of-order as a result. This is problematic for the compliance analysis in terms of accuracy, latency and computational efficiency. In this paper, we present an approach to manage these problems by exploiting knowledge of the event sources to identify the reliable prefix of the event stream that is guaranteed to be in order, allowing a reliable analysis on that prefix. Subsequently, we use the prefix alignment of the observed event stream and a process model to quantify its reliability and identify events that might arrive out-of-order, allowing to calculate a confidence score of compliance results. This improves efficiency and provides compliance results with awareness of potential uncertainty, enabling decision makers to respond appropriately to the identified issues.

Keywords: Out-of-Order Event Streams · Process Models · Online Monitoring · Compliance · Conformance · Alignments

1 Introduction

Process monitoring refers to the techniques used to analyse the compliance or conformance of processes during their execution [1,2]. Process executions are commonly recorded in event logs or streams, where events—representing tasks in a process—are captured for each step in the process execution. Generally, a *monitor* assumes to receive events in a stream from their recording source in the order in which they are produced. However, as events may be received from different sources and communication channels, events may be received by the monitor in a different order than they were produced, due to different communication speeds, delays or even failures.

Such *out-of-order* events are problematic for the ongoing analysis, as the results are computed on-the-fly [2]. As such, the evaluation is performed over states of the process that are inaccurate, and this only becomes apparent when delayed events are received and their recorded timestamp reveals the discrepancy. For example, a *payment* event

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 151–166, 2025.
https://doi.org/10.1007/978-3-031-94571-7_9

generated by one server on the network may not be received by the monitor running on a different server until after a *shipment* event was received, even though the payment was executed beforehand. This is particularly problematic in environments with strict runtime compliance requirements, as this may lead to results that are either false positives or false negatives that are then discovered too late to mitigate the situation.

When out-of-order events are detected, the ongoing evaluation must be backtracked to correct the order of the event stream before resuming the evaluation. To do this, the monitor needs to keep the event stream in memory, quickly resulting in a large memory footprint. Furthermore, since compliance results should be reported immediately, it is possible that they may be retracted when an out-of-order event is received that changes the outcome. Out-of-order events should, therefore, be anticipated before they occur.

In this paper, we present a novel approach to mitigate the out-of-order problem of streams and can (i) derive the point up to which a stream of events is factually in-order, and (ii) determine the probability of out-of-order events in the remaining stream, using knowledge about the network architecture, event history and alignment with the process model. Using this, we allow the compliance monitor to evaluate the event stream online, discard expendable events from memory and present evaluation results accordingly. We do so under the assumptions that (i) the communication channels are reliable (e.g., TCP rather than UDP) and (ii) all sources of events possess synchronised clocks.

Without requiring additional architectural components (as in the case of heartbeats) or risking possible out-of-orders for portions of the stream no longer in memory (as in the case of statistical punctuation), the approach is *conservative* in when it removes data from memory. Nonetheless, we show a significant reduction in the memory footprint of the monitor. Conversely, by leveraging knowledge about the process, the approach is *optimistic* in presenting preliminary results that are grounded in probability. We demonstrate the accuracy of these probabilities to measure the confidence of the analysis and show it is effective for different levels of conformance to the process model.

The remainder of the paper is structured as follows. In Sect. 2 we discuss the preliminary concepts, followed by a description of the problem in Sect. 3. Next, we discuss how to derive the point up to which an event stream can be deemed in-order in Sect. 4, and subsequently determine the odds of out-of-order events in Sect. 5. Using this knowledge, we evaluate the effects that out-of-order predictions have on monitoring applications in Sect. 6. Finally, we discuss related work in Sect. 7 and conclude our work in Sect. 8.

2 Preliminaries

2.1 Event Logs, Traces and Streams

When analysing processes at runtime, the observed behaviour of each running process instance is typically captured by *events* collected in an *event log* or an *event stream*. These events consist of a set of relevant data attributes, which represent information about tasks in the process. This information could contain the creation and alterations of artifacts of the process, the task or artifact id, task name, recorded sensor values, the timestamp of the event, and the resource triggering the event. An event and its attributes are defined formally as follows:

Definition 1 (Event). *An* event e *is a set of* attributes A *that describe (a change in) the state of a system. Let* $\mathbb{D}(a)$ *denote the domain of an attribute* $a \in A$*, and let* ϵ *be the empty assignment. Given the universe of events* $\mathcal{E}$*, the function* $v : \mathcal{E}, A \rightarrow \mathbb{D}(a) \cup \{\epsilon\}$ *obtains the value of the attribute* $a \in A$ *as follows:*

$$v(e, a) = \begin{cases} v \in \mathbb{D}(a) & \text{if } a \in e \\ \epsilon & \text{otherwise} \end{cases}$$

With abuse of notation, we use the shorthand $\lambda(e) = v(e, name)$ and $\theta(e) = v(e, timestamp)$ to obtain the name and timestamp of an event e. In addition, we write $e_i \prec e_j$ iff $\theta(e_i) < \theta(e_j)$.

Definition 2 (Trace, Event log). *A trace* τ *is a finite sequence of* n *events* $\tau = e_1, \ldots, e_n$*, with* $e_i \in \mathcal{E}$ *for* $i \in [1 \ldots n]$*. We write* $e \in \tau$ *iff* $\exists x \in [1 \ldots n] : e_x = e$*. An event log* L *is a finite collection of traces over* $\mathcal{E}$*.*

Definition 3 (Event stream). *Given the event universe* $\mathcal{E}$*, an event stream* ψ *is an infinite sequence* $\Psi : \mathbb{N}^+ \rightarrow \mathcal{E}$*.*

2.2 Petri Nets

The intended behaviour of a process can be represented graphically by means of a process model such as a Petri net, allowing them to be analysed formally [3]. A Petri net is a directed bipartite graph with two types of nodes: *places* (denoted by circles) and *transitions* (denoted by rectangles). Transitions represent activities within a process, while places represent conditions. Nodes are connected via directed edges called *arcs*.

Definition 4 (Petri net). *A* Petri net $\mathcal{P}$ *is a tuple* $(\mathbb{P}, \mathbb{T}, \mathbb{A}, \lambda)$*, where:*

- $\mathbb{P}$ *is a finite set of* places*,*
- $\mathbb{T}$ *is a finite set of* transitions*, such that* $\mathbb{P} \cap \mathbb{T} = \emptyset$*,*
- $\mathbb{A} \subseteq (\mathbb{P} \times \mathbb{T}) \cup (\mathbb{T} \times \mathbb{P})$ *is a set of arcs, and*
- $\lambda : \mathbb{P} \cup \mathbb{T} \rightarrow \mathcal{L}$ *is a labelling function.*

Each place of a Petri net can contain one or more *tokens*. A *net marking* μ of a Petri net $\mathcal{P} = (\mathbb{P}, \mathbb{T}, \mathbb{A}, \lambda)$ is a function that associates places with the number of tokens on it, i.e., $\mu : \mathbb{P} \rightarrow \mathbb{N}_0$. A *marked net* is a tuple $\mathcal{P} = (\mathbb{P}, \mathbb{T}, \mathbb{A}, \lambda, \mu_0)$, where $(\mathbb{P}, \mathbb{T}, \mathbb{A}, \lambda)$ is a Petri net and μ_0 is the initial marking. The *pre-set* of y, denoted by $\bullet y$, is the set of all input nodes of y: $\bullet y = \{x \in \mathbb{P} \cup \mathbb{T} \mid (x, y) \in \mathbb{A}\}$. The *post-set* of y, denoted by $y\bullet$, is the set of all output nodes of y: $y\bullet = \{x \in \mathbb{P} \cup \mathbb{T} \mid (y, x) \in \mathbb{A}\}$. Given a net marking μ, transition $t \in T$ is said to be *enabled* if $\forall p \in \bullet t : \mu(p) > 0$. If transition t is enabled, t can *fire* (denoted by $\mu \xrightarrow{t} \mu'$), which leads to a new marking μ' such that $\forall p \in \bullet t$, $\mu'(p) = \mu(p) - 1$ and $\forall p \in t\bullet$, $\mu'(p) = \mu(p) + 1$.

An execution σ of a net $\mathcal{P}$ is either the empty sequence, if no transitions are enabled in the initial marking, or a sequence of transitions $\sigma = t_1, t_2, \ldots t_n$, with $t_i \in \mathbb{T}$, $i \in [1 \ldots n]$, such that $\mu_0 \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} \mu_n$ and there are no enabled transitions in μ_n. We write $t \in \sigma$ iff $\exists x \in [1 \ldots n] : t_x = t$.

2.3 Aligning a Trace Prefix to an Execution

Conformance checking techniques aim to evaluate whether a given trace in an event log fits the behaviours as described by a process model. An *alignment* is one of the central concepts in conformance checking and describes the relation between a trace and an execution of a Petri net as a sequence of *moves*, by matching events in the log to transitions in the Petri net [4]. Some events in the trace cannot be matched by transitions in the execution (no move in the model) or vice versa (no move in the log). We denote "no move" by '$\gg$', which is a special symbol that is neither an event nor a transition.

Definition 5 (Move). A *move* over a trace τ in an event log and an execution σ of a Petri net is a pair $(e,t) \in (((\mathcal{E} \cup \{\gg\}) \times (\mathbb{T} \cup \{\gg\})) \setminus \{(\gg,\gg)\})$, where:

- (e,t) is a *synchronous move* if $e \in \tau$, $t \in \sigma$ and $\lambda(e) = \lambda(t)$;
- (e,t) is a *move on log* if $e \in \tau$ and $t = \gg$; and
- (e,t) is a *move on model* if $e = \gg$ and $t \in \sigma$.

Moves on the log or model only are referred to as *asynchronous* moves. The set of *legal* moves is denoted by $\mathbb{M}_{LM}$, containing all synchronous and asynchronous moves.

Definition 6 (Alignment). An *alignment* of a trace τ and an execution σ is a finite sequence $\gamma \in \mathbb{M}_{LM}^*$ where the first elements of the ordered moves without '$\gg$' *equal* τ and the second elements of the ordered moves without '$\gg$' equal σ.

The function $\delta : \mathbb{M}_{LM} \rightarrow \mathbb{N}_0$ assigns non-negative costs to moves. The cost of an alignment γ is denoted by $\delta(\gamma)$ and is equal to the sum of the costs of all its moves. We assign $\delta(e,t) = 0$ for synchronous moves, $\delta(e,\gg) = 1$ for asynchronous moves on the log, $\delta(\gg,t) = 1$ for each asynchronous move on the model before $e_n \in \tau$ and $\delta(\gg,t) = 0$ for each asynchronous move on the model after $e_n \in \tau$. This way, we obtain a prefix alignment and avoid penalising subsequent asynchronous moves on the model that describe how the trace can proceed in the future (i.e. transitions in the model that have not been observed yet in the trace), rather than discrepancies between the model and the trace. In the remainder of this paper, we refer to prefix alignments simply as alignments. The *optimal alignment* of τ and $\mathcal{P}$, denoted $\alpha(\tau,\mathcal{P})$, is the alignment that yields the lowest cost among all possible alignments.

3 The Problem and Its Consequences

In online compliance monitoring, the initial state of the event stream is a sequence of events that is then appended with new events, which are verified against a set of compliance rules upon arrival. These compliance rules can be specified in different types of logic, such as, e.g., temporal logic or some first-order logic. With a warm startup, this initial sequence of events is non-empty, as otherwise the compliance results for in-progress instances will be inconclusive. Events may be produced and submitted to the monitor by different distributed sources, such as servers or sensors. Therefore, there will be latency between the time that events are produced and they are received. Variance in latency can mean the monitor may not receive the events in the same order as they are produced by different sources. If the order of the events in the observed trace is not the same as the chronological order of the events according to their timestamps, the stream is said to be *out-of-order*.

Consider the Petri net in Fig. 1. A monitor receives events of this process from two distributed sources, r_1 and r_2, where events A, B, C and G are transmitted from r_1 and D, E and F from r_2. Suppose an event stream ψ is generated from a partial execution $\sigma = A, B, C, D, E$ (Fig. 2a). Due to some network latency at r_1, receipt of C is delayed. Figure 2b shows trace τ as observed by the monitor. Events A and B were received in time, but C is not included in the observed trace yet due to latency. At this point, we are unaware of out-of-order event C and the observed trace τ seems, and is, in-order.

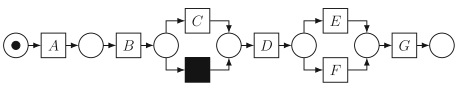


Fig. 1. Petri net example.

Event:	A	B	C	D	E	
Data source:	r_1	r_1	r_1	r_2	r_2	...
Timestamp:	θ_1	θ_2	θ_3	θ_4	θ_5	

(a) Stream of events ψ

Event:	A	B	D	E
Data source:	r_1	r_1	r_2	r_2
Timestamp:	θ_1	θ_2	θ_4	θ_5

(b) Observed sequence τ

Event:	A	B	D	E	C
Data source:	r_1	r_1	r_2	r_2	r_1
Timestamp:	θ_1	θ_2	θ_4	θ_5	θ_3

(c) Reordering τ when receiving C.

Fig. 2. Event observations representing execution σ.

Although the observed trace seems to be in-order, analysis at this point may result in an erroneously computed monitoring state, potentially affecting the compliance results produced by the monitor. Consider the rule *"E is mutually exclusive with C, and an occurrence of F requires an earlier occurrence of C"*. In the observed state of Fig. 2b, the compliance monitor would erroneously conclude that the observed trace is compliant. In addition, as C is allowed to be skipped in the model (cf. Fig. 1), it is difficult to identify whether C is skipped or missing in this case. Conversely, if F had occurred instead of E, the compliance monitor would erroneously detect a violation.

This example also highlights the existence of discrepancies between the process model and the compliance rules applicable to the process it describes. It demonstrates that there can exist executions in the model that are **non-compliant**, as well as **compliant** executions that are not in the model. In the context of this paper, a process model is used to indicate the executions that typically happen, whereas compliance rules are logical properties used to classify traces as **compliant** or **non-compliant**. As a result, an execution can **conform** to the process model (i.e. follow the behaviour as allowed by the model), but be **non-compliant** (i.e. fail a particular compliance rule), or vice versa.

By definition, the generated event stream ψ is never out-of-order and represents the execution of the process. Conversely, we use τ to refer to *observed* sequences of events by a monitor. If τ is detected to be out-of-order, the sequence is reordered according to its timestamps. For example, when receiving C, the timestamp (θ_3) reveals that it is out-of-order and τ is reordered as shown in Fig. 2c. The monitor, however, may already have wrongly concluded violations before reordering, or may miss violations due to it. Considering the out-of-order problem, a trace τ can be in three possible states:

– **In-order**: when τ is a prefix of ψ.

– **In-deceptive-order**: when τ is not a prefix of ψ and has no out-of-order event.
– **Out-of-order**: when the last received event of τ is an out-of-order event.

When τ is **in-order**, analysis by the monitor over the observed trace of events would match a hypothetical analysis over the event stream ψ, and its results are correct. However, when τ is **out-of-order**, it must be reordered before proceeding the analysis. By reordering (i) τ becomes **in-order** or **in-deceptive-order**, (ii) the compliance analysis of τ must be re-evaluated after reordering, and (iii) actions taken based on earlier analysis must potentially be rolled back. When τ is **in-deceptive-order**, from the perspective of the monitor, such a state is indistinguishable from **in-order**. As a result, τ can be in **in-deceptive-order** and become **out-of-order** at any point. In such cases, the analysis must be performed again over the re-ordered sequence. Consequently, the monitor must always be prepared for this possibility by keeping all past events in memory. To limit the resulting memory requirements and potential backtracking, we will next identify the prefix of τ that is factually in-order, such that it can be removed from memory after analysis by the monitor.

4 Reliable Prefix

The use of *reliable event sources* can limit the potentially significant memory requirements. Modern architectures typically comprise a system responsible for the management of the event log, and one or more systems that perform tasks and report their progress through events to the former. We can refer to the systems providing events as *sources* and the one receiving them as the *receiver*.

Definition 7 (Event Sources). *Given the universe of events $\mathcal{E}$ and the domain of event sources $\mathcal{R}$, an event is mapped to its source using the function $\mathfrak{r} : \mathcal{E} \rightarrow \mathcal{R}$.*

Out-of-order events occur principally due to disruptions in the communication network between the receiver and the sources. Assuming a standard communication network, we can safely assume that each individual communication channel between the monitor and its sources is handled using protocols such as TCP, to ensure that a source sending an event to the monitor is notified that the event has been successfully received before sending the next. With these considerations in mind, we assume that events from an individual source are always relatively in order, while the trace observed by the receiver when collecting events from multiple sources might contain out-of-order events.

Given the reliability of the individual communication channels between the receiver and the sources, when considering an observed trace of events from a receiver, we can partition the trace into subsequences each originating from a single source, such that each subsequence by itself is *reliable*.

Definition 8 (Source Reliable Point). *Given an observed trace of events τ, the reliable point of a source r is defined by the function $\mathsf{SRP}(\tau, r) = e_i \in \tau \mid \mathfrak{r}(e_i) = r \wedge (\nexists e_j \in \tau : \mathfrak{r}(e_j) = r \wedge e_i \prec e_j)$. If $\nexists e \in \tau : \mathfrak{r}(e) = r$ then $\mathsf{SRP}(\tau, r)$ returns $null$.*

Recall the observed out-of-order trace in Fig. 2c, where C was received after E. Events A, B and C are the reliable subsequence from source r_1 and are in-order among themselves. Similarly, D and E, the reliable subsequence from source r_2, are correctly ordered as well. As such, given an observed trace and the reliable subsequences composing it, we can calculate a prefix of the trace that is guaranteed to be reliable, such that any potential future out-of-order event will have a timestamp *after* this prefix.

Definition 9 (Reliable Prefix). *Given an observed trace events τ and the set of event sources $\mathcal{R}$, the set of source reliable points $\mathcal{S} = \{s : \cup_{r \in \mathcal{R}} \mathsf{SRP}(\tau, r)\}$. If $\mathcal{S} \cap null = \emptyset$, the reliable prefix event $h \in \mathcal{S}$ is the event such that $\nexists e \in \mathcal{S} : e \prec h$. If h exists, then the* reliable prefix *of τ (referred to as τ_h) is the prefix of τ until and including h.*

In Fig. 2b, the reliable prefix is A, B. After receiving C and re-ordering (Fig. 2c), the reliable prefix is A, B, C. Assume a new event G is received from source r_1 at time θ_6 (i.e., after E), as shown in Fig. 3. At this moment, the reliable prefix is A, B, C, D, E.

Event:	A	B	C	D	E	⋯	G
Data source:	r_1	r_1	r_1	r_2	r_2	⋯	r_1
Timestamp:	θ_1	θ_2	θ_3	θ_4	θ_5		θ_6
							new event

Fig. 3. Obtaining a reliable prefix after observing G.

Note that the proposed method of determining a reliable prefix has its own limitations. For instance, a reliable prefix can be identified only when each source has submitted a non-empty subsequence of events to the stream, meaning that in cases where sources are responsible for a few events, it is likely that such a source will only report events in the later stages of the event stream so that until that point no reliable prefix can be determined. However, event streams report events *across* different process instances and the reliable prefix can be determined from events reported across different instances as well. As such, for an ongoing event stream, we can assume that a past process instance has already been completed and each source has already reported at least one event.

Theorem 1. *Given an observed trace of events τ, its associated reliable prefix τ_h and the stream of events ψ as generated by the actual execution, τ_h is always a prefix of ψ.*

Proof. With ψ as the reference-frame for ordering relations, we prove by contradiction:

1. Assume that τ_h is not the prefix of ψ.
2. There must be an event $e \in \psi$ and $e \notin \tau_h$ as:
   - From the problem description we know that ψ is an ordering of τ. As such, all events in τ_h are also in ψ and their relative order is preserved. Therefore, event e is the only way for which 1) is true.
   - Let h be the last event in τ_h, $h \in \tau$ and $e \prec h$, otherwise if this is not the case τ_h could still be a prefix of ψ.
3. From Definition 9, h is the minimum of all SRPs. As $e \prec h$, we know $e \prec \mathsf{SRP}(\tau, \mathfrak{r}(e))$.
4. If $e \prec \mathsf{SRP}(\tau, \mathfrak{r}(e))$ and both $\mathsf{SRP}(\tau, \mathfrak{r}(e))$ and e are from the same source, then according to Definition 7 e must be in $\tau_{\mathfrak{r}(e)}$.
5. As $\tau_{\mathfrak{r}(e)}$ is a subsequence of τ, then it means that e must be in τ.
6. From 5) and since $e \prec h$, h is the last event of τ_h, and τ_h is a prefix of τ, so $e \in \tau_h$.
7. However 2) and 6) are in contradiction as $e \in \tau_h$ and $e \notin \tau_h$. □

5 Estimating the Out-of-Order Odds

After defining the reliable prefix as the part of an observed trace that is factually in-order, we discuss our approach to quantify the reliability and likelihood of out-of-order events in the remainder of the observed trace.

5.1 Candidates and Base Probabilities

We quantify the reliability and likelihood of out-of-order events by assigning a confidence score to the trace, which is akin to the probability of an observed trace τ to be ψ. The analysis returns a so-called *realities set* containing possible sequence candidates, each having assigned the probability that it matches ψ. Following Definition 8, each source restricted prefix τ_r of τ is an in-order subsequence of τ. As such, each candidate sequence in the realities set must preserve these subsequences. In other words, each candidate in the realities set is a *source restricted supersequence (SRS)* of τ.

Definition 10 (Source Restricted Supersequence). *Given a trace τ and the set of event sources $\mathcal{R}$, a source restricted supersequence $\mathsf{SRS}(\tau)$ of τ is a sequence of events $e_0, e_1, \ldots, e_n$, such that:*

- $\forall e \in \mathsf{SRS}(\tau) : e \in \mathcal{E}$,
- *τ is a subsequence of $\mathsf{SRS}(\tau)$, and*
- *$\forall r \in \mathcal{R}$ providing events appearing in τ, each τ_r is a prefix of $\mathsf{SRS}(\tau)_r$.*

Definition 11 (Realities Set). *A realities set $\mathbb{X}$ contains all sequences of events such that $\forall \mathcal{C} \in \mathbb{X}, \mathcal{C}$ is a possible $\mathsf{SRS}(\tau)$, and the last event in $\mathcal{C}$ equals the last event in τ.*

Note that $|\mathbb{X}|$ is proportional to the number of sources r. For example, if all events in ψ have the same source, then the reliable prefix $\tau_h = \tau$ and, therefore, $\tau = \psi$ and $|\mathbb{X}| = 1$. In the other extreme, if each event in ψ has a unique source, then the reliable prefix τ_h only contains the first event in τ and $\mathbb{X}$ contains all possible permutations of $e \in \mathcal{E}$ starting with $e_0 \in \tau$. For readability purposes, candidates are assumed to not contain repetitions of events. However, when implementing fairness constraints to avoid infinite candidate lengths and cardinality of $\mathbb{X}$, inclusion of repetition is straightforward.

Initially assuming every candidate to be equally probable, the probability $\Pr(\mathcal{C} \mid \tau, \mathcal{P})$ of a candidate $\mathcal{C} \in \mathbb{X}$ to be ψ can simply be calculated as $\frac{1}{|\mathbb{X}|}$. This can be used as a *base probability* for each candidate. While seemingly trivial, the effect of this is that the probability that the observed sequence of events has no missing events is higher when there are fewer ways for things to go wrong, which is directly reflected by a smaller set of possible realities. For instance, fewer event sources in an observed trace typically leads to a longer reliable prefix. Conversely, a higher number of different sources results in a relatively shorter reliable prefix, as a shorter τ_r leads to a higher amount of SRSs. In turn, this results in a higher amount of alternative candidates (i.e. different things that can go wrong), as illustrated in Sect. 6.

5.2 Refining Candidate Probabilities

Having assigned a base probability to each candidate, given a model $\mathcal{P}$, their probabilities can be further refined according to their distance from the observed trace τ, and their distance from the model. The candidate with the highest probability of representing the true behaviour as captured by ψ can be used to identify the impact on the analysis given the observed event and the likely out-of-order events.

Intuitively, the candidate $\mathcal{C} \in \mathbb{X}$ that is closest to both $\mathcal{P}$ and τ is the most likely to be the actual sequence of execution σ as produced in ψ. Considering the set of candidates $\mathbb{X}$, we can refine their probabilities using a function considering their distance from τ to weight their base probability. We refer to this function as a *distance boost function*:

Definition 12 (Distance Boost). *Given a candidate $\mathcal{C}$ and its distance $d(\mathcal{C}, \tau)$ from the observed sequence τ, a distance boost function $\mathrm{b}_1(d(\mathcal{C}, \tau))$ returns a number ≥ 1 and is non-increasing.*

This follows the intuition that candidates farther from the observed trace are less probable. An example of a *distance boost function* is:

$$\mathrm{b}_1(d(\mathcal{C}, \tau)) = 1 + 10 \times (e^{-d(\mathcal{C}, \tau)})$$

The next probability refining function considers the alignment cost between the candidate $\mathcal{C}$ and the model ($\mathcal{P}$). We refer to this function as *alignment boost function*, which is defined as follows:

Definition 13 (Alignment Boost). *Given a candidate $\mathcal{C}$, its alignment cost $\alpha(\mathcal{C}, \mathcal{P})$ and some parameter β capturing the reliability of the model when considering τ, an alignment boost function $\mathrm{b}_2(\alpha(\mathcal{C}, \mathcal{P}), \beta)$ returns a number ≥ 1 and is non-increasing with either of its input parameters.*

Intuitively, this improves the probability of a candidate that is close to the given model when the model is reliable. An example of an *alignment boost function* is:

$$\mathrm{b}_2(\alpha(\mathcal{C}, \mathcal{P}), \beta) = 1 + \beta \times 10 \times (e^{-\alpha(\mathcal{C}, \mathcal{P})})$$

Putting together the introduced boost functions, we obtain the probability weight of a candidate $\mathcal{C} \in \mathbb{X}$ representing ψ:

$$\omega(\mathcal{C} \mid \tau, \mathcal{P}) = \frac{1}{|\mathbb{X}|} \times \mathrm{b}_1(d(\mathcal{C}, \tau)) \times \mathrm{b}_2(\alpha(\mathcal{C}, \mathcal{P}), \beta) \tag{1}$$

The boosted weights of each candidate $\mathcal{C} \in \mathbb{X}$ can then be normalised as a probability of $\mathcal{C}$ representing ψ as follows:

$$\Pr(\mathcal{C} \mid \tau, \mathcal{P}) = \frac{\omega(\mathcal{C} \mid \tau, \mathcal{P})}{\sum\limits_{\mathcal{C}' \in \mathbb{X}} \omega(\mathcal{C}' \mid \tau, \mathcal{P})} \tag{2}$$

5.3 Confidence of Downstream Compliance Analysis

The confidence score of the downstream analysis can be estimated using these candidate probabilities. For example, given a compliance rule rl that has been evaluated over the current observation, we can estimate the confidence using the total probabilities of candidates who do not invalidate the current evaluation. Given a function eval, which returns a boolean indicating whether a rule is satified by a sequence of events, the confidence score CS can be computed as follows:

$$\mathrm{CS}(\mathrm{eval}(rl, \tau)) = \sum_{\mathcal{C} \in \mathbb{X}} \Pr(\mathcal{C} \mid \tau, \mathcal{P}) \times \mathrm{eval}(rl, \mathcal{C}) == \mathrm{eval}(rl, \tau) \tag{3}$$

Note that for some rule logics, the data associated with the events may affect the evaluation. In this case, candidates that contain events not in the observation the event sequence would contain missing data representing what happened when those events were executed. As such, eval cannot necessarily determine the outcome for this input and would return a third value, $\mathrm{unknown}$. The confidence score can then be broken down into two values, a total of probabilities for candidates that *do* agree with the current evaluation, and a total for candidates that *may* agree with it.

5.4 Computational Complexity and Optimisations

While identifying the reliable history is a relatively simple procedure, it should be noted that it can be computationally intensive to calculate the out-of-order odds. The worst-case size of $\mathbb{X}$ occurs when there is one event in τ and all other events are possibly missing. The possibilities are then given by all permutations of the power-set of these remaining events, each appended with the known event, which is combinatorial to the number of events in $\mathcal{E}$. We iterate over these candidates to calculate the boosters. While calculating the distance-boost is relatively simple, the alignment-boost can involve calculating the cost function for all possible executions in $\mathcal{P}$ (if a brute-force approach is used), which in the worst case is a flower model. This is again combinatorial to $|\mathcal{E}|$. We also iterate over the candidates to evaluate them against the compliance rules, whose complexity depends on the specific logic used, but should be relatively simple.

Although this worst-case scenario is unrealistic, it is worth highlighting the areas for optimisation that can be considered in scenarios that have features resembling the worst-case. The main concerning factor is the size of $\mathbb{X}$. Each candidate needs to be considered individually to evaluate the confidence of a compliance rule, as the rule may contain temporal logic such that even very similar candidates will result in different evaluations. However, heuristics could be used so that we only consider candidates that are more probable. Nonetheless, the reality set is typically much smaller than the worst-case as the observed events and source restriction would prohibit many of the permutations. The other potentially combinatorial factor arises from the alignment-boost, which can be improved using an approximation for the optimal alignment (e.g., [5]). Furthermore, this procedure is not necessarily required for every new event received. For example, it could be done only when new events are received from sources who have not been heard from for a while, as these instances create significant changes to the possible SRSs. Such an optimisation should be considered if the event stream has a high frequency.

6 Demonstration

6.1 Dataset and Experiment Setup

To demonstrate the proposed approaches, we generated a multi-instance event stream from the Petri net in Fig. 1, simulating out-of-order events. We added noise by creating 5 variants of the Petri net, to generate executions that increasingly deviate from the known model. In total, 840 traces were generated, resulting in a log size of 3,152 events.

Next, we generated *event executed* timestamps for the events in these traces. The start time of each trace was randomly chosen between the start of 2022 and the end of 2023. As such, around 1 process instance starts each day. The events in a process instance were kept in order, with a randomly defined interval between each event of $[0, 2]$ days. Consequently, each trace takes an average total of 5.6 ± 1.6 days to execute, with around 6 traces on average in process at any one time. Based on the *event executed* timestamps and the different sources for each event type, we subsequently generated *event received* timestamps. The results are presented for two assignments of event sources:

- 2 sources: r_1 for A, B, C and G; r_2 for D, E and F.
- 7 sources: each event comes from a unique source.

The *event received* times were obtained by adding a randomly generated latency time to the *event executed* time for each unique source within $[0, 6]$ hours. We then generated 50 *source outages*, where a randomly selected source would be radio silent for $[0, 5]$ days, after which the source would immediately send its accumulated events while maintaining their relative order. Any events during this period would have an *event received* time corresponding to the end of this outage. This results in a realistic simulation of the occurrences of out-of-order events, caused by distributed sources. Finally, we sorted the log by *event received* and fed the events one-by-one to the monitor.[1]

6.2 Memory Footprint Results

To demonstrate the effect of the reliable prefix approach, we plotted the number of events in the unreliable portion of the stream, at each step in the log, against the total amount of events observed (see Fig. 4). This would be the number of events whose data would need to be retained in memory. In comparison, without using our approach all events received must be stored in memory as there is always the possibility of an out-of-order, and so the memory required would steadily grow to 3,152 events. When being less conservative and assuming that a process instance can be discarded when receiving the final event (e.g., "Case closed"), on average around $\frac{1}{2} \times 6 \times 7 = 21$ events would be in memory *at all times*. However, the event stream has a number of occurrences where out-of-order events arrive after the final event[2] where this approach would not have

[1] The dataset is accessible at: https://github.com/hannahmburke/outoforder_eventstream_dataset.

[2] For 2 and 7 sources, 31 and 37 events arrive after their trace's final event, respectively.

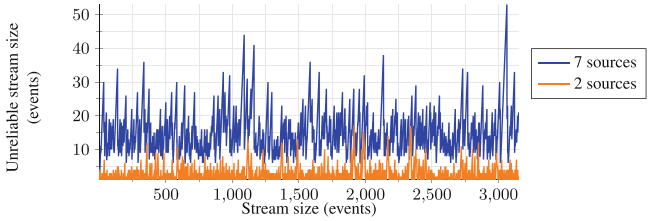


Fig. 4. Memory usage with the reliable prefix approach.

the required history to backtrack. Nevertheless, our approach always has the history required for potential out-of-order events, with generally a smaller memory footprint.

Our approach significant reduces required in-memory events at any time and, therefore, overall memory footprint. The spikes in Fig. 4 are associated to the *source outages*, where the size of the unreliable portion grows until the communication is recovered, after which it drops back down. As expected, a network distributed across many sources results in a higher overall memory usage as the reliable prefix is typically shorter.

6.3 Probabilities

Section 5 presented an approach for calculating the probabilities of possible candidates $\mathcal{C}$ that may be the actual event stream ψ in order to calculate a confidence score of the downstream analysis. It is the responsibility of the downstream analysis to calculate its confidence based on the logic it performed, so that cannot be evaluated here. However, for an insight into the accuracy of the probabilities themselves, this section investigates the accuracy of the probability of no missing out-of-order events, i.e. $Pr(\tau = \psi)$.

At each step in the event stream, we filtered for each instantiated trace. Next, we computed each SRS and calculated its probability. We used $\beta = 1$ as initial value and defined $distance(\mathcal{C}, \tau)$ as the number of additional events in $\mathcal{C}$. For each case, we compared the probability $Pr(\tau = \psi)$ to whether there were actually no missing events for τ at the given step. We collated the cases with possible missing events (i.e., the current trace has some events in its unreliable portion and not all events from $\mathcal{P}$ have been observed in the trace). The overall accuracy of this probability is visualised by a Receiver Operating Characteristic (ROC) curve in Fig. 5, where the further the curve is above the diagonal, the better the performance in predicting missing events.

For the scenario with 2 unique event sources, there are a total of 6,703 cases where a trace may be missing an event. As the reliable prefix is longer with only 2 sources, there are fewer possibilities for missing events. Of these, 430 cases indeed contain missing events. Similarly, with 7 unique event sources, there are a total of 21,409 cases where a trace may be missing an event. Of these, 395 cases indeed contain missing events.

It is clearly shown that both the base and boosted SRS probabilities outperform a random classifier. Intuitively, this means that when the approach returns a high probability of a missing event, there is indeed a significant likelihood that the trace is missing an event, as opposed to when it returns a low probability. The base SRS probabilities (where the probability is defined by the cardinality of $\mathbb{X}$) show a distinct improve-

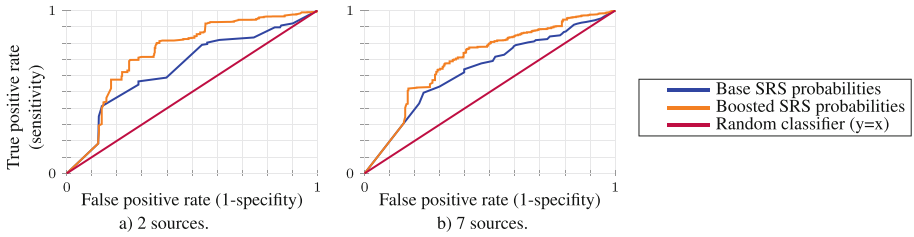


Fig. 5. ROC curves for Pr($\tau = \psi$) for 2 and 7 sources.

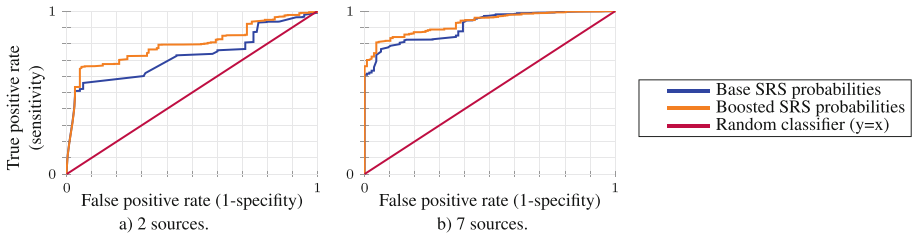


Fig. 6. ROC curves for Conf(eval(rl, τ)) as a predictor for eval(rl, τ) == eval(rl, ψ).

ment on the random classifier. This is further improved by adding the boosts related to observation distance and alignment cost (which particularly improves performance when higher sensitivity is desired), supporting the intuition discussed in Sect. 5. While this approach is still far from a perfect classifier, its benefit is clearly visible with the limited information available, even when the network is more distributed.

6.4 Downstream Confidence

We evaluated the proposed metric for the confidence score of the compliance analysis by applying the rule in Sect. 3 to the observed trace and compared it to the real event stream at each time-step. Using the same event stream as above, we took the candidate probabilities found earlier to calculate the confidence score with Eq. 3.

An incorrect rule evaluation consists of two cases: (i) when the observation leads to a violation while the actual event stream (until the current time-step) does not, and (ii) when the observation does not lead to a violation while the actual event stream does. As above, for the scenario with 2 unique event sources, there are a total of 6,703 cases where a trace may be missing an event. Of these, 153 contain an incorrect rule evaluation. When there are 7 unique event sources, there are a total of 21,409 cases where a trace may be missing an event. Of these, 107 contain an incorrect rule evaluation.

The sensitivity and specificity of the confidence score is shown in Fig. 6. Overall we can observe low confidence scores are associated with incorrect rule evaluations, while high confidence scores are associated with correct rule evaluations. Compared to Fig. 5 there is a higher accuracy, as we can often be more decisive about the given rule depending on specifically which events can be missing (based on SRS). Again the boosted SRS probabilities show an improvement compared to the base.

7 Related Work

Out-of-order events have been identified as a problem in various relevant areas, such as process discovery [6], conformance checking [7], event processing in IoT scenarios [8] and specifically for online compliance checking of business processes [1]. In [1], an online compliance monitor for event streams is presented, which uses distributed computing for more efficient analysis. However, in the version of the approach that can perform temporal reasoning with out-of-order events, violations are not conclusively detected until the event stream is closed, making it an offline monitor.

Out-of-order event arrivals require an extensive memory footprint, as the event stream needs to be reevaluated with the corrected event order. Raun et al. [5] present an event stream conformance checker based on prefix alignment, with an efficiency that is favourable to online monitoring. In [7], the approach is evaluated for handling out-of-order events while minimising the memory footprint. This uses a decay time to discard old states from memory once potential out-of-order events would not require these states for re-evaluation. Although the decay time approach is similar to the reliable prefix method presented here, it is statistical and therefore cannot guarantee the absence of out-of-orders before the cut-off. Similarly, in a different context, the probabilistic approach in [9] defines windows where late arrivals are expected, aiming to minimise both missing events (late arrivals after the window is closed) and latency in closing the window (which means postponing the evaluation). From the perspective of compliance monitoring, it is essential that events are used in the evaluation regardless of how late they arrive. Therefore, it requires a method that guarantees the necessary data is maintained in case of late arrivals.

Furthermore, the approach in [5] does not consider how the possibility of future out-of-order events affects the certainty of the current outcome of the online checker. This may not be necessary for the specific application, but in the context of compliance, reliability and confidence of the results are essential. Conversely, for the purpose of conformance checking over incomplete event logs (involving missing data), D'Iddio et al. [10] present an approach that models this uncertainty to identify when an execution can be certified as correct or anomalous. While they do not quantify its confidence or consider missing events themselves, it emphasises the importance of measuring uncertainty. Within the context of runtime verification, a temporal logic has been defined in [11] that can account for the uncertainty resulting in potential missing events. The outcome of propositions has three possible values (true, false and unknown) and these feed into the overall evaluation. While the approach in our paper uses probabilities to estimate the likelihood of the different outcomes, features of this logic could be incorporated when assessing how each candidate affects the compliance outcome.

More broadly, four categories have been identified amongst techniques to handle out-of-order data in the field of event processing [12]. These are buffering (delaying processing new elements by a given period), punctuation (waiting for a notification to indicate all past elements have arrived), speculation (proceeding with the assumption of no out-of-order elements, and revising when late arrivals appear) and approximation (providing rough / overall results that for large enough windows are not affected by some misordering). Many of these approaches require an architectural change to implement, such as e.g. the middleware solution presented in [8] or the timer-driven

punctuation method presented in [13]. In the context of process discovery, [6] present both buffering and speculation approaches.

Our approach is a balance between punctuation and speculation. The punctuation component is achieved implicitly, as we can be certain that all past events have arrived up until the minimum time of the most recent events across all data sources. The monitor then uses a speculative approach for the remaining time period, providing compliance results as soon as possible, and backtrack if needed. This is done using *conformance checking* against a model of expected behaviour to estimate the probability of future out-of-orders, to subsequently quantify the confidence of the current *compliance* results.

8 Discussion and Conclusion

In this paper, we presented an approach comprising two techniques to deal with the out-of-order problem in runtime compliance monitoring, where the events in an event stream are not guaranteed to be received in the correct order.

The first technique is referred to as *reliable prefix* and aims at identifying a prefix of the live event stream received by a monitor that is guaranteed to be currently in order and not in-deceptive-order. The reliable prefix allows the monitor to perform its analysis confidently on at least a portion of the live event stream, and memory of that portion can subsequently be released. Analysis results that include the remainder of the stream can be treated differently (i.e., tentatively). If an out-of-order arrives within this portion, the monitor is required to backtrack up to the end of the reliable prefix only.

The second technique is *probabilities of possible realities*, which aims at analysing the live event stream to quantify its reliability and identify where the observation is missing events that will eventually arrive out-of-order. This, in turn, can be used by the monitor to calculate a confidence score of its compliance results, depending on how the possible out-of-order events may affect the outcome. In practice, when the monitor returns a non-compliance result that requires action, but the confidence of this result is low, the decision-maker can choose a more measured or delayed response.

The approach was evaluated on an artificial data set simulating out-of-order events over a multi-instance event stream with varying deviation from the model. The evaluation showed that the presented approach significantly reduces memory footprint for runtime monitoring systems. Furthermore, the evaluation showed that the approach of calculating a confidence score provided an accurate indication for when the compliance outcome is incorrect due to future out-of-order events.

While the approach requires a few assumptions about the process and distributed system, these assumptions have been well-justified. Future work could further investigate the feasibility of this approach on a real-world scenario. With this, boost functions could be further refined, such as by including other variables as inputs (e.g., timestamps, network reliability, event data). The approach should also be evaluated on varying model fitness and precision, to understand how the reliability of the process model influences the results. Additionally, the approach only indirectly incorporates the reliability of the model and we expect that further enhancement to derive reliability of the model from the observed reliable prefix can improve results. Similarly, network reliability could be measured, to indicate how observation distance should be penalised.

Furthermore, while our current method does not use machine learning and, therefore, does not require historic data, in the case where historic training data exists, machine learning could be a useful tool to extract additional knowledge given the vast array of factors that may cause out-of-order events. Finally, for (very) dense inter-arrival rates between events, the approach relies on the performance of the alignment computations, and we plan to investigate the upper-bounds in future work.

References

1. Loreti, D., Chesani, F., Ciampolini, A., Mello, P.: A distributed approach to compliance monitoring of business process event streams. Futur. Gener. Comput. Syst. **82**, 104–118 (2018)
2. van Beest, N.R.T.P., Groefsema, H., Cryer, A., Governatori, G., Colombo Tosatto, S., Burke, H.: Cross-instance regulatory compliance checking of business process event logs. IEEE Trans. Softw. Eng. **49**(11), 4917–4931 (2023). https://doi.org/10.1109/TSE.2023.3319086
3. van der Aalst, W.M.P.: The application of Petri nets to workflow management. J. Circ. Syst. Comput. **8**(1), 21–66 (1998)
4. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rev. Data Min. Knowl. Disc. **2**(2), 182–192 (2012)
5. Raun, K., Tommasini, R., Awad, A.: I will survive: an event-driven conformance checking approach over process streams. In: International Conference on Distributed and Event-Based Systems, pp. 49–60 (2023)
6. Awad, A., Weidlich, M., Sakr, S.: Process mining over unordered event streams. In: International Conference on Process Mining (ICPM), pp. 81–88 (2020)
7. Raun, K., Tommasini, R., Awad, A.: Adaptive handling of out-of-order streams in conformance checking. In: DOLAP, pp. 9–17 (2024)
8. Mutschler, C., Philippsen, M.: Adaptive speculative processing of out-of-order event streams. ACM Trans. Internet Technol. **14**(1), 1–24 (2014)
9. Rivetti, N., Zacheilas, N., Gal, A., Kalogeraki, V.: Probabilistic management of late arrival of events. In: International Conference on Distributed and Event-based Systems, pp. 52–63 (2018)
10. D'Iddio, A.C., Schunck, C.H., Arcieri, F., Talamo, M.: Online and offline conformance checking of inter-organizational business processes with incomplete process logs. In: IEEE International Carnahan Conference on Security Technology (ICCST), pp. 1–8 (2016)
11. Basin, D., Klaedtke, F., Zălinescu, E.: Runtime verification over out-of-order streams. ACM Trans. Comput. Logic **21**(1), 1–43 (2020). https://doi.org/10.1145/3355609
12. Dayarathna, M., Perera, S.: Recent advancements in event processing. ACM Comput. Surv. **51**(2), 1–36 (2019). https://doi.org/10.1145/3170432
13. Li, J., Tufte, K., Shkapenyuk, V., Papadimos, V., Johnson, T., Maier, D.: Out-of-order processing: a new architecture for high-performance stream systems. Proc. VLDB Endowment **1**(1), 274–288 (2008). https://doi.org/10.14778/1453856.1453890

Translucent Alignments

Harry H. Beyel(✉), Christopher T. Schwanen, and Wil M. P. van der Aalst

Chair of Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany
{beyel,schwanen,wvdaalst}@pads.rwth-aachen.de

Abstract. Event logs, the primary data source in process mining, consist of events with a case identifier, an activity, and a timestamp. Translucent event logs extend this by also recording enabled activities alongside executed ones. Such data can be extracted from user interactions or running information systems. Fitness, a quality measure in conformance checking, assesses how well an event log aligns with a process model. While fitness has been widely studied, no prior work has considered enabled activities. We introduce the first fitness measurement incorporating this information based on the well-established alignment technique. Our generalized, parameterized approach also supports traditional fitness measurement. Through qualitative evaluation, we demonstrate our method's applicability and provide insights into the implications of parameters. Our quantitative assessment shows limited computational overhead when enabled activities are considered and provides insights into the fitness score of the classical and translucent approaches. Our results indicate that incorporating enabled activities yields appropriate fitness scores compared to traditional methods, enhancing conformance checking accuracy.

Keywords: Conformance Checking · Alignments · Fitness · Translucent Event Logs · Enabled Activities

1 Introduction

Processes are executed in various organizations and can be digitally tracked and stored in an *event log*. An event log consists of *events*, each having information on case, activity, and timestamp. Beyond these basics, additional data, like the resource executing an event, can be recorded. Moreover, it is possible to record information on *enabled* activities—besides the executed activity. Such information can be added due to domain knowledge or extraction of activities executed in a desktop environment [11,13]. Note that extracting these logs from desktop environments provides valuable insights for analyzing human-computer interaction. In addition, when creating such logs from a Graphical User Interface (GUI), they provide a great source for robotic process mining [21] and can be used to create bots for robotic process automation [41]. We call event logs that contain information on enabled activities *translucent event logs*.

We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 167–184, 2025.
https://doi.org/10.1007/978-3-031-94571-7_10

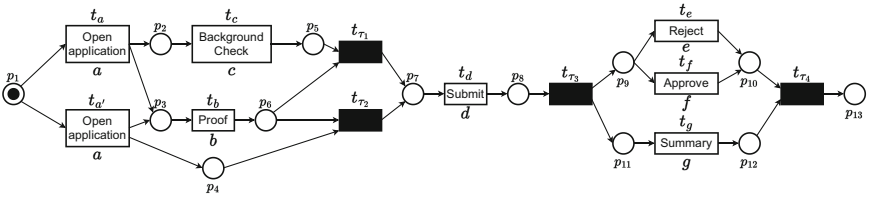
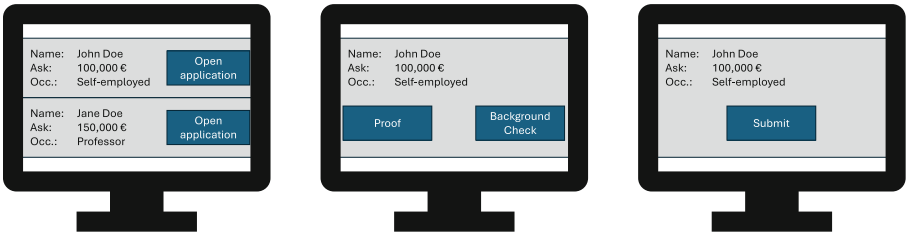


Fig. 1. Example workflow captured within a Petri net.



(a) Open an application. (b) Proof and check. (c) Submit.

Fig. 2. Example GUI for proofing a loan application.

Process mining can be used to analyze event logs to get insights into the execution of processes [1]. Dedicated process-mining techniques can be applied to translucent event logs [2,12,15]. One area in process mining is conformance checking, which consists of the fitness quality dimension [18]. Fitness measures how well a process model represents the behavior recorded in an event log. One method to assess fitness is token-based replay [38], which simulates the flow of tokens through the model's transitions according to the observed events. Discrepancies are identified by tracking token movements: missing or remaining tokens indicate deviations between the model and actual execution. However, token-based replay has two main issues: it is non-deterministic when the model contains duplicate tasks, and it can produce misleading diagnostics due to "token flooding". The state-of-the-art technique, alignments [5], overcomes these problems by mapping each event in the log to the corresponding model activities. It finds the optimal sequence of synchronous and asynchronous moves to accurately detect discrepancies and measure conformance.

When computing standard alignments, only executed activities of an event are considered, not the enabled activities. This can lead to false conclusions. To illustrate this, consider a simplified loan application process. In the first step, the application is opened. Afterward, there are two possible actions: either only a proof is conducted, or a proof and a background check are conducted. Subsequently, the results are submitted. In the end, the application is rejected or approved, and a summary is created. The workflow is depicted in Fig. 1. A possible application process handling with a GUI is depicted in Fig. 2 and relates to the former description. Assume the sequence $\langle a, b, d, e, g \rangle$ of activities

has been executed and is linked to the scenario shown in Fig. 2. When measuring fitness between this sequence and the model, the result is a perfect score of 1.0. The sequence of executed transitions is $\langle t_{a'}, t_b, t_{\tau_2}, t_d, t_{\tau_3}, t_e, t_g, t_{\tau_4} \rangle$. However, as denoted in Fig. 2b, it is also possible to perform a background check, resulting in the activity being enabled. But when executing $t_{a'}$, b and c are not enabled, thus not fitting the information available on enabled activities extracted from the GUI example. Thus, the score should not be 1.0 but lower.

The information on a mismatch of enabled activities is valuable when evaluating which process model should be used to create a bot for automating tasks in a GUI. The bot may misbehave if a wrong model is chosen based on misleading scores. Moreover, we can verify a system's implementation by utilizing information on enabled activities. Usually, a reference model exists that describes how a system should behave. An implementation should follow this model. By extracting enabled activities from screenshots during the interaction with the software, we can align the recorded behavior with the reference model. When the recording and extraction of activities from screenshots work as intended and a translucent event log is created, translucent alignments can spot context-aware deviations. This provides new insights into system verification, enabling large-scale verification across different user groups and systems. In addition, the information on enabled activities can guide alignments and increase their accuracy. Although a traditional alignment represents an optimal path through a model for a given sequence of activities, there might be multiple optimal paths. Techniques that use alignments as input, for example, the precision method presented in [6], are based on this and are not deterministic, leading to potentially different results in each run. To illustrate this, consider the trace $\langle a, b, d, e, e \rangle$. We assume that for the first e, f and g are also enabled. For the second e, we assume that f is also enabled. Within the classic alignment, a model and log move would be created to execute g, but their position is non-deterministic. However, since we know from the translucent event log that g was possible but not performed, we can create a better alignment using the enabled activity to guide the alignment computation.

In this work, we develop an approach that adjusts the fitness score by considering information on enabled activities, penalizing discrepancies. Our method allows for higher fitness when enabled activities match but executed activities differ, and enables verification of system requirements using enabled activities. At the same time, our approach acts as a generalization of the existing approach.

2 Related Work

Much work has been done to measure fitness in process mining. Besides the basic idea of footprint comparison between an event log and a process model, advanced techniques were developed. One of them is the aforementioned token-based replay [38]. The idea of token-based replay is to replay a trace in a Petri net. In this process, missing tokens might be added to allow the firing of otherwise impeded transitions. The number of tokens added in this way and the

number of tokens consumed, produced, and remained after finishing a trace are tracked. Considering the different amounts and all traces, a fitness score is computed. This approach has also been adjusted for an online setting [16]. Despite recent advances [8,9], which solve issues of token-based replay, alignments are still recognized as the state-of-the-art technique.

Alignments, presented in [5], use more advanced concepts than token-based replay. For each trace of an event log, a trace net based on Petri nets is created. Between the process model, which is given as a Petri net, and the trace net, a synchronous product net is created by following predefined rules. The firing of a transition in a synchronous product net is associated with costs. The goal is to find the cheapest sequence from the initial marking to the final marking of the synchronous product net. Over the last years, advances for the consuming computation of an alignment have been proposed [19,20,29,35–37,39,40,42,43]. Some extensions consider information beyond the control flow, e.g., data-aware alignments [27,28,34] or resource-aware alignments [7].

In [17], weighted artificial negative events are presented. Negative events represent information about activities prevented from taking place in a process. These are rarely logged in real-life event logs. The generation algorithm induces these negative events artificially by examining each position in each trace within an event log and determining which activities cannot occur at those positions based on the observed behavior. This contrasts our method since we use a source available in the data and the model to compute our score.

There are also techniques for measuring fitness in a stochastic setting. Stochastic fitness measurements were developed in [24,25,33]. In addition, techniques for object-centric event data were developed. A general approach is presented in [4], while alignments for an object-centric setting are presented in [30].

Concerning the usage of translucent event logs in process mining, recent work has shown that the information that translucent event logs convey can improve existing process-discovery techniques [12]. Concerning conformance-checking techniques using translucent event logs, there exists work in measuring precision by considering enabled activities [15]. However, no approach measures fitness between a Petri net and a translucent event log so far.

3 Preliminaries

In this section, we define the basic concepts that we use in the remainder of this work. We define sets, sequences, and tuples and operations on them. Furthermore, we define translucent event logs and traces. Finally, we define Petri nets and related concepts.

Given a set X and a function f, $f(X) = \{f(x) \,|\, x \in X\}$ denotes applying the function f on all elements of set X. For simplicity, we denote for any set $X^{\gg} = X \cup \{\gg\}$.

Definition 1 (Tversky Index). Given sets A and B, the Tversky index provides an asymmetric similarity measurement between sets defined as

$$S_{\alpha,\beta}(A,B) = \frac{|A \cap B|}{|A \cap B| + \alpha \cdot |A \setminus B| + \beta \cdot |B \setminus A|},$$

with $\alpha, \beta \geq 0$. When α is high relative to β, the index becomes more sensitive to elements in A that are not in B, highlighting the uniqueness of A. A similar situation applies to β. If $\alpha = \beta = 1$, this corresponds to the (symmetric) Jaccard index. We write $D_{\alpha,\beta}(A,B) = 1 - S_{\alpha,\beta}(A,B)$ to denote the Tversky distance.

$\mathcal{B}(X)$ denotes the set of all multisets over set X. E.g., if $X = \{x, y, z\}$, an example multiset is $[x, x, y] = [x^2, y]$. Let A be a set and let $t = (a_1, \ldots, a_n) \in A \times \cdots \times A$ be a tuple of n elements. $\pi_i(t)$ refers to the ith element of the tuple, e.g., $\pi_1((a,b)) = a$. Given a set X, $\sigma = \langle \sigma_1, \ldots, \sigma_n \rangle \in X^*$ denotes a sequence over X. σ_i denotes the sequence's ith element. The length of a sequence σ is denoted as $|\sigma|$. The concatenation of two sequences is denoted as $\sigma \cdot \sigma'$. Given a sequence σ and a set X', $\sigma{\restriction_{X'}}$ denotes a sequence projection, e.g., $\langle a,b,c,d \rangle {\restriction_{\{a,c\}}} = \langle a, c \rangle$. $pref_i(\sigma) = \langle \sigma_1, \ldots, \sigma_i \rangle$ refers to the prefix of a sequence containing the first i elements. $pref_0 = \langle\rangle$. For any sequence of tuples $\sigma \in (A \times \cdots \times A)^*$, we define $\pi_i^*(\sigma) = \langle \pi_i(\sigma_1), \ldots, \pi_i(\sigma_{|\sigma|}) \rangle$.

Translucent event logs capture information on enabled activities in addition to executed ones. Hence, the executed activity must also be enabled in the corresponding event. Also, we assume that all enabled activities in an event log are performed at some point. $\mathcal{U}_{case}$ is the universe of case identifiers, $\mathcal{U}_{act}$ is the universe of activity names, and $\mathcal{U}_{time}$ is the universe of timestamps.

Definition 2 (Translucent Event Log, Trace). $\mathcal{U}_{ev}$ is the universe of events. $e \in \mathcal{U}_{ev}$ is an event, $\pi_{case}(e) \in \mathcal{U}_{case}$ is the case of e, $\pi_{time}(e) \in \mathcal{U}_{time}$ is the time of e, $\pi_{en}(e) \subseteq \mathcal{U}_{act}$ are the enabled activities of e, $\pi_{act}(e) \in \pi_{en}(e)$ is the activity of e. A translucent event log L is a set of events $L \subseteq \mathcal{U}_{ev}$. In addition, $\bigcup_{e \in L} \pi_{en}(e) = \bigcup_{e \in L} \{\pi_{act}(e)\}$. For simplicity, we assume that events in L are totally ordered such that for $e_1, e_2 \in L$, $e_1 < e_2$ implies $\pi_{time}(e_1) \leq \pi_{time}(e_2)$. A trace is a sequence of all events of a case ordered from earliest to latest, i.e., $\sigma^{L,c} = \langle e_1, \ldots, e_n \rangle$, such that for $c \in \pi_{case}(L)$, $\{e_1, \ldots, e_n\} = \{e \in L \mid \pi_{case}(e) = c\}$ and $e_1 < \ldots < e_n$. The set of traces of L is denoted as $\Sigma^L = \{\sigma^{L,c} \mid c \in \pi_{case}(L)\}$.

An example translucent event log is showcased in Table 1. For e_1, $\pi_{case}(e_1) = 1$, $\pi_{act}(e_1) = a$, $\pi_{en}(e_1) = \{a\}$, and $\pi_{time}(e_1) =$ 2024-06-20 13:37:37. In the remainder of the paper, for simplicity, we denote the enabled activities and underline the executed activity of an event. For example, we can denote for the first trace in Table 1, $\sigma^{L,1} = \langle e_1, e_2, e_3, e_4, e_5 \rangle$, with $\langle \underline{a}, \underline{b}c, \underline{d}, \underline{e}fg, \underline{g} \rangle$.

Definition 3 (Petri Net). A Petri net is a tuple $N = (P, T, F, A, l)$, where P is a set of places, T is a set of transitions such that $P \cap T = \emptyset$, and $F \subseteq (T \times P) \cup (P \times T)$ is a set of directed arcs. $A \subseteq \mathcal{U}_{act} \cup \{\tau\}$ is a set of activity labels, and $l \colon T \to A$ is a labeling function where τ denotes the silent activity.

Table 1. Exemplary translucent event log.

Event	Case	Activity	Enabled Activities	Timestamp
e_1	1	a	{a}	2024-06-20 13:37:37
e_2	1	b	{b, c}	2024-06-20 13:37:38
e_3	1	d	{d}	2024-06-20 13:37:39
e_4	1	e	{e, f, g}	2024-06-20 13:37:40
e_5	1	g	{g}	2024-06-20 13:37:41
e_{11}	2	a	{a}	2024-06-20 13:37:51
e_{12}	2	c	{c}	2024-06-20 13:37:52
e_{13}	2	d	{d}	2024-06-20 13:37:53
e_{14}	2	x	{x, e, g}	2024-06-20 13:37:54
e_{15}	2	g	{g}	2024-06-20 13:37:55

$M \in \mathcal{B}(P)$ is a marking and (N, M) refers to the Petri net N in marking M. We define $T_\tau = \{t \in T \mid l(t) = \tau\}$.

We focus on sound workflow nets [3]. Furthermore, for computational reasons, we assume that each transition that is connected to the sink place is a non-silent transition. To ensure that, we introduce a new, non-silent transition with a unique label and a new sink place. This transition, labeled *end*, has an ingoing arc from the old sink place and an outgoing arc to the new sink place. Also, we assume that an artificial end event e_{end} is added at the end of each trace, with $\pi_{act}(e_{end}) = end$ and $\pi_{en}(e_{end}) = \{end\}$. Given the Petri net in Fig. 1, $P = \{p_1, p_2, \dots\}$, $T = \{t_a, t_{a'}, t_b, t_c, t_{\tau_1}, t_{\tau_2}, \dots\}$, $F = \{(p_1, t_a), (p_1, t_{a'}), \dots, (t_{\tau_2}, p_7)\}$ and $A = \{a, b, c, \dots, \tau\}$. The shown Petri net is in marking $[p_1]$. Petri net firing rules can be found in [1].

Definition 4 (Firing Transition and Sequence). Let $N = (P, T, F, A, l)$ be a Petri net. Let $M, M' \in \mathcal{B}(P)$ be two markings. The firing of a single transition $t \in T$ is denoted as $(N, M) \xrightarrow{t} (N, M')$. The successive firing of all transitions in $\sigma \in T^*$ is denoted as $(N, M) \xrightarrow{\sigma} (N, M')$. In addition, $(N, M) \xrightarrow{\langle\rangle} (N, M)$. Let $M_i \in \mathcal{B}(P)$ be the initial marking. We define a marking $M' \in \mathcal{B}(P)$ as reachable in (N, M_i), if there exists $\sigma \in T^*$ such that $(N, M_i) \xrightarrow{\sigma} (N, M')$. The set of all reachable markings starting in (N, M_i) is denoted as $[N, M_i\rangle$.

4 Translucent Fitness

In the classic alignment, moves on the model and log are compared. A cost function assigns costs to the different move types, and the goal is to compute the alignment with the lowest costs while still meeting defined requirements. The costs are used to determine the fitness score. In this section, we show how translucent alignments and a translucent fitness score are computed. We first define translucent alignments and the corresponding moves and costs. We provide

a high-level overview to convey the idea. Second, we show how an optimal translucent alignment is computed. Finally, we show how a fitness score is computed on the trace and log level.

4.1 Defining Translucent Alignments

Moves on the log and model side are essential for alignments. An alignment is computed for each trace of the translucent event log. There are two key requirements for an alignment: first, the sequence of events must yield the original trace; second, the sequence of transitions should represent a firing sequence from a Petri net's initial to final marking. The following definition captures this.

Definition 5 (Translucent Alignment). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, and $N = (P, T, F, A, l)$ be a Petri net with its initial marking $M_i \in \mathcal{B}(P)$ and final marking $M_f \in \mathcal{B}(P)$. An alignment $\gamma \in (L^{\gg} \times T^{\gg})^*$ between σ and N is a sequence such that:

- $\pi_1^*(\gamma)\restriction_L = \sigma$
- $(N, M_i) \xrightarrow{\pi_2^*(\gamma)\restriction_T} (N, M_f)$

For illustration, the following is an alignment between the first case of the event log shown in Table 1 and the Petri net depicted in Fig. 1:

$$\gamma = \begin{array}{|c|c|c|c|c|c|c|c|} \hline e_1 & e_2 & \gg & e_3 & \gg & e_4 & e_5 & \gg \\ \hline \underline{a} & \underline{bc} & \gg & \underline{d} & \gg & \underline{efg} & \underline{g} & \gg \\ \hline \underline{a} & \underline{b} & \tau & \underline{d} & \tau & \underline{efg} & \underline{g} & \tau \\ \hline t_{a'} & t_b & t_{\tau_2} & t_d & t_{\tau_3} & t_e & t_g & t_{\tau_4} \\ \hline \end{array}.$$

Given this alignment, $\gamma = \langle (e_1, t_{a'}), (e_2, t_b), \ldots, (e_5, t_g), (\gg, t_{\tau_4}) \rangle$, we denote $\pi_1^*(\gamma)\restriction_L = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ and $\pi_2^*(\gamma)\restriction_T = \langle t_{a'}, t_b, t_{\tau_2}, t_d, t_{\tau_3}, t_e, t_g, t_{\tau_4} \rangle$. For our work, information on enabled activities is necessary. While events in a translucent event log have this information, we also need to access this information in a model. Note that silent transitions do not contribute useful information since their execution is not recorded. As they serve as synchronous points in the model, we should seek enabled activities beyond them when possible.

Definition 6 (Enabled Activities in a Model). Let $N = (P, T, F, A, l)$ be a Petri net and $M_i \in \mathcal{B}(P)$ the initial marking. For a marking $M \in [N, M_i\rangle$, we define its set of enabled activities as $en_M^N(M) = \{l(t) \mid \exists_{\sigma \in T_\tau^*, t \in T \setminus T_\tau, M' \in \mathcal{B}(P)} (N, M) \xrightarrow{\sigma \cdot \langle t \rangle} (N, M')\}$. Given a sequence $\sigma \in T^*$, we define $en_\sigma^N(\sigma) = en_M^N(M'')$ if $(N, M_i) \xrightarrow{\sigma} (N, M'')$.

Given the Petri net in Fig. 1, we denote for marking $[p_2, p_3]$ activities b and c as enabled, and for marking $[p_8]$ activities e, f, and g as enabled. As denoted, alignments contain a sequence of transitions. This sequence leads to a marking in a Petri net. We use the transition sequence to access the information on enabled activities at certain points in an alignment.

Definition 7 (Enabled Activities in an Alignment) Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, $N = (P, T, F, A, l)$ be a Petri net, and $\gamma = \langle (e_1, t_1), \ldots, (e_{|\gamma|}, t_{|\gamma|}) \rangle \in (L^{\gg} \times T^{\gg})^*$ be an alignment. For any $i \in \{1, \ldots, |\gamma|\}$ and if $t_i \neq \gg$ we define $en^{\gamma}(t_i) = en^N_{\sigma}(\pi^*_2(pref_{i-1}(\gamma))\restriction_T)$.

Given the Petri net in Fig. 1 and the alignment shown before, we can denote $en^{\gamma}(t_{a'}) = \{a\}$ and $en^{\gamma}(t_b) = \{b\}$.

In our work, we reinterpret the existing moves (synchronous, visible model, invisible model, and log move) and add three new moves that act as a generalization of the previous moves:

- *execution-change move*: when the enabled activities in the model and the event match, but the executed activities do not match. To illustrate this, consider the model displayed in Fig. 1 and the trace $\langle \underline{a}, \underline{b}, \underline{d}, \underline{e}\, f\, g, \underline{e}\, f \rangle$. For the second-to-last event, changing the execution from e to g results in reaching the final marking.
- *enabled-change move*: when the executed activities match, but the enabled activities do not match. As an example, consider the first case of the event log in Table 1. For e_2, b and c are enabled, but in the model, after executing $t_{a'}$, b is enabled, but c is not. Thus, more activities are enabled in the log than in the model. Also, the other way around is possible.
- *execution-enabled-change move*: a mixture of the previous two moves. Consider the second trace shown in Table 1. When executing $t_{a'}$, only b is enabled. In e_{12}, only c is enabled and gets executed. Thus, by changing the set of enabled activities and the execution change, we can align e_{12} with t_b.

The following definition captures the different moves formally.

Definition 8 (Moves). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, $N = (P, T, F, A, l)$ be a Petri net, and $\gamma \in (L^{\gg} \times T^{\gg})^*$ an alignment between σ and N. For all tuples $(e_i, t_i) \in \gamma$, $i \in \{1, \ldots, |\gamma|\}$, we define the following movements:

- *synchronous move* if $t_i \in T$, $e_i \in L$, $\pi_{act}(e_i) = l(t_i)$ and $\pi_{en}(e_i) = en^{\gamma}(t_i)$.
- *log move* if $e_i \in L$ and $t_i = \gg$.
- *invisible move on model* if $e_i = \gg$ and $t_i \in T$, $l(t_i) = \tau$.
- *visible move on model* if $e_i = \gg$ and $t_i \in T$, $l(t_i) \neq \tau$.
- *execution-change move* if $t_i \in T$, $e_i \in L$, $\pi_{act}(e_i) \neq l(t_i)$, $l(t_i) \neq \tau$, but $\pi_{en}(e_i) = en^{\gamma}(t_i)$.
- *enabled-change move* if $t_i \in T$, $e_i \in L$, $\pi_{act}(e_i) = l(t_i)$, but $\pi_{en}(e_i) \neq en^{\gamma}(t_i)$.
- *execution-enabled-change move* if $e_i \in L$ and $t_i \in T$, $l(t_i) \neq \tau$, $\pi_{act}(e_i) \neq l(t_i)$, and $\pi_{en}(e_i) \neq en^{\gamma}(t_i)$.

Next, we define the costs for the different types of moves. Note that the synchronous move is a special case of the enabled-change move. Also, the execution-change move is a special case of the execution-enabled-change move.

Definition 9 (Costs). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, $N = (P, T, F, A, l)$ be a Petri net, and $\gamma \in (L^{\gg} \times T^{\gg})^*$

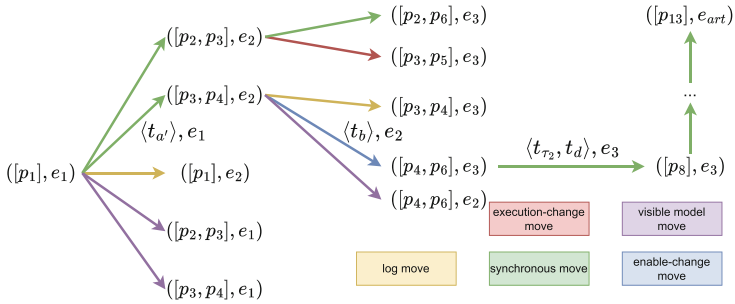


Fig. 3. Part of the synchronous state space between the trace $\langle \underline{a}, \underline{b}c, \underline{d}, \underline{e}fg, \underline{g} \rangle$ from the translucent event log shown in Fig. 1 and the Petri net displayed in Fig. 1.

an alignment between σ and N. For all tuples $(e_i, t_i) \in \gamma$, $i \in \{1, \ldots, |\gamma|\}$, we define the following cost function $\delta \colon L^{\gg} \times T^{\gg} \to [0, 2]$:

- synchronous and enabled-change move: $\delta(e_i, t_i) = D_{\alpha,\beta}(\pi_{en}(e_i), en^{\gamma}(t_i))$.
- log move: $\delta(e_i, t_i) = 1$.
- invisible move on model: $\delta(e_i, t_i) = 0$.
- visible move on model: $\delta(e_i, t_i) = 1$.
- execution-change move and execution-enabled-change move: $\delta(e_i, t_i) = 1 + D_{\alpha,\beta}(\pi_{en}(e_i), en^{\gamma}(t_i))$

Note that these moves do not result in higher costs than the classic worst-case scenario. Concerning the execution-enabled-change move, the worst case of a mismatch results in costs of 2, which correspond to a log and model move.

4.2 Computing Optimal Translucent Alignments

The cost of an alignment corresponds to the sum of costs of the moves composing the alignment. An optimal alignment is an alignment with minimal costs among all possible alignments. Compared to the standard alignment approach, we just extended the possible moves and used a customized cost function. This adjustment still allows us to apply the general alignment computation method, which essentially constructs the synchronous state space and solves a shortest-path problem.

Figure 3 illustrates the finding of the shortest path in the synchronous state space for our exemplary trace $\langle \underline{a}, \underline{b}c, \underline{d}, \underline{e}fg, \underline{g} \rangle$ and Petri net. This also corresponds to the alignment example introduced previously. The distance between states corresponds to the cost of the move that connects them. We annotated the first three arcs of the shortest path with information on the moves they are representing. Although invisible log moves are not seen directly and, therefore, not given a separate color, they are still contained in the execution sequences of the model and thus implicitly captured by the arcs in the given example of the synchronous state space.

Eventually, any shortest path from the initial to the final state represents an optimal alignment. For the exemplary alignment, we obtain the following costs: $0 + 1 - \frac{|\{b,c\} \cap \{b\}|}{|\{b,c\} \cap \{b\}| + \alpha|\{c\}| + \beta|\emptyset|} + 0 + 0 + 0 + 0 = 1 - \frac{1}{1+\alpha}$. When we want to emphasize that there are enabled activities in the translucent event log that are not represented in the model, we set $\alpha > 0$. When $\alpha = 0$, the alignment has no costs; when $\alpha = 1$, the alignment has costs of 0.5.

4.3 Fitness Score

After defining translucent alignments, their costs, and their computation, we define a translucent fitness score. Similarly to the traditional setting, we thereby take the original approach to compute a trace score.

Definition 10 (Trace Score). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, $N = (P, T, F, A, l)$ be a Petri net, $M_i \in \mathcal{B}(P)$ be the initial marking, $M_f \in \mathcal{B}(P)$ the final marking, and $\gamma \in (L^{\gg} \times T^{\gg})^*$ an alignment between σ and N. Let $\sigma_T \in T^*$ be the shortest sequence of transitions through a Petri net, i.e., $(N, M_i) \xrightarrow{\sigma_T} (N, M_f)$ and $\nexists_{\sigma'_T \in T^*} \colon (N, M_i) \xrightarrow{\sigma'_T} (N, M_f) \wedge |\sigma'_T| < |\sigma_T|$. The trace score is defined as:

$$\theta(\sigma, N, \gamma) = 1 - \frac{\sum_{(e,t) \in \gamma} \delta(e, t)}{\sum_{t \in \sigma_T} \delta(\gg, t) + \sum_{e \in \sigma} \delta(e, \gg) - 2},$$

where the numerator is the actual costs of the alignment, and the denominator is the worst-case costs (shortest firing sequence plus length of the trace). To compensate for the artificial end activity in each trace and transition in the model, respectively, we subtract their additional cost from the denominator. This is not necessary for the numerator since the move between the artificial ends will always be synchronous, leading to costs of 0. As a result, the actual fitness score is obtained.

For our example alignment, the score is $1 - \frac{0.5}{(7+1)+(5+1)-2} \approx 0.96$ if $\alpha = 1$.

Next, we consider all traces of a translucent event log to compute a translucent log score. The following captures that.

Definition 11 (Log Score). Let $L \subseteq \mathcal{U}_{ev}$ be a translucent event log, Σ^L its set of traces, $\sigma \in \Sigma^L$ a trace, $N = (P, T, F, A, l)$ be a Petri net, and $\gamma_\sigma \in (L^{\gg} \times T^{\gg})^*$ an alignment between σ and N. The log score is defined as

$$\Theta(\Sigma^L, N) = \frac{\sum_{\sigma \in \Sigma^L} \theta(\sigma, N, \gamma_\sigma)}{|\Sigma^L|}.$$

5 Evaluation

Our evaluation is divided into two sections. First, we provide a qualitative evaluation, highlighting the implications of the different moves and α and β values. Afterward, we present a quantitative assessment of our method. We provide a large-scale evaluation using various logs to compare the traditional fitness score and the translucent fitness score. Moreover, we compare computation times per (translucent) variant.

Table 2. Qualitative evaluation for different traces and models.

Trace	Traditional Fitness Score	Translucent Fitness Score		
		$\alpha = 1, \beta = 1$	$\alpha = 0, \beta = 2$	$\alpha = 2, \beta = 0$
$\langle \underline{a}, \underline{b}, \underline{d}, \underline{e}\,\underline{f}\,\underline{g}, \underline{g} \rangle$	1.00	1.00	1.00	1.00
$\langle \underline{a}, \underline{b}c, \underline{d}, \underline{e}\,\underline{f}\,\underline{g}, \underline{g} \rangle$	1.00	0.95	1.00	0.93
$\langle \underline{a}, \underline{b}, \underline{d}, \underline{e}\underline{g}, \underline{g} \rangle$	1.00	0.97	0.95	1.00
$\langle \underline{a}, \underline{b}, \underline{d}, \underline{x}eg, \underline{g} \rangle$	0.80	0.85	0.85	0.85
$\langle \underline{a}, \underline{b}, \underline{e}\,\underline{f}\,\underline{g}, \underline{g} \rangle$	0.89	0.89	0.89	0.89
$\langle \underline{a}, \underline{b}, \underline{d}, \underline{e}\,\underline{f}\,\underline{g}, \underline{e}\,\underline{f} \rangle$	0.80	0.90	0.90	0.90

5.1 Qualitative Evaluation

First, we highlight the differences between the traditional method and our approach and investigate the impact of the parameters α and β. Results for different scenarios are shown in Table 2. As model, we use our example in Fig. 1.

The first row shows a perfectly fitting trace of the executed and enabled activities within the model. As we can denote, the score for all methods is 1.0. The second row highlights when more activities are enabled in the event log than allowed by the model path (specifically activities b and c, while the best path does not allow for c). Traditional alignment methods still return a perfect score because they do not consider enabled activities. When $\alpha = 0$, we also get a perfect score since we do not penalize additional enabled activities in the translucent event log. As we increase α to 1 or 2, the mismatch is penalized more heavily, reducing the fitness score. This reflects the discrepancy between the model and the event log regarding enabled activities. The third row shows the opposite of the former case: fewer activities are enabled in the data than by the model. As before, the traditional methods still produce a perfectly fitting score. When $\beta = 0$, the score is still perfect since this behavior is not penalized. Adjusting β allows us to penalize the mismatch, reducing the fitness score to reflect fewer enabled activities in the event log compared to the model. The fourth row highlights the results when an activity not contained in the model gets executed. The traditional method returns costs 2 (model and log move) and a score of 0.8. Our method assigns a lower cost of 1.5, resulting in a higher fitness score of 0.85. The costs are based on an execution change (cost 1) and the change in enabled activities. This approach accounts for potential noise in the data, such as false recordings of executed activities, since all activities match the enabled activities in the model. Therefore, information on enabled activities makes the fitness score more robust. Thus, our method provides holistic scoring and can determine the execution sequence more accurately, whereas traditional methods might allow for ambiguous alignments (e.g., permitting a model move before or after activity g). The fifth row is about the missing execution of d. All methods provide the same fitness score. The last row highlights the execution

change move and the resulting difference in scoring. Again, false recordings can lead to such data. However, our method seems more robust when this happens.

5.2 Quantitative Evaluation

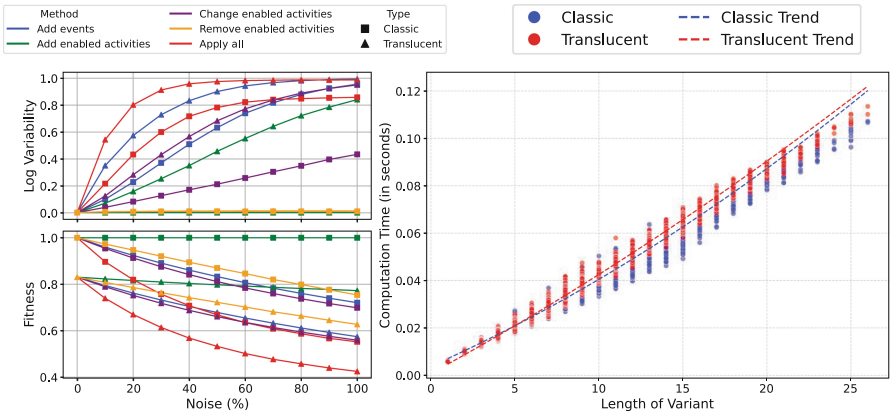
We provide an implementation in Python[1] using *PM4Py* [10]. Our method also computes classic alignments. Thus, we used our implementation for this part. Our data are available [14].

Although there are tools to create translucent event logs (see [11,13]), we manipulated existing event logs to allow for a large-scale comparison and different complexities. The logs we use are the road traffic fine management [26], the sepsis [31], and the hospital billing log [32]. For each log, we used the *Inductive Miner - infrequent* [23] with a threshold of 40 %, to discover a process model. Using the traditional alignment, we checked for fitting traces and kept them. We converted the Petri nets of each model into a Deterministic Finite Automaton (DFA) using the reachability graph and automata theory. By replaying each trace of a log on a DFA, we added information on enabled activities, thus creating a translucent event log. For more details, we refer to [11,12,15]. For each log's fitting and enhanced traces, we applied the *Inductive Miner* [22]. In the next step, we randomly modified the created event logs by either adding, changing, or removing enabled activities of events. If the selected enabled activity of an event is the executed activity, we also modify the executed activity or remove the event. Also, we introduced new events. In addition, we also applied all mentioned modifications to the logs. We added noise with respect to the number of events per log. Besides the classic definition of trace variants, we further introduce translucent variants. Translucent variants also consider the information of enabled activities. For example, $\langle \underline{a}, \underline{b}, \underline{c} \rangle$ and $\langle \underline{ab}, \underline{b}, \underline{c} \rangle$ belong to the same classic variant, but are different translucent variants. For the alignment computation, we used the standard setting of $\alpha = \beta = 1$ and the models discovered by the Inductive Miner [22] on the translucent logs without noise as reference models.

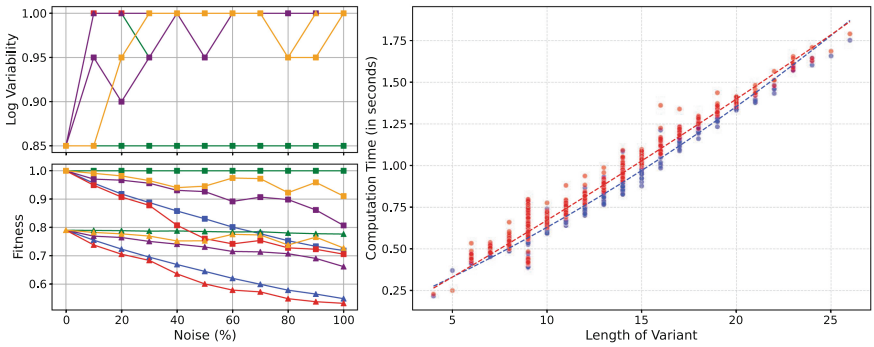
In our evaluation, we checked the log variability for each modified event log, i.e., the ratio between variants and traces. A value of 1 means that each variant has exactly one trace. Alignments are usually computed per trace variant. To measure the computation time, we took the median out of five repetitions of the computation per variant. Since the number of translucent variants is at least the same as for classic variants, we do not display the computation time per log but the computation time for all variants per log for the different logs. This allows for a fairer comparison. Our results are depicted in Fig. 4.

Regarding log variability, we generally observe more translucent variants than classic ones. In general, applying all modifications leads to the highest variability. This is followed by adding events, changing enabled activities, adding enabled activities, and removing enabled activities. Regarding the fitness scores, we denote that the translucent fitness score with no induced noise leads to lower scores than the classic approach. The reason is the enabled activities, which

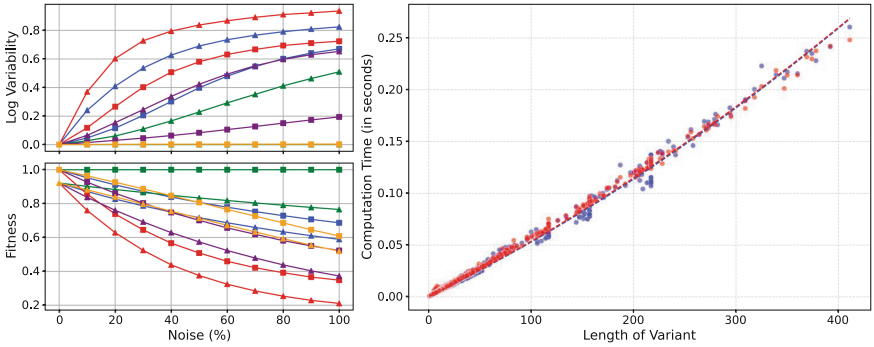
[1] https://github.com/hherbertb/translucent_alignment.



(a) Results for the road traffic fine management log [26].



(b) Results for the sepsis log [31].



(c) Results for the hospital billing log [32].

Fig. 4. Results of different translucent event logs based on real-life data.

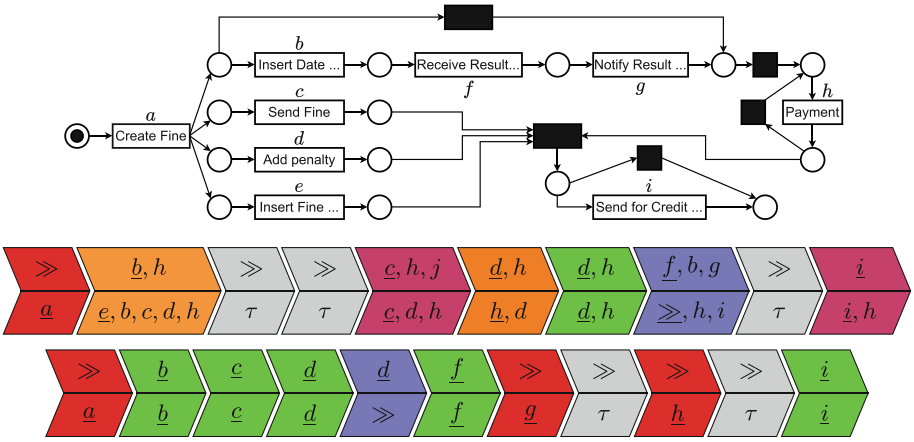


Fig. 5. Petri net with an exemplary translucent alignment of the translucent trace $\langle \underline{b}h, \underline{c}hj, \underline{d}h, \underline{d}h, \underline{f}bg, \underline{i} \rangle$ where j represents the activity "Appeal to Judge" which does not appear in the model. The translucent alignment begins with a model move (red), followed by an execution-enabled-change move (orange, cost of 1.6), and two silent moves (gray). The next two moves are an enabled-change move (purple, cost of 0.5) and an execution-change move (orange, cost of 1). The color saturation represents the cost and, thereby, the degree of changes in the enabled activities. The alignment continues with a synchronous move (green) and a log move (blue) before ending with another silent move (gray) and enabled-change move (purple, cost of 0.5). For comparison, the classic alignment ignoring translucent information is shown underneath.

were extracted from a different model. This again highlights the value of considering enabled activities for fitness measurement. As before, applying all noise methods had the worst impact on fitness values. Overall, adding enabled activities had no impact on the classical view, as expected. For the other methods of inducing noise, the difference in the score exists, up to roughly 0.2. Considering the computation time, we observe that it is nearly quadratic. Moreover, considering enabled activities in the computation has a negligible computational overhead. One of the main reasons is that the Tversky index can be quickly computed. Nonetheless, alignments are usually computed per variant, and since the number of translucent variants is at least the same as classic variants, the computation time for translucent event logs is at least roughly the same as for classic logs.

A translucent alignment from the enhanced road traffic fine management log with all methods and 20 % noise is shown in Fig. 5. Enabled-change moves occur, indicating that the model allows for behavior different from the data. With the execution-change move, we may conclude that there is noise in the data. Comparing the optimal translucent alignment with the classic optimal alignment, we observe that the latter contains more synchronous moves. The reason is that it does not account for deviations within the enabled activities. In practice, this information can help determine whether the desktop application works as

intended. The translucent alignment has costs of 5.6, the classic alignment costs of 4. Hence, the fitness score of the classic alignment is higher.

6 Conclusion

We provided the first method that considers information on enabled activities in a log and a model when evaluating fitness by introducing three moves that also generalize existing ones. The first deals with situations where the information on enabled activities matches but not the information on executed activities. The second deals with situations where the information on executed activities matches but not the information on enabled activities. The third combines the former two. We can verify system requirements using translucent logs from capturing desktop environments. Our method builds on state-of-the-art techniques and provides holistic scoring. In future work, we plan to extend token-based replay to consider enabled activities, making it suitable for industrial-scale tools due to its faster runtime and relevance to our applications. Additionally, translucent event logs should be considered when assessing generalization and evaluating different models, such as process trees.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action. Second Edition. Springer (2016). https://doi.org/10.1007/978-3-662-49851-4
2. van der Aalst, W.M.P.: Lucent process models and translucent event logs. Fundam. Informaticae **169**(1–2), 151–177 (2019). https://doi.org/10.3233/FI-2019-1842
3. van der Aalst, W.M.P., Stahl, C.: Modeling Business Processes - A Petri Net-Oriented Approach. MIT Press (2011)
4. Adams, J.N., van der Aalst, W.M.P.: Precision and fitness in object-centric process mining. In: ICPM, pp. 128–135. IEEE (2021). https://doi.org/10.1109/ICPM53251.2021.9576886
5. Adriansyah, A.: Aligning observed and modeled behavior. Phd thesis, Mathematics and Computer Science (2014). https://doi.org/10.6100/IR770080
6. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In: La Rosa, M., Soffer, P. (eds) BPM - International Workshops, pp. 137–149. Springer (2012). https://doi.org/10.1007/978-3-642-36285-9_15
7. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.: Linking data and process perspectives for conformance analysis. Comput. Secur. **73**, 172–193 (2018). https://doi.org/10.1016/J.COSE.2017.10.010
8. Berti, A., van der Aalst, W.M.P.: Reviving token-based replay: Increasing speed while improving diagnostics. In: ATAED@Petri Nets/ACSD 2019, pp. 87–103. CEUR-WS.org (2019)
9. Berti, A., van der Aalst, W.M.P.: A novel token-based replay technique to speed up conformance checking and process enhancement. Trans. Petri Nets Other Model. Concurr. **15**, 1–26 (2021). https://doi.org/10.1007/978-3-662-63079-2_1
10. Berti, A., van Zelst, S., Schuster, D.: PM4Py: a process mining library for python. Software Impacts **17**, 100556 (2023). https://doi.org/10.1016/j.simpa.2023.100556

11. Beyel, H.H., van der Aalst, W.M.P.: Creating translucent event logs to improve process discovery. In: Montali, M., Senderovich, A., Weidlich, M. (eds) ICPM Workshops, pp. 435–447. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-27815-0_32
12. Beyel, H.H., van der Aalst, W.M.P.: Improving process discovery using translucent activity relationships. In: Marrella, A., Resinas, M., Jans, M., Rosemann, M. (eds) BPM, pp. 146–163. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-70396-6_9
13. Beyel, H.H., Manuel, S., van der Aalst, W.M.P.: ActivityGen: extracting enabled activities from screenshots. In: ECAI, pp. 712–720. IOS Press (2024). https://doi.org/10.3233/FAIA240553
14. Beyel, H.H., Schwanen, C.T., van der Aalst, W.M.P.: Data for Translucent Alignments (2025). https://doi.org/10.5281/zenodo.15210626
15. Beyel, H.H., van der Aalst, W.M.P.: Translucent precision: Exploiting enabling information to evaluate the quality of process models. In: Araújo, J., de la Vara, J.L., Santos, M.Y., Assar, S. (eds) RCIS, pp. 29–37. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-59468-7_4
16. vanden Broucke, S.K.L.M., Munoz-Gama, J., Carmona, J., Baesens, B., Vanthienen, J.: Event-based real-time decomposed conformance analysis. In: Meersman, R., et al., OTM. Springer, Cham (2014). https://doi.org/10.1007/978-3-662-45563-0_20
17. vanden Broucke, S., De Weerdt, J., Vanthienen, J., Baesens, B.: Determining process model precision and generalization with weighted artificial negative events. IEEE Trans. Knowl. Data Eng. **26**(8), 1877–1889 (2014). https://doi.org/10.1109/TKDE.2013.130
18. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018). https://doi.org/10.1007/978-3-319-99414-7
19. van Dongen, B.F.: Efficiently computing alignments - using the extended marking equation. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds) BPM, pp. 197–214. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_12
20. van Dongen, B.F., Carmona, J., Chatain, T., Taymouri, F.: Aligning modeled and observed behavior: a compromise between computation complexity and quality. In: Dubois, E., Pohl, K. (eds) CAiSE, pp. 94–109. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_7
21. Dumas, M., Rosa, M.L., Leno, V., Polyvyanyy, A., Maggi, F.M.: Robotic process mining. In: van der Aalst, W.M.P., Carmona, J. (eds) Process Mining Handbook, pp. 468–491. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08848-3_16
22. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, JM., Desel, J. (eds) PETRI NETS, pp. 311–329. Springer, Cham (2013). https://doi.org/10.1007/978-3-642-38697-8_17
23. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds) Business Process Management Workshops, pp. 66–78. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-06257-0_6
24. Leemans, S.J.J., Polyvyanyy, A.: Stochastic-aware conformance checking: an entropy-based approach. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds) CAiSE, pp. 217–233. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49435-3_14

25. Leemans, S., Polyvyanyy, A.: Stochastic-aware precision and recall measures for conformance checking in process mining. Inf. Syst. **115**, 102197 (2023). https://doi.org/10.1016/J.IS.2023.102197
26. de Leoni, M.M., Mannhardt, F.: Road traffic fine management process (2015). https://doi.org/10.4121/UUID:270FD440-1057-4FB9-89A9-B699B47990F5
27. de Leoni, M., van der Aalst, W.M.P.: Aligning event logs and process models for multi-perspective conformance checking: an approach based on integer linear programming. In: Daniel, F., Wang, J., Weber, B. (eds) BPM, pp. 113–129. Springer, Cham (2013). https://doi.org/10.1007/978-3-642-40176-3_10
28. de Leoni, M., van der Aalst, W.M.P., van Dongen, B.F.: Data- and resource-aware conformance checking of business processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds) BIS, pp. 48–59. Springer, Cham (2012). https://doi.org/10.1007/978-3-642-30359-3_5
29. de Leoni, M., Marrella, A.: Aligning real process executions and prescriptive process models through automated planning. Expert Syst. Appl. **82**, 162–183 (2017). https://doi.org/10.1016/J.ESWA.2017.03.047
30. Liss, L., Adams, J.N., van der Aalst, W.M.P.: Object-centric alignments. In: Almeida, J.P.A., Borbinha, J., Guizzardi, G., Link, S., Zdravkovic, J. (eds) ER, pp. 201–219. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47262-6_11
31. Mannhardt, F.: Sepsis cases - event log (2016). https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460
32. Mannhardt, F.: Hospital billing - event log (2017). https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741
33. Mannhardt, F., Leemans, S.J.J., Schwanen, C.T., de Leoni, M.: Modelling data-aware stochastic processes - discovery and conformance checking. In: Gomes, L., Lorenz, R. (eds) PETRI NETS, pp. 77–98. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33620-1_5
34. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2016). https://doi.org/10.1007/S00607-015-0441-1
35. Padró, L., Carmona, J.: Computation of alignments of business processes through relaxation labeling and local optimal search. Inf. Syst. **104**, 101703 (2022). https://doi.org/10.1016/J.IS.2020.101703
36. Reißner, D., Armas-Cervantes, A., Conforti, R., Dumas, M., Fahland, D., Rosa, M.L.: Scalable alignment of process models and event logs: an approach based on automata and s-components. Inf. Syst. **94**, 101561 (2020). https://doi.org/10.1016/J.IS.2020.101561
37. Reißner, D., Conforti, R., Dumas, M., Rosa, M.L., Armas-Cervantes, A.: Scalable conformance checking of business processes. In: Panetto, H., et al. OTM, pp. 607–627. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69462-7_38
38. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008). https://doi.org/10.1016/J.IS.2007.07.001
39. Schwanen, C.T., Pakusa, W., van der Aalst, W.M.P.: A dynamic programming approach for alignments on process trees. In: ICPM Workshops (2025). https://doi.org/10.1007/978-3-031-82225-4_7
40. Schwanen, C.T., Pakusa, W., van der Aalst, W.: Process tree alignments. In: EDOC (2025). https://doi.org/10.1007/978-3-031-78338-8_16
41. Syed, R., et al.: Robotic process automation: contemporary themes and challenges. Comput. Ind. **115**, 103162 (2020). https://doi.org/10.1016/j.compind.2019.103162

42. Taymouri, F., Carmona, J.: An evolutionary technique to approximate multiple optimal alignments. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds) BPM, pp. 215–232. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_13
43. Taymouri, F., Carmona, J.: Structural computation of alignments of business processes over partial orders. In: ACSD, pp. 73–81. IEEE (2019). https://doi.org/10.1109/ACSD.2019.00012

Object-Centric Processes with Structured Data and Exact Synchronization

Formal Modelling and Conformance Checking

Alessandro Gianola[1(✉)], Marco Montali[2], and Sarah Winkler[2]

[1] INESC-ID/Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
alessandro.gianola@tecnico.ulisboa.pt
[2] Free University of Bozen-Bolzano, Bolzano, Italy
{montali,winkler}@inf.unibz.it

Abstract. Real-world processes often involve interdependent objects that also carry data values, such as integers, reals, or strings. However, existing process formalisms fall short to combine key modeling features, such as tracking object identities, supporting complex datatypes, handling dependencies among them, and object-aware synchronization. Object-centric Petri nets with identifiers (OPIDs) partially address these needs but treat objects as unstructured identifiers (e.g., order and item IDs), overlooking the rich semantics of complex data values (e.g., item prices or other attributes). To overcome these limitations, we introduce data-aware OPIDs (DOPIDs), a framework that strictly extends OPIDs by incorporating structured data manipulation capabilities, and full synchronization mechanisms. In spite of the expressiveness of the model, we show that it can be made operational: Specifically, we define a novel conformance checking approach leveraging satisfiability modulo theories (SMT) to compute data-aware object-centric alignments.

Keywords: Object-centric conformance checking · Universal Synchronization · Data-aware Processes · Complex Datatypes · SMT

1 Introduction

In recent years, research in information systems supporting the execution of business and work processes has increasingly stressed the need for multi-perspective process models. Prominently, the goal has been to tackle intricate links between the control flow of a process, and the data with which the control flow interacts.

Several new process modelling paradigms and corresponding languages consequently emerged, ranging from case-handling [5,21] to artifact-centric [9,33] and object-aware approaches [7,22,31]. In parallel, a growing stream of research has spawned different formal models of data-aware processes with the twofold aim to support *representation* and *computation*, on the one hand covering relevant modelling constructs, on the other providing effective algorithmic techniques for process analysis and mining, such as automated discovery and conformance

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 185–202, 2025.
https://doi.org/10.1007/978-3-031-94571-7_11

checking. Within this stream, representations typically rely on data-aware extensions of Petri nets, the most widely employed formalism for describing, analysing, and mining processes, with two emerging directions.

The first focuses on enriching case-centric processes by incorporating structured case attributes (e.g., the price of a product, the age and name of a customer, as well as more complex structures such as a persistent relational storage). This supports expressing how activities in the process read and write these variables, and how decision points use these variables to express routing conditions for cases. A prime example in this vein is that of Data Petri nets (DPNs [23,26]).

The second direction aims instead at lifting the case-centricity assumption, tackling so-called *object-centric processes* where multiple objects, interconnected via complex one-to-many and many-to-many relationships, are co-evolved by the process (e.g., orders containing multiple products, shipped in packages that may mix up products from different orders). It has been pointed out that straightjacketing this complexity through a single case notion yields misleading process analysis and mining results [1,6]. Several proposals have been brought forward in this direction, such as object-centric Petri nets [4], synchronous proclets [11], and Petri nets with identifiers (PNIDs) [18,29,34], possibly equipped with an external relational storage [16,28]. To a varying extent, they cover essential modelling features like: (*i*) tracking objects and their possibly concurrent flows, enabling the independent progression of objects (e.g. package shipments and order notifications);(*ii*) object creation and manipulation, handling dependencies such as one-to-one and one-to-many relationships (e.g., adding items to an order or splitting it into multiple packages); (*iii*) object-aware full synchronization, creating flow dependencies on objects, where an object can flow through an activity only if some (*subset synchronization*) or all (*exact synchronization*) related objects simultaneously flow through that activity; this ensures that an object can proceed through an activity only when certain conditions are met (e.g., initiating order billing only when some or all associated packages have been delivered).

Two main open challenges emerge in the tradeoff of these two directions. First, there is a lack of object-centric models that offer both object relationship manipulation and corresponding synchronization mechanisms: while approaches based on PNIDs provide fine-grained constructs for handling objects and their mutual relationships, they fall short in fully addressing synchronization in its different flavors; complementarily, alternative approaches in the object-centric spectrum suffer from under-specification issues [2,18]. Second, object-centric models completely lack support for attributes and corresponding data conditions.

The ultimate goal of this work is to address these two research questions through a unified formalism supporting at once fine-grained modelling features for objects, relationships, attributes, and complex data conditions to express transition guards, subset/exact synchronization, and combinations thereof. The following example inspired by [4,18] motivates the need for such a formalism:

Example 1. In an order-to-shipment process, executions involve the following activities: (*i*) place order creates a new order with an arbitrary number of

products, and the customer can indicate the number of days expected for delivery;.(*ii*) Payment can be done via credit card or bank transfer, reflected by pay cc and pay bt, respectively. However, the former is only applicable if the total cost of the order is below 1000 €.(*iii*) pick item fetches a product from the warehouse; *iv* ship collects an order with all its products for shipment. It also determines one of two shipment modes, namely car or truck, depending on whether the number of days for delivery is below or above 5.

This example requires a variety of sophisticated modelling features: during order creation, an arbitrary number of product objects can be included (in the sequel we call this *multi-object transfer*); items are associated with an order (*object relations*), and *all* items of an order must be included in shipment (in the sequel called *exact synchronization*, as opposed to *subset synchronization*); one needs to reason over arithmetic data like the cost of a product and the total cost of an order, and transition guards are needed (*structured data support*).

Specifically, we provide a twofold contribution in *representation* and *computation*, to handle such processes. As for *representation*, we start from OPIDs [18], the most sophisticated PNID-based formalism: they support all main object-centric modelling features except exact synchronization. We lift OPIDs into a new class of PNIDs called DOPIDs, which at once close the gap regarding synchronization, and add rich data support for a variety of data types, together with conditions expressed over such data that can involve arithmetic, uninterpreted functions, object properties, and advanced forms of aggregation. Aggregation emerges as a natural, non-trivial new modelling construct arising from the interplay between data conditions à la DPNs, and the fact that they are now applied over the attributes of possibly multiple objects at once. As for *computation*, we consider conformance checking, and show that existing SMT-based techniques can be lifted to fully cover all features of DOPIDs. We do so by integrating and extending the SMT encodings for conformance checking separately studied for OPIDs [18] and DPNs [12,13]. Finally, we provide a novel proof-of-concept implementation of our approach to witness its feasibility.

2 Related Work and Modelling Features

To highlight key modeling features, we reviewed literature on Petri nets enriched with case attributes [8,14,17] and object-centric features [1,3,6]. Table 1 shows a summary of these features and their implementation in various approaches. At the end of Sect. 4 we discuss the expressivity of DOPIDs more generally.

The first crucial feature is the incorporation of constructs for *creating and deleting objects*. Different approaches vary based on whether objects are *explicitly referenced* within the model or are only *implicitly manipulated*. Another critical aspect is the ability of objects to *flow concurrently* and independently; for example, items can be picked while their corresponding order is paid (cf. *divergence* in [1]). Additionally, models may support the *simultaneous transfer of multiple objects* of the same type, such as processing several items in a single

Table 1. Comparison of Petri net-based object-centric process modelling languages along main modelling features, tracking which approaches support conformance. ✓ indicates full, direct support, ✗ no support, and ∼ indirect or partial support.

	object creation	object removal	concurrent object flows	multi-object transfer	multi-object spawning	object relations	subset sync	exact sync	coreference	struct. data	object reference	conformance
OC nets [4]	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	imp.	[24]
synchronous proclets [11]	✓	✓	✓	∼	✓	✓	✓	✓	✗	✗	imp.	✗
DPNs [23]	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	imp.	[26]
PNIDs [16,29,34]	✓	✓	✓	∼	∼	✓	∼	✗	✓	✗	exp.	✗
OPIDs [18]	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	exp.	[18]
DABs [8]	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	no	✗
DOPIDs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	exp.	*here*

transaction. Transitions in these models must account for the manipulation of multiple objects, either of the same type or different types, at the same time (cf. *convergence* in [1]). A type of convergence occurs when a single transition, given a parent object, *generates an unbounded number of child objects* that are all linked to the parent; e.g., placing an order can create unboundedly many associated items. Once this parent-child *one-to-many relationship* is established, other forms of convergence, such as synchronizing transitions, can be introduced. These transitions allow a parent object to evolve only if *some* (*subset synchronization*) or *all* its child objects (*exact synchronization*) are in a certain state. In addition, advanced *coreference* techniques can be employed to simultaneously examine and evolve multiple interconnected objects. Finally, an essential feature is the support for advanced and *structured data types*, such as integers, reals, lists, and arrays. They enhance the objects manipulated by the process with additional information, allowing users to define complex constraints that act as guards for the transitions in the process model, possibly incorporating background knowledge. This final feature, unlike the others, is more characteristic of data-aware extensions of *case-centric* process models [27], where the focus is traditionally placed on the complex structure of data values, often governed by relational logical theories [10]. Among these approaches, the most advanced framework is the DAB model [8,15], which supports rich forms of database-driven data and sophisticated forms of reasoning. In process mining, multi-perspective models capable of incorporating richer data representations while expressing concurrent flows have also been introduced. A prominent example are Data Petri Nets (DPNs) [14], a Petri net-based formalism that, while more expressive, remains case-centric.

Regarding object-centric models, several Petri net-based formal models have been introduced [4,11,16,29,32,34]. Resource-constrained ν-Petri nets [32] constitute the first formal model supporting a basic form of object-centricity, but without relationships between objects. Object-centric nets [4] offer an implicit approach to object centricity where places and transitions have different object types. Simple arcs match with a single object at a time, while variable arcs handle arbitrarily many objects of the same type. However, the lack of object

relationships prevents modeling object synchronization and coreference. Alignment-based conformance checking for this approach is developed in [24]. Synchronous proclets [11] offer a framework that can implicitly express the tracking of objects and their mutual relationships. They include specialized constructs to support the described types of convergence, including subset and exact synchronization, though other forms of coreference are not supported. Multi-object transfers are approximated through iteration, processing objects one by one. Conformance checking for proclets is implicitly tackled here for the first time, considering that DOPIDs generalize proclets. Petri nets with identifiers (PNIDs), and variants, have been studied in [16,29,34], though without addressing conformance checking. PNIDs build upon ν-Petri nets by explicitly managing objects and their relationships through identifier tuples. Unlike object-centric nets and proclets, PNIDs lack constructs for manipulating unboundedly many objects in a single transition. As demonstrated in [16], multi-object transfers, spawning, and subset synchronization can be achieved through object coreference and iterative operations. In OPIDS [18], multi-object transfer is natively supported, but no data and only subset synchronization; though during conformance checking, exact synchronization can be obtained as a by-product. Indeed, no variant of PNID supports data-aware wholeplace operations as necessitated by exact synchronization. DOPIDs strictly subsume OPIDS, extending them with full object-aware synchronization and rich data support. This includes data of numeric, string, or free data types, and complex transition guards involving arithmetic and string operations as well as uninterpreted functions. These features significantly boost expressivity of the process model.

3 Object-Centric Event Logs with Data Attributes

We start from object-centric event logs as in [4,18], and enrich them with data attributes. To this end, we assume that data types are divided in two classes: a class Σ_{obj} of *object id* types, and a class Σ_{val} of *data-value* types.

Consistently with the literature, every object id type $\sigma \in \Sigma_{obj}$ has an (uninterpreted) domain $dom(\sigma) \subseteq \mathcal{O}$, given by all object ids in $\mathcal{O}$ of type σ. Such identifiers are used to refer to objects in the real world, and can be compared only for equality and inequality. Examples are order and product identifiers.

To capture the data attributes attached to objects and recorded in event logs, such as for example the price of a product and the delivery address of an order, we also introduce *data-value domains* for data-value types in Σ_{val}: $dom(\texttt{bool}) = \mathbb{B}$, the booleans; $dom(\texttt{int}) = \mathbb{Z}$, the integers; $dom(\texttt{rat}) = \mathbb{Q}$, the rational numbers; and $dom(\texttt{string}) = \mathbb{S}$, the strings over some fixed alphabet; unconstrained finite sets $dom(\texttt{finset}_i) = \mathbb{D}_i$, for some finite set $\mathbb{D}_i$, We will also support function and relation symbols over all these types to capture implicit properties of objects that are not explicitly manipulated by the process. E.g., in an order management process where every order has a delivery address (explicitly manipulated by the process) and an owner (implicitly assumed, but not directly manipulated), one may opt for modelling the delivery address as a string, and the owner as an uninterpreted function taking an order id as its only argument.

In addition to the sets Σ_{obj} and Σ_{val} for object and data-value types, we fix a set $\mathcal{A}$ of activities and a set $\mathbb{T}$ of timestamps equipped with a total order $<$. We also consider (partial) assignments from a set of variables $\mathcal{V}$ to elements of their domain. The set of all such assignments is denoted $Assign^{\mathcal{V}}$.

Definition 1. (Event log). *An* event log (with objects and attributes) *is a tuple* $L = \langle E, \mathcal{O}, \pi_{act}, \pi_{obj}, \pi_{time}, \pi_{val} \rangle$ *where: (i)* E *is a set of event identifiers; (ii)* $\mathcal{O}$ *is a set of object identifiers that are typed by a function type*: $\mathcal{O} \to \Sigma_{obj}$*; (iii) the functions* $\pi_{act}\colon E \to \mathcal{A}$, $\pi_{obj}\colon E \to \mathcal{P}(\mathcal{O})$, *and* $\pi_{time}\colon E \to \mathbb{T}$ *associate each event* $e \in E$ *with an activity, a set of affected objects, and a timestamp, respectively, such that for every* $o \in \mathcal{O}$ *the timestamps* $\pi_{time}(e)$ *of all events* e *such that* $o \in \pi_{obj}(e)$ *are all different; (iv) the function* $\pi_{val}\colon E \to Assign^{\mathcal{V}}$ *associates each event* $e \in E$ *with a set of data-values for attributes in* $\mathcal{V}$*.*

For an event log and an object $o \in \mathcal{O}$, we write $\pi_{trace}(o)$ for the tuple of events involving o, ordered by timestamps. Formally, $\pi_{trace}(o) = \langle e_1, \ldots, e_n \rangle$ such that $\{e_1, \ldots, e_n\}$ is the set of events in E with $o \in \pi_{obj}(e)$, and $\pi_{time}(e_1) < \cdots < \pi_{time}(e_n)$. In examples, we often leave $\mathcal{O}$, $\mathcal{A}$ and domains implicit and present an event log L as a set of tuples $\langle e, \pi_{act}(e), \pi_{obj}(e), \pi_{time}(e), \pi_{val}(e) \rangle$ representing events. Timestamps are shown as natural numbers, and concrete event ids as $\#_0, \#_1, \ldots$. The next example demonstrates an event log related to the process outlined in Example 1.

Example 2. Consider the set of objects $\mathcal{O} = \{\mathsf{o}_1, \mathsf{p}_1, \mathsf{p}_2\}$ with $type(\mathsf{o}_1) = order$ and $type(\mathsf{p}_1) = type(\mathsf{p}_2) = product$. The event log $E = \{\#_0, \#_1, \#_2, \#_3\}$ with the events detailed below reports that order o_1 is placed with two products $\mathsf{p}_1, \mathsf{p}_2$ and 3 days for delivery. Then o_1 is paid by credit card, p_1 is picked, and finally o_1 is shipped only with p_1, confirming the 3 days and selecting truck mode:

$$\langle \#_0, \mathsf{place\ order}, \{\mathsf{o}_1, \mathsf{p}_1, \mathsf{p}_2\}, \{d \mapsto 3\}, 1 \rangle, \langle \#_1, \mathsf{pay\ cc}, \{\mathsf{o}_1\}, \emptyset, 4 \rangle,$$
$$\langle \#_2, \mathsf{pick\ item}, \{\mathsf{o}_1, \mathsf{p}_1\}, \emptyset, 5 \rangle, \langle \#_3, \mathsf{ship}, \{\mathsf{o}_1, \mathsf{p}_1\}, \{d \mapsto 3, s \mapsto \mathsf{truck}\}, 9 \rangle$$

The notions of object and trace graphs from [4,18] remain identical also in our setting, but we report them here for completeness. The *object graph* $\mathcal{G}_L$ of an event log L is the undirected graph with node set $\mathcal{O}$, and an edge from o to o' if there is some event $e \in E$ such that $o \in \pi_{obj}(e)$ and $o' \in \pi_{obj}(e)$. Thus, the object graph indicates which objects share events. Let X be a connected component in $\mathcal{G}_L$. Then, the *trace graph* induced by X is the directed graph $T_X = \langle E_X, D_X \rangle$ where: (i) the set of nodes E_X is the set of all events $e \in E$ that involve objects in X, i.e., such that $X \cap \pi_{obj}(e) \neq \emptyset$, and (ii) the set of edges D_X consists of all $\langle e, e' \rangle$ such that for some $o \in \pi_{obj}(e) \cap \pi_{obj}(e')$, it is $\pi_{trace}(o) = \langle e_1, \ldots, e_n \rangle$ and $e = e_i$, $e' = e_{i+1}$ for some $0 \leq i < n$. Notably, the notion of trace graph is not modified despite the presence of data: edges only relate events that share objects, independent of possibly shared data values.

4 Data-Aware Object-Centric Petri Nets with Identifiers

We define *data-aware object-centric Petri nets with identifiers* (DOPIDs), enriching OPIDs [18] with complex data and full synchronization capabilities.

As in PNIDs and OPIDs, objects can be created in DOPIDs using ν variables. However, in DOPIDs tokens can carry object ids, data values, or tuples combining these. The latter account at once for relationships among objects, and attributes connecting objects to data values. So objects can be linked to other objects or data values, e.g. as in Example 4, where a product tracks the order it belongs to, and an order is associated to its shipment mode. Arcs are labeled with (tuples of) variables to match with objects and relations, as explained later. In the style of object-centric nets [4] and synchronous proclets [11], DOPIDs can spawn and transfer multiple objects at once, using an extension of the mechanism in OPIDs based on special "list variables" that match with *lists of objects*. The refinement consists in the possibility of indicating, when operating over a number of objects by consuming multiple tokens at once, whether one wants to consume *some* or *all* matching objects. The latter case could not be tackled in OPIDs, and is essential to cover exact synchronization.

Formal Definition. Let $\Sigma = \Sigma_{obj} \uplus \Sigma_{val}$ be the set of base types, including object types and data-value types. As in colored Petri nets, each place has a *color*: a cartesian product of data types from Σ. More precisely, the set of colors Col is the set of all $\sigma_1 \times \cdots \times \sigma_m$ such that $m \geq 1$ and $\sigma_i \in \Sigma$ for all $1 \leq i \leq m$. In addition, let the set of list types Σ_{list} consist of all $[\sigma]$ such that $\sigma \in \Sigma_{obj}$.

In DOPIDs, tokens are tuples of object ids and data values, each associated with a color. E.g., to model the process in Example 1, we want to use $\langle \mathtt{o}_1 \rangle$, $\langle \mathtt{o}_1, \mathtt{p}_1 \rangle$, and $\langle \mathtt{o}_1, 3 \rangle$ as tokens – respectively representing the order $\mathtt{o}_1$, the relationship indicating that product $\mathtt{p}_1$ is contained in order $\mathtt{o}_1$, and the fact that $\mathtt{o}_1$ has 3 days of desired delivery by the customer. In contrast to standard Petri nets which have only indistinguishable black tokens, DOPIDs keep track of object identity when firing transitions. To this end, we use a set of variables $\mathcal{X}$ in arc labels, to act as placeholders for objects or lists of objects. The variables in $\mathcal{X}$ are assumed to be *typed* in the sense that there is a function $type \colon \mathcal{X} \to \Sigma \cup \Sigma_{list}$ assigning a type to each variable.
The set of variables $\mathcal{X} = \mathcal{V} \uplus \mathcal{V}_{list} \uplus \mathcal{V}_{list}^{\subseteq} \uplus \mathcal{V}_{list}^{=} \uplus \Upsilon$ is the disjoint union of:

1. a set $\mathcal{V}$ of "normal" variables that refer to single objects or data values, denoted by lower-case letters like v, with a type $type(v) \in \Sigma$;
2. a set $\mathcal{V}_{list}$ of list variables referring to a list of objects of the same type, denoted by upper case letters like U, with $type(U) = [\sigma] \in \Sigma_{list}$;
3. two sets $\mathcal{V}_{list}^{\subseteq}$ and $\mathcal{V}_{list}^{=}$ that contain *annotated* list variables $U^{\subseteq}$ and $U^{=}$ resp., for each list variable U in $\mathcal{V}_{list}$ – these will be used to express whether *some* or *all* objects matching the variable must be considered;
4. a set Υ of variables ν referring to fresh objects, with $type(\nu) \in \Sigma_{obj}$.

We assume that infinitely many variables of each kind exist, and for every $\nu \in \Upsilon$, that $dom(type(\nu))$ is infinite, for unbounded supply of fresh objects [30].

To capture relationships between objects in consumed and produced tokens when firing transitions, we need arc *inscriptions*, which are tuples of variables.

Definition 2. (Inscription). *An* inscription *is a tuple* $\boldsymbol{v} = \langle v_1, \ldots, v_m \rangle$ *such that* $m \geq 1$ *and* $v_i \in \mathcal{X}$ *for all* i*, but at most one* $v_i \in \mathcal{V}_{list} \uplus \mathcal{V}_{list}^{\subseteq} \uplus \mathcal{V}_{list}^{=}$ *for*

$1 \leq i \leq m$. *We call* $\boldsymbol{v}$ *a* transfer-template inscription *if* $v_i \in \mathcal{V}_{list}$, $\subseteq$-template inscription *if* $v_i \in \mathcal{V}_{list}^{\subseteq}$, *or* $=$-template inscription *if* $v_i \in \mathcal{V}_{list}^{=}$ *for some* i, *and a* simple inscription *otherwise.*

For instance, for $o, p \in \mathcal{V}$ and $P, Q \in \mathcal{V}_{list}$, $\langle o, P \rangle$ is a transfer inscription and $\langle p \rangle$ a simple one. The inscription $\langle o, P^{=} \rangle$ is a $=$-template inscription as it contains the variable $P^{=}$ in $\mathcal{V}_{list}^{=}$, but $\langle P, P \rangle$ and $\langle P, Q \rangle$ are not valid inscriptions as they have two list variables. Note that a variable like P is only a placeholder for a list, it will be instantiated during execution by a concrete list, e.g. $[\mathtt{p}_1, \mathtt{p}_2, \mathtt{p}_3]$. By allowing at most one list variable in inscriptions, we restrict to many-to-one relationships between objects. However, recall that many-to-many relationships can be modeled as many-to-one with auxiliary objects, through reification.

Template inscriptions will be used to capture an arbitrary number of tokens of the same color: e.g., if o is of type *order* and P of type [*product*], then $\langle o, P \rangle$ refers to a single order with an arbitrary number of products. As we will see later, simple, $\subseteq$- and $=$-template inscriptions will be used when consuming tokens, while simple and transfer-template inscriptions will be employed when producing tokens. Specifically, when consuming tokens e.g. carrying order-product pairs from a place q, $\langle o, P^{\subseteq} \rangle$ selects *some* tokens from q, while $\langle o, P^{=} \rangle$ selects *all* of them. This is essential to model subset and exact synchronization (cf. Sect. 2).

We define the *color* of an inscription $\iota = \langle v_1, \ldots, v_m \rangle$ as the tuple of the types of the involved variables, i.e., $color(\iota) = \langle \sigma_1, \ldots, \sigma_m \rangle$ where $\sigma_i = type(v_i)$ if $v_i \in \mathcal{V} \cup \Upsilon$, and $\sigma_i = \sigma'$ if v_i is a list variable of type $[\sigma']$. Moreover, we set $vars(\iota) = \{v_1, \ldots, v_m\}$. E.g. for $\iota = \langle o, P \rangle$ with o, P as above, we have $color(\iota) = \langle order, product \rangle$ and $vars(\iota) = \{o, P\}$. The set of all inscriptions is denoted Ω.

To define guards on transitions, we consider the following definition of *constraints*, where we assume that uninterpreted functions and relations are defined over Σ (i.e., all object id and data value domains):

Definition 3. (Constraints). *For a set of variables* $\mathcal{V}$ *with list variables* $\mathcal{V}_{list} \subseteq \mathcal{V}$, *a* constraint c *and* expressions s, n, r, d, k, t_D, *and* t_K *are defined as follows:*

$$
\begin{aligned}
c &::= v_b \mid b \mid d = d \mid k \geq k \mid k > k \mid R(d, \ldots, d) \mid R(k, \ldots, k) \mid c \wedge c \mid \neg c \\
n &::= v_n \mid z \mid sum(Z) \mid min(Z) \mid max(Z) \mid n + n \mid -n \\
r &::= v_r \mid q \mid sum(Q) \mid min(Q) \mid max(Q) \mid mean(Q) \mid r + r \mid -r \\
s &::= v_s \mid h \mid f(s, \ldots, s) \quad d ::= s \mid f_w(d, \ldots, d) \mid f_w(k, \ldots, k) \\
k &::= n \mid r \mid g_w(k, \ldots, k) \mid g_w(d, \ldots, d) \mid k + k \mid -k \mid sum(t_K) \\
t_D &::= D \mid f_y(t_D) \mid f_y(t_K) \quad t_K ::= Z \mid Q \mid g_y(t_K) \mid g_y(t_D)
\end{aligned}
$$

where $v_b, v_s, v_n, v_r \in \mathcal{V}$, $type(v_b) = \mathtt{bool}$, $b \in \mathbb{B}$, $type(v_n) = \mathtt{int}$, $z \in \mathbb{Z}$, $type(v_r) = \mathtt{rat}$, $q \in \mathbb{Q}$, $type(v_s) = \mathtt{finset}_i$, $h \in \mathbb{D}_i$ *(some* i*)*, $Z, Q, D \in \mathcal{V}_{list}$, $type(Z) = [\mathtt{int}]$, $type(Q) = [\mathtt{rat}]$, D *has non-arithmetic type,* f_w, f_y *are functions with arithmetic codomains,* g_w, g_y *are functions with non-arithmetic ones.*

This definition may seem quite involved, but it captures essentially simple concepts. Term s defines a string as a variable, constant, or inductive function application. For expressions of integer type, term n allows variables, integers, or aggregators *sum*, *min*, and *max* applied to lists of integers. Term r is analogous to n but for rationals, for which also the aggregator *mean* is defined. Terms k and d define, by mutual induction, mixed terms that can combine different types: the only difference is that the root symbol for k lives in a arithmetical domain ($\mathbb{Z}$ or $\mathbb{Q}$), wheres for d it lives in a non-arithmetical domain. An analogous mutual induction defines the list terms t_D and t_K, which are built from list variables and functions, but differ in the fact that t_K lives in a arithmetical domain. Here a function applied to a list term is applied component-wise, and returns another list. Notice also that a term k can be produced by applying aggregator *sum* to a list variable t_k. Standard equivalences apply, hence disjunction (i.e., $\vee$) of constraints can be used, as well as comparisons $=, \neq, <, \leq$ on integer and rational expressions. The set of variables in a constraint φ is denoted $vars(\varphi)$, and the set of all constraints over variables $\mathcal{X}$ by $\mathcal{C}(\mathcal{X})$.

Example 3. We consider two constraints that will express transition guards for our running example. First, let d be an integer variable representing the maximum number of days expected for delivery by a customer, and m a string variable denoting the shipment mode of an order. Constraint $(d \leq 5 \wedge m = \mathsf{car}) \vee (d > 5 \wedge m = \mathsf{truck})$ expresses that either d is at most 5 d and the shipment mode is car, or that d is 6 d or more and the shipment mode is truck. Second, consider a list variable P for products and a unary function *cost* that returns the cost of each product, a rational number. This expresses the background knowledge that every product has a cost, that is however not explicitly manipulated by the process (so it will not appear in the log). Consistently with Definition 3, $cost(P)$ represents the list that contains the costs of all elements in P, and $sum(cost(P)) \leq 1000$ expresses that the overall cost of all products in P does not exceed 1000 €.

Definition 4. (DOPID). *A* data-aware object-centric Petri net with identifiers *(DOPID) is a tuple* $\mathcal{N} = (\Sigma_{obj}, \Sigma_{val}, P, T, F_{in}, F_{out}, color, \ell, guard)$*, where:*

1. P *and* T *are finite sets of places and transitions such that* $P \cap T = \emptyset$*;*
2. $color\colon P \to Col$ *maps every place to a color over* Σ*;*
3. $\ell\colon T \to \mathcal{A} \cup \{\tau\}$ *is the transition labelling where* τ *marks an invisible activity,*
4. $F_{in}\colon P \times T \to \Omega$ *is a partial function called* input flow *that satisfies* $color(F_{in}(p,t)) = color(p)$ *for every* $(p,t) \in dom(F_{in})$*;*
5. $F_{out}\colon T \times P \to \Omega$ *is a partial function called* output flow *that satisfies* $color(F_{out}(t,p)) = color(p)$ *for every* $(t,p) \in dom(F_{out})$*;*
6. $guard\colon T \to \mathcal{C}(\mathcal{X})$ *is a partial functions assigning guards, such that for every* $t \in T$ *and* $guard(t) = \varphi$*,* $vars(\varphi) \subseteq vars_{in}(t) \cup vars_{out}(t)$*, where* $vars_{in}(t) = \cup_{p \in P} vars(F_{in}(p,t))$ *and* $vars_{out}(t) = \cup_{p \in P} vars(F_{out}(t,p))$*.*

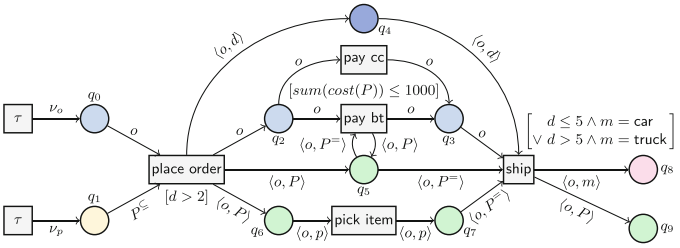


Fig. 1. DOPID of an order-to-ship process.

As a well-formedness condition, we assume that in F_{in} one can only use only simple, $\subseteq$-template and $=$-template inscriptions, while in F_{out} one can only use simple and transfer-template inscriptions (cf. Definition 2).

For a DOPID $\mathcal{N}$ as in Definition 4, we also use the common notations for presets $\bullet t = \{p \mid (p,t) \in dom(F_{in})\}$ and postsets $t\bullet = \{p \mid (t,p) \in dom(F_{out})\}$.

Simple flows (i.e., flows with simple inscriptions) are meant to consume and produce single tokens, whereas template flows (i.e., flows with template inscriptions) to consume and produce multiple matching tokens. Consumption in this case can be fine-tuned by indicating whether some (in the case of a $\subseteq$-template inscription) or all (for a $=$-template inscription) matching tokens have to be consumed. Production transfers such matched tokens to the corresponding output places, using transfer-template inscriptions. As illustrated by the next example and clarified by the definition of the semantics of DOPIDs, this is used to capture variable arcs in [24], but also to reconstruct different forms of synchronization. In particular, $=$-template inscriptions realize a form of data-aware wholeplace operations: they do not consume all tokens contained in a place, but all those that match the inscription. Example 4 illustrates the most important features of DOPIDs; it would not be expressible in existing object-centric formalisms.

Example 4. Figure 1 graphically depicts a DOPID for the simple yet sophisticated order-to-ship process informally described in Example 1. Variables ν_o of type *order* and ν_p of type *product*, both in Υ, refer to new orders and products. Normal variables $o, p \in \mathcal{V}$ of type *order* and *product* refer to existing orders and products, and variable P of type [*product*] to lists of products. We also use the data value variables d and m described in Example 3. For readability, single-component tuples are written without brackets (e.g., we write o instead of $\langle o \rangle$).

We explain the model transition by transition. The two silent transitions on the left have the purpose of injecting fresh orders and products in the net. Transition place order takes an order o from place q_0 and *some* available products P from place q_1 that are assigned to the order. Here the output transfer-template inscription $\langle o, P \rangle$, transfers to place q_5 $|P|$ tokens, each carrying a pair $\langle o, p \rangle$ with p taken from P. In this respect, place q_5 explicitly represents what in proclets is called *correlation set*, describing for every order in the system, which

products belong to it. After firing place order, also q_6 contains products of o, but from there single products will be consumed independently through pick item. Besides its products, place order assigns to the order o also the maximum number of days of delivery d, inserting the pair $\langle o, d\rangle$ in place q_4, responsible for tracking this attribute for every active order. Since d is only used in output flows, this reconstructs what in DPNs is called a "write" variable, which is moreover constrained by condition $d > 2$, capturing that d can only take values above 2 d. Finally, place order changes the state of the picked order o, moving it from place q_0 to q_2.

From q_2, two transitions can be fired for o, reflecting two payment modes. Specifically, order o either flows through pay cc, or through pay bt, but the latter can only be selected if the overall cost of o does not exceed 1000€ (cf. Example 3). To obtain all products of o, pay bt needs to fetch those products from place q_5, using inscription $\langle o, P^{=}\rangle$. This inscription requires that the first component matches order o consumed from place q_2, while $P^{=}$ forces all matching pairs for o to be included. As the aim is to use the products, but not to remove the corresponding pairs, they are all transferred back to q_5 using the inscription $\langle o, P\rangle$.

Concurrently with order payment, the state of single products (recalling their order) is changed when they are, one by one, picked via the pick item transition.

Finally, the ship transition is enabled for a paid order o under the following conditions: First, *all* its products must have been picked. This is expressed through the two $=$-template input flows with the same inscription $\langle o, P^{=}\rangle$, which has the effect of consuming all pairs containing products of o from places q_5 and q_7. The output transfer-template inscription $\langle o, P\rangle$ to place q_9 has the effect of transferring all those pairs there. At the same time, ship considers the value d for o, and through the attached constraint (cf. Example 3) determines the shipment mode for o, which is recorded in place q_8, linking m to o using inscription $\langle o, m\rangle$.

Two important remarks are in place wrt. Example 4. First, the modelling pattern in Fig. 1 that employs the "correlation" place q_5 to keep track of products contained in an order, paired with the two $=$-template inscriptions in shipment to ensure that *all* products of an order have been actually picked, is what makes DOPIDs able to support exact synchronization in the full generality of synchronous proclets [11], something that was out of reach until now for formal models based on PNIDs. Second, as DOPIDs handle multiple objects and data values at once, they are not only able to express read-write conditions and guards as in DPNs [23], but also more sophisticated conditions using aggregation expressions.

Semantics. Given the set of object ids $\mathcal{O}$ and a set of data-values $\mathcal{DV}$, the set of *tokens* $\boldsymbol{\mathcal{TOK}}$ is the set of tuples that consist of object ids and data values $\boldsymbol{\mathcal{TOK}} = \{(\mathcal{O} \uplus \mathcal{DV})^m \mid m \geq 1\}$. The *color* of a token $\omega \in \boldsymbol{\mathcal{TOK}}$ of the form $\omega = \langle d_1, \ldots, d_m\rangle$ is given by $color(\omega) = \langle type(d_1), \ldots, type(d_m)\rangle$. To define the execution semantics, we first introduce a notion of a *marking* of a DOPID $\mathcal{N} = \langle \Sigma_{obj}, \Sigma_{val}, P, T, F_{in}, F_{out}, color, \ell, guard\rangle$, namely as a function $M\colon P \to 2^{\boldsymbol{\mathcal{TOK}}}$, such that for all $p \in P$ and $\langle d_1, \ldots, d_m\rangle \in M(p)$, it holds that

$color(\langle d_1, \ldots, d_m \rangle) = color(p)$. Let $Lists(\mathcal{O})$ denote the set of object lists of the form $[o_1, \ldots, o_k]$ with $o_1, \ldots, o_k \in \mathcal{O}$ such that all o_i have the same object type; the type of such a list is then $[type(o_1)]$. Analogously, given $Lists(\mathcal{DV})$ the set of data-value lists $[dv_1, \ldots, dv_l]$ with $dv_1, \ldots, dv_l \in \mathcal{DV}$ such that all dv_i have the same data-value type, the type of such a list is $[type(dv_1)]$. Next, we define *bindings* to fix which data are involved in a transition firing.

Definition 5. (Binding). *A* binding *for a transition t and a marking M is a type-preserving function b: $vars_{in}(t) \cup vars_{out}(t) \to (\mathcal{O} \cup Lists(\mathcal{O})) \uplus (\mathcal{DV} \cup Lists(\mathcal{DV}))$, such that for all $U \in \mathcal{V}_{list}$, we have $b(U) = b(U^{=}) = b(U^{\subseteq})$. To ensure freshness of created values, we demand that b is injective on $\Upsilon \cap vars_{out}(t)$, and that $b(\nu)$ does not occur in M for all $\nu \in \Upsilon \cap vars_{out}(t)$.*

E.g., for transition ship in Example 4 the mapping b that sets $b(o) = \mathtt{o}_1$ and $b(P) = [\mathtt{p}_1, \mathtt{p}_2, \mathtt{p}_3]$ is a binding. Next, we extend bindings to inscriptions to fix which tokens participate in a transition firing. The extension of a binding b to inscriptions, i.e., variable tuples, is denoted $\boldsymbol{b}$. For an inscription $\iota = \langle v_1, \ldots, v_m \rangle$ and binding b such that $o_i = b(v_i)$ for all $1 \leq i \leq m$, let $\boldsymbol{b}(\iota)$ be the set of object tuples defined as follows: if ι is a simple inscription then $\boldsymbol{b}(\iota) = \{\langle o_1, \ldots, o_m \rangle\}$. Otherwise, there must be one v_i, $1 \leq i \leq n$, such that $v_i \in \mathcal{V}_{list}$, and consequently o_i must be a list, say $o_i = [u_1, \ldots, u_k]$ for some $u_1, \ldots, u_k$. Then $\boldsymbol{b}(\iota) = \{\langle o_1, \ldots, o_{i-1}, u_1, o_{i+1}, \ldots, o_m \rangle, \ldots, \langle o_1, \ldots, o_{i-1}, u_k, o_{i+1}, \ldots, o_m \rangle\}$. The set of all bindings is denoted by $\mathcal{B}$. We next define when a transition together with a binding is enabled in a marking.

Definition 6. (Enablement). *A transition $t \in T$ and a binding b for marking M are* enabled *in M if $b(guard(t))$ is satisfiable, for all $p \in \bullet t$, $\boldsymbol{b}(F_{in}(p,t)) \subseteq M(p)$ and if $F_{in}(p,t)$ is a $=$-variable flow with list variable $V^{=}$ there is no binding b' that differs from b only wrt. $V^{=}$ s.t. $\boldsymbol{b'}(F_{in}(p,t)) \subseteq M(p)$ and $b(V^{=}) \subset b'(V^{=})$.*

E.g., the binding b with $b(o) = \mathtt{o}_1$ and $b(P) = [\mathtt{p}_1, \mathtt{p}_2, \mathtt{p}_3]$ is enabled in marking M of the net in Example 4 with $\langle \mathtt{o}_1 \rangle \in M(q_{blue})$ and $\langle \mathtt{o}_1, \mathtt{p}_1 \rangle, \langle \mathtt{o}_1, \mathtt{p}_2 \rangle, \langle \mathtt{o}_1, \mathtt{p}_3 \rangle \in M(q_{green})$, for q_{blue} and q_{green} the input places of ship with respective color.

Definition 7. (Firing). *Let transition t and binding b be enabled in marking M. The* firing *of t with b yields the marking M' given by $M'(p) = M(p) \setminus \boldsymbol{b}(F_{in}(p,t))$ for all $p \in \bullet t \setminus t\bullet$, $M'(p) = M(p) \cup \boldsymbol{b}(F_{out}(p,t))$ for all $p \in t\bullet \setminus \bullet t$, and $M'(p) = M(p)$ for all $p \in t\bullet \cap \bullet t$.*

We write $M \xrightarrow{t,b} M'$ to denote that t is enabled with binding b in M, and its firing yields M'. A sequence of transitions with bindings $\rho = \langle (t_1, b_1), \ldots, (t_n, b_n) \rangle$ is called a *run* if $M_{i-1} \xrightarrow{t_i, b_i} M_i$ for all $1 \leq i \leq n$, in which case we write $M_0 \xrightarrow{\rho} M_n$. For such a binding sequence ρ, the *visible subsequence* ρ_v is the subsequence of ρ consisting of all (t_i, b_i) such that $\ell(t_i) \neq \tau$.

An *accepting* object-centric Petri net with identifiers is an object-centric Petri net $\mathcal{N}$ together with a set of initial markings M_{init} and a set of final markings

M_{final}. For instance, for Example 4, M_{init} consists only of the empty marking, whereas M_{final} consists of all (infinitely many) markings in which each of the two right-most places has at least one token, and all other places have no token. The *language* of the net is given by $\mathcal{L}(\mathcal{N}) = \{\rho_v \mid m \xrightarrow{\rho} m',\ m \in M_{init}, \text{ and } m' \in M_{final}\}$, i.e., the set of visible subsequences of accepted sequences.

The next example relates an observed event log with a DOPID, preluding to the conformance checking problem tackled in the next section.

Example 5. The event log described in Example 2 cannot be suitably replayed in the DOPID $\mathcal{N}$ of Fig. 1, due to two mismatches: according to $\mathcal{N}$, $\mathtt{o}_1$ must be shipped by car (as the preferred days are below 5), and with both products $\mathtt{p}_1$ and $\mathtt{p}_2$. This in turn requires that, before shipping, also product $\mathtt{p}_1$ must be picked.

Modelling Considerations. We briefly relate DOPIDs to the two reference formalisms that infuse Petri nets with case attributes (namely DPNs [23]) and multiple objects with complex synchronization mechanisms (namely sychronous proclets [11]). DPNs can be expressed as DOPIDs using an approach similar to the encoding of DPNs into Colored Petri nets in [23], using a "data place" for each variable x that contains a single token carrying the current value of x, and is linked to all transitions that read or write x.

Proclets are structurally encoded into DOPIDs following a schema similar to the one described for OPIDs [18]. Since DOPIDs provide full support for subset and exact synchronization, the crux is to refine the approach in [18] to reflect correlation sets, and their consequent usage for synchronization. This is done as follows: for every correlation set linking multiple child objects of the many side to the single parent object of the one side, a special "correlation" place holding the pairs is introduced. Upon synchronization, this correlation place is inspected to extract some or all the required pairs. This reconstructs and generalizes proclet synchronization, as one can now operate over the correlation place to define different regeneration strategies for the correlation set.

As for more general modelling languages, we leave as future work to provide a systematic formalization into DOPIDs. This appears to be a feasible route, building on previous encodings of artifact-centric and case-handling approaches into (extensions of) Petri nets [20,25].

5 Alignment-Based Conformance Checking for DOPIDs

We follow alignment-based approaches for object-centric processes [18,24], which relate trace graphs to model runs to find deviations. In the sequel, we consider a trace graph T_X and an accepting DOPID $\mathcal{N}$, assuming that the language of $\mathcal{N}$ is not empty. In our data-aware setting, moves also contain assignments:

Definition 8. (Moves). *A* model *move is a tuple in* $\{\gg\} \times ((\mathcal{A} \cup \{\tau\}) \times \mathcal{P}(\mathcal{O}) \times Assign^{\mathcal{V}})$, *a* log move *a tuple in* $(\mathcal{A} \times \mathcal{P}(\mathcal{O}) \times Assign^{\mathcal{V}}) \times \{\gg\}$, *and a* synchronous *move is of the form* $\langle\langle a, O_M, \alpha_M\rangle, \langle a', O_L, \alpha_L\rangle\rangle \in (\mathcal{A} \times \mathcal{P}(\mathcal{O}) \times Assign^{\mathcal{V}})^2$ *such*

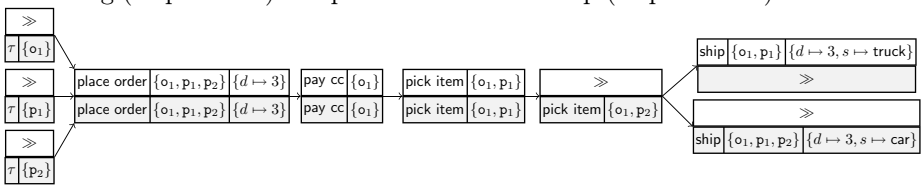
that $a = a'$ and $O_L = O_M$. The set of all synchronous, model, and log moves over T_X and $\mathcal{N}$ is denoted moves$(T_X, \mathcal{N})$.

In the object-centric setting, an alignment is a *graph* of moves G. We use the notions of log projection $G|_{log}$ and model projection $G|_{mod}$ as defined in [18, 24], but provide an intuitive explanation here: The log projection is the graph obtained from G by projecting nodes to their log part, while omitting model moves, i.e. nodes where the log component is $\gg$. Edges are as in G, except that edges are added that "shortcut" over model moves in G. The model projection is defined similarly. Next we define an alignment as a graph over moves where the log and model projections are a trace graph and a run, respectively.

Definition 9. (Alignment). *An* alignment *of a trace graph T_X and an accepting DOPID $\mathcal{N}$ is an acyclic directed graph $\Gamma = \langle C, B \rangle$ with $C \subseteq moves(T_X, \mathcal{N})$ such that $\Gamma|_{log} = T_X$, there is a run $\rho = \langle \langle t_1, b_1 \rangle, \ldots, \langle t_n, b_n \rangle \rangle$ with $\rho_v \in \mathcal{L}(\mathcal{N})$, and the model projection $\Gamma|_{mod} = \langle C_m, B_m \rangle$ admits a bijection $f \colon \{\langle t_1, b_1 \rangle, \ldots, \langle t_n, b_n \rangle\} \to C_m$ such that*

- *if $f(t_i, b_i) = \langle a, O_M, \alpha_M \rangle$ then $\ell(t_i) = a$, $O_M = range(b_i) \cap \mathcal{O}$, and $\alpha_M = \{x \mapsto d \mid x \in dom(b_i),\ b_i(x) = d,\ and\ type(x) \in \Sigma_{val}\}$;*
- *for all $\langle r, r' \rangle \in B_m$ there are $1 \leq i < j \leq n$ such that $f(t_i, b_i) = r$ and $f(t_j, b_j) = r'$.*

Example 6. Below is an alignment Γ for the log in Example 2 wrt. $\mathcal{N}$ in Example 4. The log (resp. model) component is shown on top (resp. bottom) of moves.



Note that a synchronous ship move is not possible by Definition 8 because the sets of involved objects would differ.

We adopt the cost function from [24], but extend it to account for mismatching data values. Other definitions are, however, possible as well.

Definition 10. (Cost). *The cost of a move is: (1) if $M = \langle \langle a_L, O_L, \alpha_L \rangle, \gg \rangle$ is a log move then $cost(M) = |O_L| + |dom(\alpha_L)|$, (2) if $M = \langle \gg, \langle a_M, O_M, \alpha_M \rangle \rangle$ is a model move then $cost(M) = 0$ if $a_{mod} = \tau$, and $cost(M) = |O_M| + |dom(\alpha_M)|$ otherwise, (3) if M is a synchronous move $\langle \langle a_L, O_L, \alpha_L \rangle, \langle a_M, O_M, \alpha_M \rangle \rangle$ then $cost(M)$ is the number of variables in $dom(\alpha_L) \cup dom(\alpha_M)$ for which α_L and α_M differ. For an alignment $\Gamma = \langle C, B \rangle$, we set $cost(\Gamma) = \sum_{M \in C} cost(M)$, i.e., the cost of an alignment Γ is the sum of the cost of its moves.*

E.g., Γ in Example 6 has cost 11, as it involves one log move (cost 4) and two non-silent model moves (costs 2 and 5). In fact, Γ is optimal:

Definition 11. (Optimal alignment). *An alignment Γ of a trace graph T_X and an accepting DOPID $\mathcal{N}$ is* optimal *if $cost(\Gamma) \leq cost(\Gamma')$ for all alignments Γ' of T_X and $\mathcal{N}$.*

The *conformance checking task* for an accepting DOPID $\mathcal{N}$ and a log L is to find optimal alignments with respect to $\mathcal{N}$ for all trace graphs in L.

SMT Encoding for Conformance Checking. An SMT encoding of the conformance checking task for a given DOPID $\mathcal{N}$ and trace graph T_X can be done in a similar way as for OPIDs [18]. For reasons of space, we focus on the differences.

First, in encoding-based conformance checking, it is essential to fix upfront an upper bound on the size of an optimal alignment Γ. For DOPIDs, we can exploit [18, Lemma 1]: DOPIDs differ from OPIDs in the presence of data and synchronization, but this does not affect the reasoning of that proof. We thus get an upper bound n on the number of nodes in the model projection of Γ and an upper bound K on the number of objects used in a transition. From T_X and K, we can get a finite set of objects O such that Γ uses only objects in O (up to renaming). Let m be the number of nodes in T_X.

The encoding uses the SMT variables from [18]:

(a) transition variables $\mathtt{T}_i$, $1 \leq i \leq n$, to encode the i-th transition in the run; (b) marking variables $\mathtt{M}_{i,p,\boldsymbol{o}}$ for every time point $0 \leq i \leq n$, every place p, and every vector $\boldsymbol{o}$ of objects with elements in O; (c) a variable $\mathtt{len}$ to encode the length of the run and (d) object variables $\mathtt{O}_{i,k}$ for all $1 \leq i \leq n$ and $0 \leq k \leq K$ to encode which objects populate inscriptions, and (e) distance variables $\delta_{i,j}$ to optimize the cost of the alignment. In addition, to keep track of data values, if X is the set of inscription variables of non-object type in $\mathcal{N}$, and M the maximal number of data values in tokens, we use (f) a data inscription variable $\mathtt{D}_{i,x}$ to represent the data value of x in the i-th transition, for all $1 \leq i \leq n$ and $x \in X$; and (g) a data store variable $\mathtt{S}_{i,p,\boldsymbol{o},l}$ to represent the l-th data value stored with token $\boldsymbol{o}$ in place p at instant i, for all $1 \leq l \leq M$, object vectors $\boldsymbol{o}$ over O, places p, and $1 \leq i \leq n$.

There are then two main differences in the encoding wrt. [18]. First, transitions guards need to be taken into account, similar to [12], using the data variables $\mathtt{D}_{j,x}$. Uninterpreted function symbols as well as numeric predicates and aggregation functions are natively supported by SMT solvers. Second, to model synchronization, in contrast to the subset synchronization employed in [18] it must be ensured that inscription variables from $\mathcal{V}^{=}_{list}$ are always instantiated by all matching tokens currently in the respective places. Details of the encoding can be found in [19]. Notably, we show that from a satisfying assignment to all constraints, an optimal alignment for $\mathcal{N}$ and T_X can be decoded.

Implementation. We extended the conformance checker CoCoMoT (https://github.com/bytekid/cocomot) to support DOPIDs, using the SMT solver Yices 2 as backend and the aforementioned encoding. We tested it on a series of examples that can be found in the repository. For Example 4 and traces of length in the

same scale as in the running example, conformance checking is done below one second.

6 Conclusions

We have introduced DOPIDs, a new process formalism that unifies modelling features of case-centric data-aware processes and object-centric processes, especially offering an object-centric paradigm with full synchronization and support for complex data. We also showed a novel operational approach leveraging the SMT technology to tackle alignment-based conformance checking for DOPIDs.In future work, we intend to conduct an experimental evaluation of this approach, and study discovery techniques for DOPIDs.

Acknowledgements. M. Montali was partially supported by the NextGenerationEU FAIR PE0000013 project MAIPM (CUP C63C22000770006) and the PRIN MIUR project PINPOINT Prot. 2020FNEB27. S. Winkler was partially supported by the UNIBZ project TEKE. A. Gianola was partly supported by Portuguese national funds through Fundação para a Ciência e a Tecnologia, I.P. (FCT), under projects UIDB/50021/2020 (DOI:10.54499/UIDB/50021/2020). This work was partially supported by the 'OptiGov' project, with ref. n. 2024.07385.IACDC (DOI: 10.54499/2024.07385.IACDC), fully funded by the 'Plano de Recuperação e Resiliência' (PRR) under the investment 'RE-C05-i08 - Ciência Mais Digital', measure 'RE-C05-i08.m04' (in accordance with the FCT Notice No. 04/C05-i08/2024), framed within the financing agreement signed between the 'Estrutura de Missão Recuperar Portugal' (EMRP) and FCT as an intermediary beneficiary.

References

1. van der Aalst, W.M.P.: Object-centric process mining: dealing with divergence and convergence in event data. In: Proceedings of the 17th SEFM (2019). https://doi.org/10.1007/978-3-030-30446-1_1
2. van der Aalst, W.M.P.: Toward more realistic simulation models using object-centric process mining. In: Proceedings of the 37th ECMS, pp. 5–13 (2023). https://doi.org/10.7148/2023-0005
3. van der Aalst, W.M.P.: Twin transitions powered by event data - using object-centric process mining to make processes digital and sustainable. In: Joint Workshop Proceedings of ATAED/PN4TT (2023)
4. van der Aalst, W., Berti, A.: Discovering object-centric Petri nets. Fundam. Informaticae **175**(1–4), 1–40 (2020). https://doi.org/10.3233/FI-2020-1946
5. van der Aalst, W., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. **53**(2), 129–162 (2005). https://doi.org/10.1016/J.DATAK.2004.07.003
6. Berti, A., Montali, M., van der Aalst, W.M.P.: Advancements and challenges in object-centric process mining: a systematic literature review. CoRR abs/2311.08795 (2023). https://doi.org/10.48550/ARXIV.2311.08795

7. Breitmayer, M., Arnold, L., Pejic, M., Reichert, M.: Transforming object-centric process models into BPMN 2.0 models in the PHILharmonicFlows framework. In: Proceedings of Modellierung 2024. LNI, vol. P-348, pp. 83–98 (2024). https://doi.org/10.18420/MODELLIERUNG2024_009
8. Calvanese, D., Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Formal modeling and SMT-based parameterized verification of data-aware BPMN. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 157–175. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_12
9. Cohn, D., Hull, R.: Business artifacts: a data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. **32**(3), 3–9 (2009)
10. Damaggio, E., Deutsch, A., Hull, R., Vianu, V.: Automatic verification of data-centric business processes. In: Proceedings of BPM 2011. LNCS, vol. 6896, pp. 3–16 (2011). https://doi.org/10.1007/978-3-642-23059-2_3
11. Fahland, D.: Describing behavior of processes with many-to-many interactions. In: Proceedings of PETRI NETS (2019). https://doi.org/10.1007/978-3-030-21571-2_1
12. Felli, P., Gianola, A., Montali, M., Rivkin, A., Winkler, S.: Data-aware conformance checking with SMT. Inf. Syst. **117**, 102230 (2023). https://doi.org/10.1016/J.IS.2023.102230
13. Felli, P., Gianola, A., Montali, M., Rivkin, A., Winkler, S.: Multi-perspective conformance checking of uncertain process traces: an SMT-based approach. Eng. Appl. Artif. Intell. **126**, 106895 (2023). https://doi.org/10.1016/J.ENGAPPAI.2023.106895
14. Felli, P., de Leoni, M., Montali, M.: Soundness verification of data-aware process models with variable-to-variable conditions. Fundam. Informaticae **182**(1), 1–29 (2021). https://doi.org/10.3233/FI-2021-2064
15. Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Delta-BPMN: A concrete language and verifier for data-aware BPMN. In: Polyvyanyy, A., Wynn, M.T., Van Looy, A., Reichert, M. (eds) Proceedings of BPM 2021. Lecture Notes in Computer Science, vol. 12875, pp. 179–196. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85469-0_13
16. Ghilardi, S., Gianola, A., Montali, M., Rivkin, A.: Petri net-based object-centric processes with read-only data. Inf. Syst. **107**, 102011 (2022). https://doi.org/10.1016/J.IS.2022.102011
17. Gianola, A.: Verification of Data-Aware Processes via Satisfiability Modulo Theories, Lecture Notes in Business Information Processing, vol. 470. Springer (2023). https://doi.org/10.1007/978-3-031-42746-6
18. Gianola, A., Montali, M., Winkler, S.: Object-centric conformance alignments with synchronization. In: Proceedings of the 36th CAiSE. LNCS, vol. 14663, pp. 3–19 (2024). https://doi.org/10.1007/978-3-031-61057-8_1
19. Gianola, A., Montali, M., Winkler, S.: Object-centric processes with structured-data and universal synchronization (extended version). CoRR abs/2505.15409 (2025). https://doi.org/10.48550/arXiv.2505.15409
20. Haarmann, S., Montali, M., Weske, M.: Refining case models using cardinality constraints. In: La Rosa, M., Sadiq, S., Teniente, E. (eds.) Proceedings of the 33rd CAiSE. pp. 296–310 (2021). https://doi.org/10.1007/978-3-030-79382-1_18
21. Hewelt, M., Weske, M.: A hybrid approach for flexible case modeling and execution. In: Proceedings of the Business Process Management Forum. LNBIP, vol. 260, pp. 38–54 (2016). https://doi.org/10.1007/978-3-319-45468-9_3

22. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. J. Softw. Maintenance Res. Pract. **23**(4), 205–244 (2011). https://doi.org/10.1002/SMR.524
23. de Leoni, M., Felli, P., Montali, M.: A holistic approach for soundness verification of decision-aware process models. In: ER. LNCS, vol. 11157, pp. 219–235 (2018). https://doi.org/10.1007/978-3-030-00847-5_17
24. Liss, L., Adams, J.N., van der Aalst, W.: Object-centric alignments. In: Proceedings of the ER (2023). https://doi.org/10.1007/978-3-031-47262-6_11
25. Lohmann, N., Wolf, K.: Artifact-centric choreographies. In: Service-Oriented Computing, pp. 32–46 (2010). https://doi.org/10.1007/978-3-642-17358-5_3
26. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2016). https://doi.org/10.1007/S00607-015-0441-1
27. Montali, M., Calvanese, D.: Soundness of data-aware, case-centric processes. Int. J. Softw. Tools Technol. Transf. **18**(5), 535–558 (2016). https://doi.org/10.1007/S10009-016-0417-2
28. Montali, M., Rivkin, A.: DB-Nets: on the marriage of colored petri nets and relational databases. Trans. Petri Nets Other Model. Concurr. **12**, 91–118 (2017). https://doi.org/10.1007/978-3-662-55862-1_5
29. Polyvyanyy, A., van der Werf, J.M.E.M., Overbeek, S., Brouwers, R.: Information systems modeling: language, verification, and tool support. In: Proceedings of the 31st CAiSE (2019). https://doi.org/10.1007/978-3-030-21290-2_13
30. Rosa-Velardo, F., de Frutos-Escrig, D.: Decidability problems in Petri nets with names and replication. Fundam. Informaticae **105**(3), 291–317 (2010). https://doi.org/10.3233/FI-2010-368
31. Snoeck, M., Verbruggen, C., Smedt, J.D., Weerdt, J.D.: Supporting data-aware processes with MERODE. Softw. Syst. Model. **22**(6), 1779–1802 (2023). https://doi.org/10.1007/S10270-023-01095-4
32. Sommers, D., Sidorova, N., van Dongen, B.: Aligning event logs to resource-constrained ν-petri nets. In: Proceedings of the 43rd PETRI NETS. LNCS, vol. 13288, pp. 325–345 (2022). https://doi.org/10.1007/978-3-031-06653-5_17
33. Terry Heath III, F.F., Boaz, D., Gupta, M., Vaculín, R., Sun, Y., Hull, R., Limonad, L.: Barcelona: a design and runtime environment for declarative artifact-centric BPM. In: Proceedings of the 11th ICSOC. LNCS, vol. 8274, pp. 705–709 (2013). https://doi.org/10.1007/978-3-642-45005-1_65
34. van der Werf, J.M.E.M., Rivkin, A., Polyvyanyy, A., Montali, M.: Data and process resonance - identifier soundness for models of information systems. In: Proceedings of the PETRI NETS (2022). https://doi.org/10.1007/978-3-031-06653-5_19

Cloud Systems

ThreatTrace: Cyber-Attack Detection Through Trace Abstraction and Soft Clustering

Andrzej Janusz[1,2,3(✉)], Savandi Kalukapuge[1], and Moe Thandar Wynn[1,2]

[1] School of Information Systems, Queensland University of Technology, Brisbane, Australia
{andrzej.janusz,s.kalukapuge,m.wynn}@qut.edu.au
[2] Centre for Data Science, Queensland University of Technology, Brisbane, Australia
[3] Institute of Informatics, University of Warsaw, Warsaw, Poland

Abstract. The growing sophistication of cyber threats has raised the importance of providing a proper level of cybersecurity to networks and interconnected devices. Process mining provides methods to analyse sequences of activities performed by operators or programs on such devices to detect cyber-attacks from a novel perspective, i.e., as continuous processes composed of observable system events. However, processes like that, especially in domains such as the cybersecurity of the Internet of Things (IoT) or network devices, are usually characterised by large variability and complexity, which makes them difficult to model. We propose a method called ThreatTrace for simplifying and creating embeddings of complex process traces that can often be observed in event logs from such domains. The novelty of ThreatTrace stems from integrating process mining techniques into the cyber-threats discovery workflow. It combines process trace abstraction, embedding, and soft clustering to generate compact process variant representations that preserve information about patterns of interest, e.g., traces of potential cyber-attacks. Our experiments show that such information extracted from event logs improves the detection of cyber-attacks over approaches that do not consider the process perspective, particularly in the context of IoT device and network cybersecurity.

Keywords: Process Mining · Cybersecurity · Trace Abstraction · Process Trace Embeddings · Soft Clustering of Traces

1 Introduction

The cybersecurity landscape is increasingly critical due to our reliance on connected technology. Process mining [1] offers a fresh perspective by analysing

We would like to thank QUT Centre for Data Science for supporting this research through a travel grant.

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 205–222, 2025.
https://doi.org/10.1007/978-3-031-94571-7_12

sequences of actions performed by users or system programs as distinct process instances. This approach helps to uncover hidden patterns and detect deviations that may signal malicious activity. In an ideal scenario, network devices, particularly IoT devices, exhibit regular behaviour. When a deviating activity is observed, it may be a potential indicator of cyber-attacks. In practice, however, cybersecurity-related processes are challenging to analyse due to their inherent complexity and variability [17]. Process mining methods can empower cybersecurity analysts to trace and better understand attack paths, enabling them to undertake proactive cybersecurity measures.

Understanding cyber-attacks as processes allows for a more nuanced comprehension of the intricate sequences of events that lead to security breaches. However, a significant challenge in viewing a cyber-attack as a process lies in identifying the appropriate level of abstraction. Without proper aggregation, cyber-attacks that share similar characteristics may be treated as disparate incidents, hindering effective analysis and detection efforts. In real-world systems, the presence of multiple concurrent processes further compounds this challenge. While some processes may exhibit indicators of malicious activity, others may operate entirely within the bounds of normal behaviour. The ability to detect cyber-attacks with the shortest possible lags, i.e., in near real-time, amidst the noise of routine operations in a system, is essential [9].

In this research, we aim to address the above-mentioned challenges. We propose a novel method, called ThreatTrace, for the analysis of cybersecurity event data that allows accurate detection of cyber-attacks with minimal delays. ThreatTrace combines a frequent pattern-based trace abstraction technique with embedding learning and soft clustering. Once trained on historical data that includes examples of normal and malicious behaviours, ThreatTrace allows processing event streams and inferring cyber-threats in fixed-size time windows of event logs. Although various techniques in the cybersecurity domain use rules and patterns to identify possible cyber-attacks [13], there has been a limited focus on using trace abstraction methods from the field of process mining to address this challenge. The novelty of our approach stems from the use of a new stochastic model in which traces are represented by multivariate Gaussian distributions defined in the activity embedding space. On the one hand, it addresses the challenges related to the complexity of processes commonly observed in the cybersecurity context. On the other hand, unlike other trace embedding methods, it facilitates continuous monitoring of cyber-threats and processing new traces that were not observed in the historical training data. It can be viewed as a process trace abstraction technique that creates a concise representation of event sequences in cybersecurity logs. We show that traces in this representation can be clustered using standard methods. We apply the fuzzy c-means (FCM) algorithm to extract trace-cluster membership values and use them as higher-level features. We then train machine learning (ML) models to detect cyber-attacks in newly observed traces. Our experiments show that using information extracted from event logs improves the detection of cyber-attacks over approaches that do not consider the process perspective.

In the following Sect. 2, we discuss related work. Then, in Sect. 3, we explain the major steps in our workflow, focusing on how ThreatTrace abstracts process traces and extracts higher-level features for the detection of cyber threats. In Sect. 4, we describe datasets, which we use to evaluate our approach and discuss the results of our experiments. Then, in Sect. 5, we reflect on the main limitations of ThreatTrace and point to possible directions for future research. Lastly, Sect. 6 concludes the paper.

2 Related Work

Cybersecurity in IoT and network environments presents unique challenges due to the volume, velocity, and variety of data generated by interconnected devices. Macák et al. [17] identified seven main research directions where process mining is applied to cybersecurity: security of Industrial Control Systems, security of smartphones, web-application security, network traffic security, attack inspection, outlier user behaviour detection, and fraud detection. They note that traditional cybersecurity detection approaches often rely on signature-based systems or anomaly detection techniques that analyse individual events in isolation [17,30]. While effective for known attack patterns, these methods struggle with previously unseen threats and often produce high false positive rates [5,15]. Furthermore, they typically lack behavioural interpretability, making it difficult for security analysts to understand attack pathways and take proactive measures.

Current intrusion detection systems (IDS) primarily focus on isolated events rather than sequences of activities [3], making them ineffective against sophisticated multi-stage attacks. Ahmad et al. [3] highlight that while typical machine learning approaches have improved detection capabilities, they fail to capture the temporal relationships between events that characterise modern attack patterns. Similarly, Apruzzese et al. [4] demonstrate that adversarial evasion techniques can easily circumvent traditional detection methods by manipulating individual events while maintaining the overall attack flow. This limitation is further emphasised by Czerwiński et al. [9], who show that contextual information about event sequences significantly improves detection accuracy compared to traditional feature-based approaches. Additionally, Gómez et al. [13] note that real-time attack pattern recognition requires understanding the sequential progression of events, which is missing in conventional cybersecurity solutions. These limitations create a significant gap in cybersecurity defence capabilities that process mining techniques are uniquely positioned to address. By modelling attacks as processes rather than isolated incidents, process mining enables security analysts to identify attack patterns as they unfold across time and system components. This process-oriented perspective offers a promising solution to the challenge of detecting complex, evolving attacks that conventional IDS approaches consistently miss.

A significant challenge in modelling cyber-attacks as processes is the high variability and complexity of event sequences, which can lead to so

called *spaghetti* process models. Van Zelst et al. [29] survey event abstraction approaches used in various applications to simplify event logs while preserving relevant patterns. They also propose a taxonomy of event abstraction methods that includes both supervised and unsupervised approaches. Folino et al. [12] propose a trace abstraction method that involves clustering of low-level events. Mannhardt and Tax [18] combine frequent pattern discovery with the construction of local process models to adapt the granularity of events and create abstracted process traces.

Trace and event clustering methods have proven as an effective tool for trace abstraction [29]. Traditional hard clustering assigns each trace to exactly one cluster [27], which can potentially miss traces or activities that could occur in both normal and malicious behaviours. This limitation is particularly problematic for detecting sophisticated attacks that partially mimic legitimate operations. Soft clustering allows traces to belong to multiple clusters with different degrees of membership [6], making it more suitable for evolving attack pattern detection, as highlighted by Xu and Wunsch [27].

For capturing and representing complex process behaviours, trace embedding techniques have emerged as powerful tools. However, existing approaches present significant limitations for cybersecurity applications. NLP-inspired methods such as trace2vec [16] have been applied to process mining tasks, including trace clustering, but they struggle with generalising to unseen trace variants – a common requirement in cybersecurity environments. RNN-based representations [11] and transformer-based embeddings [23] can capture sequential patterns but are computationally intensive and considerably reduce the interpretability of the cyber-threat detection systems. GloVe [21], on the other hand, offers a robust alternative for generating activity-level embeddings that can be aggregated to represent traces. Studies have demonstrated that GloVe consistently outperforms word2vec-based approaches in capturing semantic relationships within sequential data [21,25].

ThreatTrace integrates the above-stated complementary approaches, i.e., process trace abstraction, embedding learning, and soft clustering, to address the challenges of cyber-attack detection in IoT and network environments. Its distinctive contribution is a novel stochastic model representing traces as multivariate Gaussian distributions in the activity embedding space. We show that such a representation not only enhances cyber-threat detection accuracy but is also computationally efficient and can be used for monitoring complex processes in near-real time. As identified by Macák et al. [17], while most process mining applications in cybersecurity analyse past events with limited real-time capabilities, our approach enables both comprehensive modelling and near real-time monitoring of complex processes, filling a significant gap in current research.

3 Constructing Abstractions of Traces in ThreatTrace

When dealing with real-world processes in complex domains such as cybersecurity, standard approaches to process modelling may face the problem of large

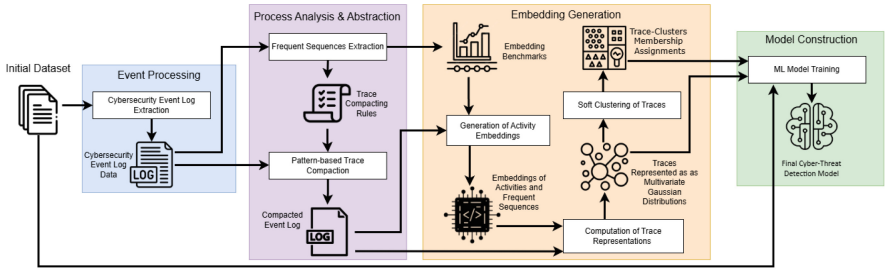


Fig. 1. A high-level overview of the ThreatTrace workflow.

variability in the observed process traces. As a result, constructed process models can become overly large and difficult for domain experts to analyse. ThreatTrace addresses this issue by constructing abstractions of process traces and concisely representing them as multivariate Gaussian distributions over the activity embedding space. Figure 1 illustrates the ThreatTrace workflow. It consists of four main phases: event processing, process analysis and abstraction, and embedding generation, followed by model construction. The workflow begins with event log extraction from the initial dataset. In the process analysis and abstraction phase, frequent sequences are extracted and used to create trace abstracting rules, which are then applied to simplify the event logs. The embedding generation phase is the core of ThreatTrace. Activity embeddings are generated and evaluated using automatically created benchmarks. These embeddings, along with embeddings of frequent sequences, are used to construct trace representations, i.e., the traces are modelled as multivariate Gaussian distributions defined in the space of activity embeddings, and processed through soft clustering to determine cluster memberships. Finally, in the model construction phase, these derived features are used to train ML models for cyber-attack detection.

3.1 Computing Abstractions of Process Variants

Our overall goal for this step is to reduce the number of variants that are considered different, even though they are essentially the same from a cyber-attack point of view, while preserving outliers that may be indicators of potential attacks. Our secondary goal is to capture the most substantial examples of the *directly follows* relation that we want to reflect in constructed embeddings.

Let us denote the j-th process trace in available data by a sequence of action symbols $T^j = [X_1^j, X_2^j, \ldots, X_{N_j}^j] \in T$, where each X_i^j is a symbol corresponding to one of the possible actions from the set A, i.e., $X_i^j \in A = \{A_1, \ldots, A_n\}$. Let us also assume that we have an ordered list of trace abstracting rules $R = [R_1, \ldots, R_m]$, where each rule R_i indicates a substitution of a subsequence $[Y_1, Y_2]$ with a new symbol $A'_k \in A'$. For simplicity, we will denote such a rule by an implication $[Y_1, Y_2] \rightarrow A'_k$. It is worth noting that symbols $Y_1, Y_2 \in A \cup A'$.

The original process traces from the available data can be abstracted by recursively applying rules R_i to each trace $T^j \in T$ in the order indicated by R. For example, if we consider a list of rules $R = [R_1, R_2, R_3]$, such that:

$$R_1 = ([a, b] \rightarrow b'),\ R_2 = ([a, a] \rightarrow a'),\ R_3 = ([a', a] \rightarrow a'),$$

then a process trace

$$t = [a, a, b, a, a, a, b, c, a, a, a]$$

would be abstracted to:

$$t_a = [a, b', a', b', c, a'] \ .$$

After the application of R_1 trace t would be reduced to: $[a, b', a, a, b', c, a, a, a]$. Then, the application of R_2 would further reduce it to: $[a, b', a', b', c, a', a]$, and the final sequence of symbols would be obtained by the application of R_3.

Rules from R can be automatically extracted from data using an arbitrary frequent sequence mining algorithm, such as SPADE [28], or they could be provided by domain experts. It is typically beneficial to sort the list R decreasingly by rule support. Rules that are closer to the beginning of the list are likely to contribute more to the reduction of process traces. However, more advanced optimisation heuristics, such as the genetic algorithm [24], can also be used to find the ordering of abstracting rules that yields the highest trace compression ratio.

3.2 Computation of Activity and Trace Embeddings

Trace embedding techniques have been used by other researchers to facilitate the clustering of process traces [16]. One major limitation of the *trace2vec* technique used in that study is that it is not able to infer the representation of new, previously unobserved trace variants. Traces like that are common in practice, especially in domains such as cybersecurity, where the number of different possible activities can be large. To bypass this limitation, we propose to compute the embeddings on the activity level and represent traces as aggregations – clusters modelled by Gaussian distributions of activity embeddings. In this way, ThreatTrace can embed newly observed traces as long as the set of possible activities remains fixed. Moreover, we can apply this approach in an event streaming setup to facilitate cyber-threat detection in near real-time.

Another major difference with the *act2vec* and *trace2vec* methods proposed in [16] is that the embedding learning algorithm used in ThreatTrace is based on GloVe [21]. Several studies demonstrated that Glove consistently outperforms *word2vec* used in *act2vec* and *trace2vec* in various NLP tasks, such as word analogies, prediction of synonyms, NER, and text classification [21,25]. We train it on event log data by considering consecutive action codes in each trace. To guide the embedding learning process, we add special tokens to the list of activities, which indicate whether the trace was observed during the cyber-attack. We use standard settings for the context window size and other GloVe parameters. However, we tune the embedding size using a set of manually extracted benchmark similarity queries that we derive from the frequent patterns discovered during the initial trace abstraction.

The construction of benchmark queries is motivated by the word similarity task, commonly used as an evaluation task for natural language word embeddings. A desired feature for word embeddings is their ability to represent semantic patterns as linear translations [20]. For instance, we would like to see that vec("Warsaw") - vec("Poland") + vec("Australia") is more similar to vec("Canberra") than to any other word embeddings, especially for words representing names of other capital cities. Since the abstracting patterns in ThreatTrace are sequences of individual activities, and the embeddings are computed for both the individual activities and their sequences, we can construct and evaluate similar algebraic expressions. For example, if we have the embeddings for $\{a, b, c, [a, b], [a, c]\} \subseteq A \cup A'$, we would expect that

$$vec([a, b]) - vec(b) + vec(c) \simeq vec([a, c]),$$

where $\simeq$ denotes the similarity relation. We extract multiple benchmark similarity queries using templates like the one above. We compute the average similarity and rank of expected outcomes and use this indicator to select the optimal vector size for the activity embeddings. In this way, we avoid conducting multiple train-test cycles, which could be time-consuming and lead to overfitting.

Finally, having the embeddings for activities and activity sequences, we represent traces as set of the corresponding embedding vectors. We model these sets as multivariate Gaussian distributions. Trace embeddings are the descriptions of such sets, which can be regarded as clusters of observed activities, i.e., for each trace, ThreatTrace stores the centroid and a covariance matrix of the distribution. Note that it is often impractical to store the whole covariance matrix. For the sake of conducted experiments, we only consider its diagonal, i.e., the vector of variances in each dimension. Thus, the vector representation for a trace T^j in ThreatTrace can be denoted as:

$$vec(T^j) = (\mu_1^j, \ldots, \mu_k^j, \sigma_1^j, \ldots, \sigma_k^j) \tag{1}$$

where μ_i^j is the average of the i-th dimension of activity embeddings from the trace T^j, i.e.,

$$\mu_i^j = \frac{1}{N_j} \sum_{l=1}^{N_j} vec_i(X_l^j) \tag{2}$$

and σ_i^j is the standard deviation of the i-th dimension of activity embeddings from the trace T^j, i.e.,

$$\sigma_i^j = \sqrt{\frac{1}{N_j - 1} \sum_{l=1}^{N_j} \left(vec_i(X_l^j) - \mu_i^j \right)^2} = \sqrt{\frac{1}{N_j - 1} \sum_{l=1}^{N_j} \left(vec_i(X_l)^2 - (\mu_i^j)^2 \right)} \tag{3}$$

In the notation above, we denote the i-th dimension of the embedding for activity X_l as $vec_i(X_l)$. From the Formulas 2 and 3, we see that to facilitate the aggregation of activity embeddings and trace representations, instead of the vector

(1), it is more practical to store the following vector of sums:

$$vec^*(T^j) = (N_j, \sum_{l=1}^{N_j} vec_1(X_l), \ldots, \sum_{l=1}^{N_j} vec_k(X_l), \sum_{l=1}^{N_j} vec_1(X_l)^2, \ldots, \sum_{l=1}^{N_j} vec_k(X_l)^2) \tag{4}$$

Such a representation is additive, making it trivial to combine two or more partial traces into a single trace representation. This characteristic is vital, e.g., when dealing with streams of system events. Additionally, the same representation can be used to represent whole trace clusters and facilitate the cluster analysis task.

3.3 Soft Clustering of Process Variants

After the computation of the process trace representations, the next step in ThreatTrace is the cluster analysis of process variants. In our approach, individual process traces are modelled as Gaussian-distributed clusters of embeddings of activities and activity sequences. Thus, a natural metric for the computation of distances between the traces would be the Wasserstein distance [8]. If we denote the Wasserstein distance between two normal distributions by $d_W(N(\overline{\mu^1}, V^1), N(\overline{\mu^2}, V^2))$, were $\overline{\mu^j}$, are vectors of means, and V^j are the corresponding covariance matrices, then:

$$d_W(N(\overline{\mu^1}, V^1), N(\overline{\mu^2}, V^2)) = \sqrt{||\overline{\mu^1} - \overline{\mu^2}||_2^2 + ||\sqrt{V^1} - \sqrt{V^2}||_{Frobenius}^2} \tag{5}$$

Since in our representation of traces (Formula 1), we consider only the diagonal of the covariance matrix, the Wasserstein distance formula simplifies to:

$$\begin{aligned} d_W(T^1, T^2) &= \sqrt{||\overline{\mu^1} - \overline{\mu^2}||_2^2 + ||\overline{\sigma^1} - \overline{\sigma^2}||_2^2} \\ &= \sqrt{\sum_{i=1}^{k} (\mu_i^1 - \mu_i^2)^2 + \sum_{i=1}^{k} (\sigma_i^1 - \sigma_i^2)^2} \\ &= d_E\left(vec(T^1), vec(T^2)\right) \end{aligned} \tag{6}$$

where $d_E\left(vec(T^1), vec(T^2)\right)$ is the Euclidean distance between vector representations of the traces T^1 and T^2. Equation (6) allows us to cluster traces using the Euclidean distance and standard algorithms, such as *k-means*. However, we propose a more refined approach. Instead of dividing traces into disjoint sets, we apply the *fuzzy c-means* (FCM) algorithm [14] and compute their membership degrees to the identified clusters. FCM minimises the following objective function:

$$\begin{aligned} L(W, C) &= \sum_{j=1}^{|T|} \sum_{k=1}^{|C|} w_{j,k}^m ||vec(T^j) - c_k||^2, \\ w_{j,k} &= \frac{1}{\sum_{i=1}^{|C|} \left(\frac{||vec(T^j) - c_k||}{||vec(T^j) - c_i||} \right)^{\frac{2}{m-1}}} \end{aligned} \tag{7}$$

where $W = w_{j,k} \in [0,1], j = 1,\ldots,|T|, k = 1,\ldots,|C|$ are cluster membership degrees for each trace and cluster, C is the set of identified cluster centroids, and $m \in (1,\infty)$ is a constant called the fuzziness degree. By using the online version of the FCM algorithm [6] in combination with our internal representation of process traces, we can easily apply ThreatTrace in the streaming setting for continuous monitoring of cyber threats.

4 Dataset Descriptions and Experimental Results

In our experiments, we consider two datasets from the cybersecurity domain. The first one is related to the detection of malicious activity in the operations of IoT devices. The second one considers the recognition and scoring of suspicious network activity. The datasets come from our industry partners and were previously used in international data science competitions organised at the KnowledgePit.ai web platform. We provide a processed version of those datasets, as well as the source code of conducted experiments in our GitHub repository[1].

4.1 Cyber-Attacks on IoT Devices

Our first case study uses a dataset collected from an IoT device, specifically a Raspberry Pi, as well as other devices that were responsible for generating and executing HTTP traffic and attacks [9]. It was created using an experimental environment developed in a project conducted by three companies, i.e., EFIGO, EMAG, and QED Software [2], focused on the cybersecurity of IoT devices. To obtain a sample that contained both regular and anomalous device operations, the authors modelled the normal operation of the device and conducted cyber-attacks by exploiting some known system vulnerabilities. This involved the generation of typical network traffic and triggering standard system processes. It also required the manual design and automatic execution of scenarios for two different types of cyber-attacks on the device. More details about this experimental environment and the data generation process can be found in [9].

The raw data was collected from the monitored devices as system audit logs. These logs were parsed and transformed into separate CSV tables corresponding to 60-second time windows (20 044 files in total). We refer to each such file as a single data case. The files were automatically labelled using information about the timings of the scheduled attacks. Different types of attacks were not distinguished – only the binary information about the presence of an attack was kept. The distribution of this binary target variable is highly imbalanced. Only 698 files were collected during cyber-attacks on the device (3.48%). Such a class imbalance is typical in the cybersecurity domain. This dataset was divided into two sets of files. Training data, containing 15 027 CSV tables, and test data, containing 5017 tables. The original data division in the competition was based

[1] The supplementary materials and source code are available at https://github.com/janusza/ThreatTrace-Cyber-Attack-Detection.

on time, i.e., test tables corresponded to time windows strictly after the training data. We keep this division in all our experiments.

The rows of each CSV table correspond to individual system events recorded in the corresponding period. Events were described by 40 attributes, including information such as the event timestamp, process ID (PID), system call (action type), whether the system call was successful, and the process path. The attributes also included higher-level features extracted from the logs, such as counts of system kernel calls or the number of active services. We show an exemplary snippet of one of the CSV tables in Appendix A included in supplementary materials(see footnote 1). We focus on system calls associated with particular PIDs, i.e., we define an activity as a combination of the system call, user group, and the indicator of whether the system call was successful. We associate sequences of activities with the same PID in the 60-second period with a single instance (a case) of the investigated process. In this way, we have traces of multiple observed processes in each of the available CSV tables. The average number of process traces in a single table is 32.36, and the average number of events associated with a trace is 59.21. The total number of process traces in the training and test data is 648 539, and they are composed of 38 399 367 events. There are 77 different basic action types in the data and we extend this set to 193 activities through mining frequent sequences and applying abstracting rules. Our experiments investigate the impact of information about the underlying processes, extracted using ThreatTrace, on the performance of cyber-threat detection models that were used as the baseline in the data science competition FedCSIS 2023 Challenge [9].

It is worth noting that the original dataset used in the challenge was flawed. The operating system on the monitored IoT device was never restarted during the data generation. As a result, the scheduled attacks were performed using programs that were given the same PIDs for every execution. Consequently, even though the training and test data corresponded to different time periods, the cyber-attacks could have been easily identified by the presence of specific PIDs in the audit logs from a given time window. Of course, such facilitation would be unrealistic in a real situation and can be considered as a severe data leak – in real life, each cyber-attack instance would be associated with a different set of randomly generated PIDs. Thus, in our experiments, we make independent PID assignments in every considered time window.

4.2 Prioritisation of Alerts for Suspicious Network Event

The second case study is based on data provided by the Security on Demand (SOD) company (currently DeepSeas) for IEEE BigData 2019 Cup [15]. The task in this competition was to distinguish between truly suspicious events and false alarms within the set of alerts derived from network traffic data that Security Operations Center (SOC) team members at SOD analyse on an everyday basis.

The dataset from this competition is still available at KnowledgePit.ai. It is composed of three tables. The first one contains 59 427 rows corresponding to

so-called, *threatwatch alerts* investigated by the SOC team at SOD during the period between October 1, 2018, and March 31, 2019. Alerts in this table are described by 61 columns which represent information that is available to security operators and analysts during the investigation of suspicious events. Additionally, for each row, there was a label indicating whether the associated alert was considered "serious" by the SOD's analysts. Similarly to the first dataset, the target class in this dataset is highly imbalanced with only 5.76% positive cases. The competition organisers divided this data into training and test sets. The test set covered the alerts investigated in the last two months of the data collection period, and we kept this division in our experiments.

The second and third parts of the data are composed of two types of log files. We use them jointly in our experiments to extract features and learn representations of process traces. The second data part contains so-called *localized alert* logs, i.e., sequences of alerts related to individual *threatwatch alerts*. Events in this log correspond to alerts generated by expert system rules developed by the Threat Reconnaissance Unit at SOD [15]. In total, this data part contains 8 690 704 events described by a mixture of 20 numeric and symbolic features. The third part of the dataset contains a comprehensive event log of all network events registered by SOD's data collector devices within 24 h, counting back from the corresponding *threatwatch alert's* timestamp. Such event logs can be extremely large - our dataset has 4 384 234 352 events described by 26, mostly symbolic, features (430GB of data). The complexity of this event log is also reflected in the large number of different activity types and the variability of process traces. Originally, there were 323 action types in the data and after mining frequent sequences and applying trace abstracting rules, this set was extended to 772 action types.

4.3 Experimental Results

In this subsection, we present the results of experiments conducted to verify how the information extracted using ThreatTrace from event log data impacts the performance of cybersecurity threat detection models. We use datasets described in previous subsections and compare the quality of two prediction algorithms, i.e., the LASSO-regularised logistic regression (GLMNet) implemented in the *glmnet* library [22] and the decision tree gradient boosting model (XGBoost) implemented in the *xgboost* library [7]. While gradient boosting can be regarded as state-of-the-art in the tabular data classification task, logistic regression is still a common choice due to its high power efficiency. Moreover, the simplicity and low computational cost of inference make the logistic regression suitable for operating on mobile IoT devices. The baselines used in the comparison come from the data science competitions that are the source of our datasets [9,15]. We also added a reference model that is based on the feature extraction procedure inspired by descriptions from post-competition publications.

We trained the prediction models with the same hyperparameter setting for each of the compared representations. In the case of GLMNet, we set the *alpha* value to 1.0, i.e., only the LASSO regularisation was used. The *lambda* value

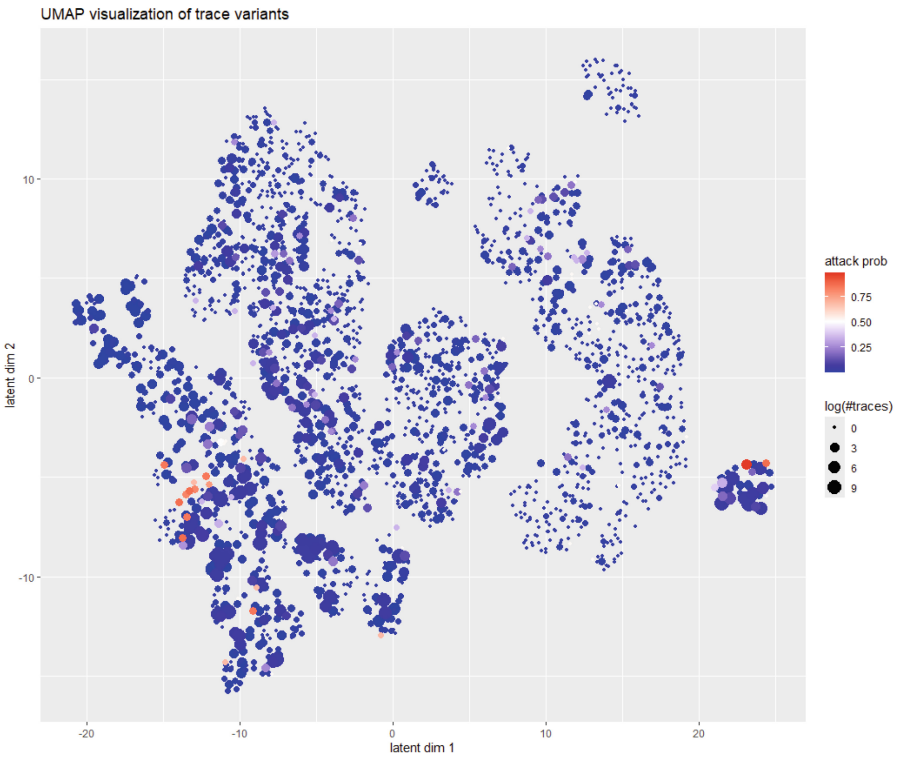


Fig. 2. UMAP visualisation of abstracted process variants from the FedCSIS 2023 Challenge dataset with a cyber-attack conditional probability heatmap.

was automatically tuned during the model fitting based on the results of 10-fold cross-validation on training data. The gradient boosting model used 2000 trees whose maximal depth was set to 10. The regularisation parameters *alpha* and *lambda* were set to 20.0 and 5.0, respectively. Additionally, to avoid overfitting, column and case sub-sampling were used with sampling rates of 0.8 and 0.95. These settings were chosen through the grid search for the representation that was obtained using a feature extraction approach inspired by descriptions of the corresponding competition baselines.

We tested ThreatTrace with three different embedding sizes. Their selection was made based on the results of the embedding evaluation procedure described in Subsect. 3.2. The evaluation results suggested that increasing the dimensionality of activity embeddings is likely to improve their quality. However, it also brings diminishing returns. This observation is consistent with the final evaluation outcomes shown in Table 1. We visualised abstracted process traces from both datasets using the UMAP algorithm [19]. In Figs. 2 and 3, the heat-map expresses the probability of the positive class conditioned on the presence of the corresponding abstracted trace variant. They were obtained using the 32-

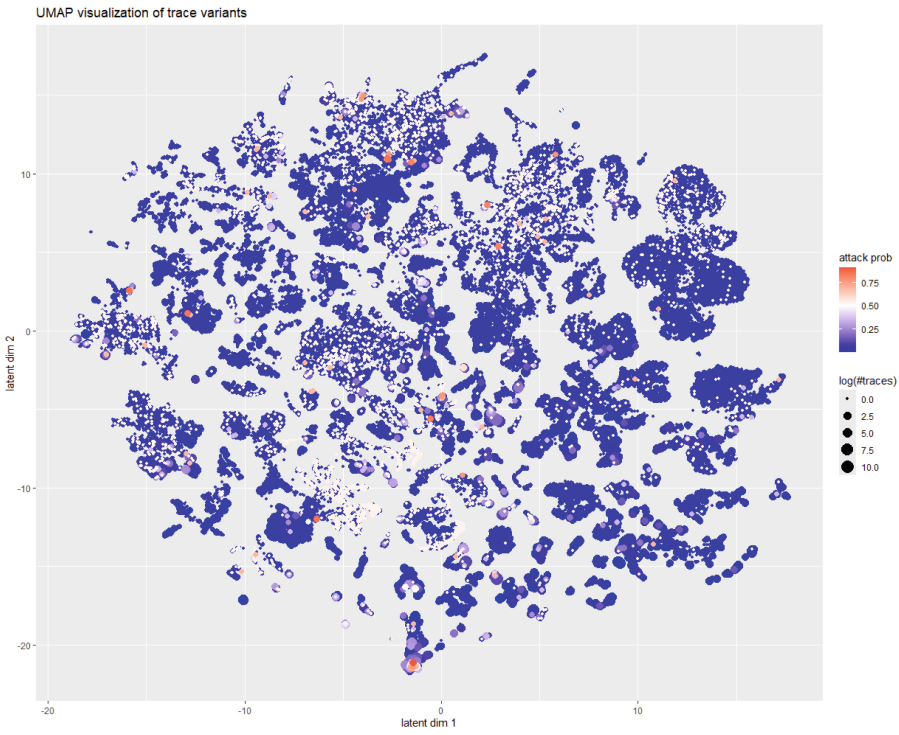


Fig. 3. UMAP visualisation of abstracted process variants from the IEEE BigData 2019 Cup dataset with a heatmap of the conditional probability of positive cases.

dimensional activity embeddings for the FedCSIS 2023 Challenge data and 40-dimensional embeddings for the IEEE BigData 2019 Cup data. We notice that variants associated with high cyber-attack probabilities are usually located in the boundary regions of local variant clusters. We also notice that for both datasets the initial stage of trace abstraction using the abstracting rules significantly reduced the overall number of trace variants. For the first dataset, the reduction was by 21.69% (from 4119 to 3226), and for the second dataset, the reduction was by 26.93% (from 170 259 to 124 400). Finally, the evaluation results from Table 1 confirm that ThreatTrace can improve the performance of ML models in detecting cyber threats. The statistical significance of these results was checked and reaffirmed using the DeLong test [10] for comparing ROC curves.

5 Limitations and Future Works

Although we have demonstrated that ThreatTrace enables learning useful representations of complex processes and facilitates training of accurate anomaly detection models, it should not be regarded as a one-size-fits-all solution. Our

Table 1. Results of two prediction models for various data representations and the baselines reported in [9] and [15] for FedCSIS 2023 Challenge and IEEE BigData 2019 Cup data, respectively. In addition to the ROC AUC values, the column *embedding size* indicates the sizes of activity embeddings used in ThreatTrace.

FedCSIS 2023 Challenge data			
representation type	embedding size	GLMnet	XGBoost
baseline from [9]	–	–	0.6910
basic feat. extraction	–	0.5400	0.8913
ThreatTrace	4	0.7123	0.9067
ThreatTrace	16	0.7393	0.9245
ThreatTrace	32	0.7459	0.9395
IEEE BigData 2019 Cup data			
representation type	embedding size	GLMnet	XGBoost
baseline from [15]	–	–	0.8704
basic feat. extraction	–	0.8351	0.9014
ThreatTrace	8	0.8675	0.9194
ThreatTrace	24	0.8791	0.9284
ThreatTrace	40	0.8807	0.9273

approach to modelling traces through abstracting activity sequences and associating them with Gaussian-distributed clusters is motivated by the continuous (i.e., without a well-defined beginning and end, with many possible activity types) and inherent stochastic nature of the underlying processes. As such, ThreatTrace may not be suitable for the analysis of typical business process models.

Moreover, ThreatTrace captures the *directly follows* relation between activities through the identification of frequent sequential patterns and the use of a trace abstracting technique. This approach, however, doesn't allow us to identify global dependencies between activity sequences, such as the *eventually follows* relation. This is caused by the fact that when constructing representations of individual traces from activity embeddings, we neglect their ordering. It is worth noting that in ThreatTrace, global dependencies, such as the *eventually follows* relation, are partially reflected in the activity embeddings. The GloVe algorithm [21] uses a fixed context window to count co-occurrences of activities. Thus, the proximity of activities in a trace is captured in their representation. However, we acknowledge that this problem requires a more thorough investigation in future research. One possible way to address this issue is to modify the embedding learning algorithm to consider the global position of activities in traces. In NLP, it is commonly done by using positional encoding techniques [26], and we believe that a similar approach can be adapted by ThreatTrace.

We also consciously simplify the trace representation by only considering the diagonal of the covariance matrix during the aggregation of activity embeddings. In practice, such simplification is equivalent to assuming the independence of embedding dimensions, which is usually not true. It limits the shapes of activity embedding clusters that we can accurately summarise with Gaussian distributions. To consider whole covariance matrices, we would have to keep track of the sums of all products of embedding dimension pairs. As a result, maintaining the trace representation would require $O(k^2)$ computations and memory, and it would be impractical for larger k values. We noticed, however, that increasing the dimensionality of embeddings after a certain point brings diminishing returns. In future, it would be worthwhile to explore the possibility of using smaller activity embeddings and full covariance matrices.

Finally, we acknowledge the need for further validation of ThreatTrace with diverse datasets, including different data modalities such as packet capture files, to increase its robustness and applicability across various environments. In particular, we see the need to further explore and compare different existing algorithms for trace abstraction, including methods for the automatic extraction of sequential patterns from data and optimise the sorting of rules used for trace abstraction. We would also like to investigate more systematic methods for including other perspectives on the event log that are typical in the BPM domain. Combined with previously mentioned ideas, this research direction holds the potential to further refine our methodology and contribute to the ongoing efforts to advance cybersecurity practices.

6 Concluding Remarks

Our research introduces a promising direction for enhancing cybersecurity analysis in network and IoT ecosystems. It is strongly inspired by the unique perspective on event log data offered by process mining. As a result, we proposed a new trace representation method that is suitable for representing complex processes. By analysing sequences of activities performed by system programs and network events, ThreatTrace captures the *directly follows* relation, commonly considered in process mining applications. We demonstrated the benefits of using ThreatTrace to detect cyber-attacks on IoT devices and recognise truly suspicious network activities. Our method combines frequent pattern-based trace abstracting with a novel way of representing traces of continuous processes as multivariate Gaussian distributions defined in the activity embedding space. From a theoretical perspective, we showed that clustering such distributions using the Wasserstein distance is equivalent to clustering our representations in the Euclidean space, which improves ThreatTrace efficiency and facilitates its applications in a streaming setting. We used this fact to construct soft clusters of process traces and utilise the fuzzy cluster membership information as features for predictive models. Finally, through our experiments, we demonstrated that ThreatTrace

helps to improve the quality of ML models for detecting cybersecurity threats. By leveraging the process view of cyber-systems, we pave the way for proactive detection and mitigation of cyber threats, ultimately fostering safer and more resilient systems.

References

1. van der Aalst, W.M.P.: Process Mining – Data Science in Action, Second Edition. Springer (2016)
2. Adamczyk, B., et al.: Dataset generation framework for evaluation of IoT Linux host-based intrusion detection systems. In: 2022 IEEE International Conference on Big Data (Big Data), pp. 6179–6187 (2022)
3. Ahmad, Z., Khan, A.S., Shiang, C.W., Abdullah, J., Ahmad, F.: Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Trans. Emerging Telecommun. Technol. **32**(1), e4150 (2021). https://doi.org/10.1002/ett.4150
4. Apruzzese, G., Andreolini, M., Marchetti, M., Venturi, A., Colajanni, M.: Deep reinforcement adversarial learning against botnet evasion attacks. IEEE Trans. Netw. Serv. Manage. **19**(1), 368–384 (2022). https://doi.org/10.1109/TNSM.2021.3091783
5. Bernardi, M.L., Cimitile, M., Distante, D., Martinelli, F., Mercaldo, F.: Dynamic malware detection and phylogeny analysis using process mining. Int. J. Inf. Secur. **18**, 257–284 (2019)
6. Bodyanskiy, Y., Boiko, O.: Online fuzzy clustering of data streams. In: Mashtalir, V., Ruban, I., Levashenko, V. (eds.) Advances in Spatio-Temporal Segmentation of Visual Data. SCI, vol. 876, pp. 211–241. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-35480-0_5
7. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, pp. 785-794. Association for Computing Machinery (2016). https://doi.org/10.1145/2939672.2939785
8. Clement, P., Desch, W.: An elementary proof of the triangle inequality for the wasserstein metric. Proc. Am. Math. Soc. **136**(1), 333–339 (2008). https://doi.org/10.1090/S0002-9939-07-09020-X
9. Czerwiński, M., et al.: Cybersecurity threat detection in the behavior of IoT devices: analysis of data mining competition results. In: FedCSIS 2023. ACSIS, vol. 35 (2023)
10. DeLong, E.R., DeLong, D.M., Clarke-Pearson, D.L.: Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics **44**(3), 837–845 (1988)
11. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behavior using deep learning. Decis. Support Syst. **100**, 129–140 (2017)
12. Folino, F., Guarascio, M., Pontieri, L.: Mining predictive process models out of low-level multidimensional logs. In: Jarke, M., et al. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 533–547. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_36
13. Gómez, J.R., et al.: An automatic complex event processing rules generation system for the recognition of real-time IoT attack patterns. Eng. Appl. Artif. Intell. **123**(Part B), 106344 (2023). https://doi.org/10.1016/J.ENGAPPAI.2023.106344

14. Hashemi, S.E., Gholian-Jouybari, F., Hajiaghaei-Keshteli, M.: A fuzzy c-means algorithm for optimizing data clustering. Expert Syst. Appl. **227**(C) (2023). https://doi.org/10.1016/j.eswa.2023.120377
15. Janusz, A., Kałuża, D., Chądzyńska-Krasowska, A., Konarski, B., Holland, J., Ślęzak, D.: IEEE BigData 2019 Cup: Suspicious Network Event Recognition. In: 2019 IEEE International Conference on Big Data (IEEE BigData), Los Angeles, CA, USA, 9-12 December 2019. pp. 5881–5887. IEEE (2019). https://doi.org/10.1109/BIGDATA47090.2019.9005668
16. De Koninck, P., vanden Broucke, S., De Weerdt, J.: act2vec, trace2vec, log2vec, and model2vec: representation learning for business processes. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 305–321. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_18
17. Macák, M., Daubner, L., Sani, M.F., Buhnova, B.: Cybersecurity analysis via process mining: A systematic literature review. In: ADMA. LNCS, vol. 13087, pp. 393–407. Springer (2021). https://doi.org/10.1007/978-3-030-95405-5_28
18. Mannhardt, F., Tax, N.: Unsupervised event abstraction using pattern abstraction and local process models. In: Joint Proceedings of the Radar tracks at the 18th International Working Conference on Business Process Modeling, Development and Support (BPMDS), and the 22nd International Working Conference on Evaluation and Modeling Methods for Systems Analysis and Development (EMMSAD), and the 8th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA) co-located with the 29th International Conference on Advanced Information Systems Engineering 2017 (CAiSE 2017), Essen, Germany, 12-13 June 2017. CEUR Workshop Proceedings, vol. 1859, pp. 55–63. CEUR-WS.org (2017). https://ceur-ws.org/Vol-1859/bpmds-06-paper.pdf
19. McInnes, L., Healy, J., Melville, J.: Umap: uniform manifold approximation and projection for dimension reduction (2020). https://arxiv.org/abs/1802.03426
20. Mikolov, T., et al.: Distributed representations of words and phrases and their compositionality. In: NIPS 2013, vol. 2. Curran Associates Inc., USA (2013)
21. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP, pp. 1532–1543 (2014)
22. Tay, J.K., Narasimhan, B., Hastie, T.: Elastic net regularization paths for all generalized linear models. J. Statist. Softw. **106**(1), 1–31 (2023). https://doi.org/10.18637/jss.v106.i01
23. Teinemaa, I., Dumas, M., Maggi, F.M., Di Francescomarino, C.: Predictive business process monitoring with structured and unstructured data. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 401–417. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_23
24. Vázquez-Barreiros, B., et al.: Prodigen: mining complete, precise and minimal structure process models with a genetic algorithm. Inf. Sci. **294**, 315–333 (2015)
25. Wang, C., Nulty, P., Lillis, D.: A comparative study on word embeddings in deep learning for text classification. In: NLPIR, pp. 37-46. Association for Computing Machinery (2021)
26. Wang, Y.A., Chen, Y.N.: What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 6840–6849. Association for Computational Linguistics (Nov 2020). https://doi.org/10.18653/v1/2020.emnlp-main.555
27. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)

28. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1/2), 31–60 (2001)
29. van Zelst, S., et al.: Event abstraction in process mining: literature review and taxonomy. Granular Comput. **6**, 719–736 (2021)
30. Zhong, Y., Goulermas, J.Y., Lisitsa, A.: Process mining algorithm for online intrusion detection system. In: Yavorskiy, R., Cavalli, A.R., Kalenkova, A. (eds.) Tools and Methods of Program Analysis, pp. 15–25. Springer Nature Switzerland, Cham (2024)

Automated Analysis of Pricings in SaaS-Based Information Systems

Alejandro García-Fernández(✉), José Antonio Parejo, Pablo Trinidad, and Antonio Ruiz-Cortés

SCORE Lab, I3US Institute, Universidad de Sevilla, Seville, Spain
{agarcia29,japarejo,ptrinidad,aruiz}@us.es

Abstract. Software as a Service (SaaS) pricing models, encompassing features, usage limits, plans, and add-ons, have grown exponentially in complexity, evolving from offering tens to thousands of configuration options. This rapid expansion poses significant challenges for the development and operation of SaaS-based Information Systems (IS), as manual management of such configurations becomes time-consuming, error-prone, and ultimately unsustainable. The emerging paradigm of Pricing-driven DevOps aims to address these issues by automating pricing management tasks, such as transforming human-oriented pricings into machine-oriented (iPricing) or finding the optimal subscription that matches the requirements of a certain user, ultimately reducing human intervention. This paper advances the field by proposing seven analysis operations that partially or fully support these pricing management tasks, thus serving as a foundation for defining new, more specialized operations. To achieve this, we mapped iPricings into Constraint Satisfaction Optimization Problems (CSOP), an approach successfully used in similar domains, enabling us to implement and apply these operations to uncover latent, yet non-trivial insights from complex pricing models. The proposed approach has been implemented in a reference framework using MiniZinc, and tested with over 150 pricing models, identifying errors in 35 pricings of the benchmark. Results demonstrate its effectiveness in identifying errors and its potential to streamline Pricing-driven DevOps.

Keywords: Automated Analysis · Software as a Service · iPricings

1 Introduction and Motivation

Software as a Service (SaaS) has rapidly gained popularity in recent years [10], driven by its flexibility to adapt to diverse user needs through subscription-based models. However, this adaptability has led to an exponential increase in configuration complexity, with modern pricing models evolving from offering a few options to thousands of potential configurations [6]. For example, Salesforce's November 2019 pricing offered up to 10 different configurations, while its July 2024 version allowed for over 12,000, as can be seen here. This rapid expansion presents significant challenges for the development and operation of

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 223–239, 2025.
https://doi.org/10.1007/978-3-031-94571-7_13

SaaS-based Information Systems (IS), as manual management of these configurations becomes a demanding task —being time-consuming and error-prone— especially when adapting to frequent pricing changes.

Addressing these challenges requires a shift towards automated, scalable solutions. The paradigm of *Pricing-driven Development and Operation* has emerged to tackle this issue, aiming to automate tasks related to the management of pricing models[1] in SaaS-based Information Systems [6]. These may include, for example, designing pricing models, converting human-oriented representations of pricings into machine-oriented (iPricings), or finding the optimal subscription that meets particular user requirements.

A foundational step toward this vision was presented in CAISE Forum, where a metamodel for representing pricings was proposed [7], and has laid the groundwork for the idea of machine-oriented pricings (iPricings) [3]. Building on this foundation, subsequent efforts have focused on leveraging iPricings to develop solutions that automate tasks within the scope of Pricing-driven DevOps. For instance, [8] proposed a reference architecture for automating the variability management in client-server architectures leveraging feature toggling [4], and AI4Pricing introduced initial advancements towards automating the transformation between human-oriented pricings and iPricings [3]. Although these contributions represent early steps, they highlight the progress being made toward realizing the vision of Pricing-driven DevOps.

However, a crucial pillar remains unexplored: the automated analysis (AA) of iPricings, i.e. the process of uncovering latent, yet non-trivial, information from machine-oriented pricings. For instance, this approach has the potential to enable real-time validation of pricing structures, strengthen data-driven pricing decision-making, or assist customers in identifying the most cost-effective subscription that meets their specific requirements. In light of this, the paper introduces the following original contributions, inspired by solutions previously applied in the automated analysis for feature models in CAISE 2005 [1] and service level agreements [13]:

1. An unprecedented formalization of iPricings as Constraint Satisfaction Optimization Problems (CSOPs), enabling their automated analysis through constraint programming techniques.
2. A set of seven fundamental analysis operations, built on the CSOP-based formalization, to support pricing structural validation, configuration space cardinality computation, and optimal subscription selection based on user requirements.
3. A tool to apply these operations, built with the MiniZinc solver [14] and integrated within a real platform (see Fig. 1).
4. Two curated datasets:
   - A novel synthetic dataset comprising 20 pricing models with seeded consistency errors. Although our solution successfully detected most of them, it missed 6 instances, making this dataset a valuable resource for future work aimed at improving automated inconsistency detection.

[1] For brevity, the term "pricing models" and "pricings" will be used interchangeably.

– An updated version of the benchmark introduced in [6], incorporating fixes for the 35 structural issues identified by our approach, providing a more reliable baseline for future research.

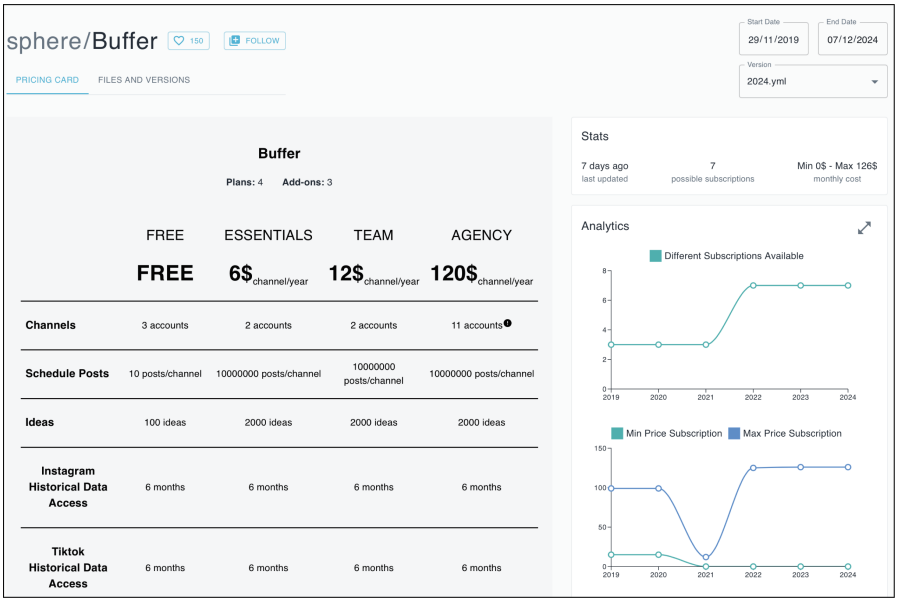


Fig. 1. A sample final product integrating the proposed set of analysis operations. Public access available here, along with other examples

The remainder of this paper is structured as follows: Sect. 2 frames the contribution within the existing literature and presents the core concepts, challenges, and research advancements in SaaS pricing. Section 3 presents an automated approach for iPricings analysis using CSOP-based operations. Next, Sect. 4 presents a reference implementation of the approach and tests it using benchmark and synthetic datasets. Finally, Sect. 5 showcases the conclusions.

2 Background & Related Work

2.1 User-Oriented SaaS Pricings

A pricing is a part of a SaaS customer agreement [5]. It structures the *features* of a service —defined as the distinctive characteristics whose presence/absence may guide a user's decision towards a particular subscription [7]— into *plans* and *add-ons* to control users' access to such features. While users can only subscribe to one of the available plans, they can subscribe to as many add-ons as they

want, as long as they are available for the contracted plan. As can be noted, in this domain, a feature encompasses both *functional features*, which constitute the core software product, and *extra-functional features*, which, while external to the product itself, enhance its perceived value (e.g., support, SLA guarantees, etc.). By including both types of features, pricings capture a more comprehensive view of what influences customer value, aligning the service offering with the overall customer agreement.

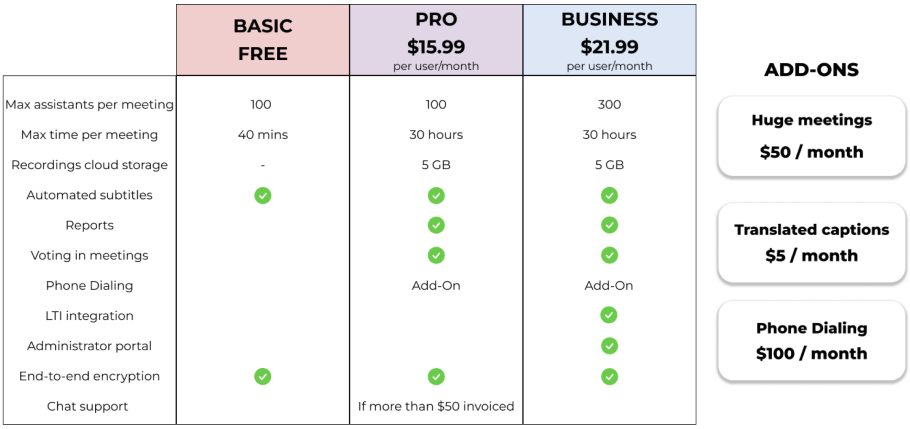


Fig. 2. Excerpt of Zoom's pricing with 13 features, three plans, and three add-ons, with a configuration space size of 20

In order to illustrate the upcoming concepts, we will use a real-world running example: Zoom. This is a cloud-based video conferencing service that enables users to virtually meet with others and optionally record the sessions to watch them later. An excerpt of its pricing, consisting of 13 features, is presented in Fig. 2.[2] In this example, nine features are managed through plans, one feature is associated with an add-on ("translated captions"), and one is governed by both ("phone dialing"). The pricing also enforces usage limits on the "meetings" feature (e.g., maximum assistants per meeting and maximum meeting duration) meaning that although the feature is available in all plans, the extent of their usage differs —higher-priced plans offer higher limits.

2.2 Machine-Oriented SaaS Pricings (iPricings)

The paradigm of *Pricing-driven Development and Operation* has emerged to address the challenges associated with pricing management tasks. Despite growing interest, there has been limited progress in this area. The Pricing4SaaS

[2] Pricing entries that impose or extend limits on meetings are considered usage limits rather than individual features-the overarching feature in this case is "meetings".

model, introduced as a formal metamodel, represents one of the first generalized attempts to standardize pricings [7]. This model defines pricings as a combination of *plans* and *add-ons* that regulate access to a set of features whose usage may be limited by usage limits. Additionally, its YAML-based serialization, Pricing2Yaml (see Fig. 3), has served as the initial step towards automated tooling for Pricing-driven DevOps [8] and, therefore, iPricings.

```yaml
saasName: Zoom
version: "2.0"
createdAt: 2024-11-04
currency: $
hasAnnualPayment: false
features:
  meetings:
    valueType: BOOLEAN
    defaultValue: true
    type: DOMAIN
  cloudRecordings:
    valueType: BOOLEAN
    defaultValue: false
    type: DOMAIN
  reports:
    valueType: BOOLEAN
    defaultValue: false
    type: INFORMATION
  ...
usageLimits:
  maxAssistantsPerMeeting:
    valueType: NUMERIC
    defaultValue: 2
    unit: use/month
    type: RENEWABLE
    linkedFeatures:
      - meetings
  ...
plans:
  BASIC:
    description: Basic plan
    price: 0.0
    unit: user/month
    features: null
    usageLimits: null
  PRO:
    description: Pro plan
    price: 15.99
    unit: user/month
    features:
      cloudRecordings:
        value: true
      reports:
        value: true
      votingInMeetings:
        value: true
      chatSupport:
        value: true
    usageLimits:
      maxTimePerMeeting:
        value: 1800
      recordingsCloudStorage:
        value: 5
  ...
addOns:
  hugeMeetings:
    availableFor:
      - BASIC
      - PRO
      - BUSINESS
    price: 50.00
    unit: /month
    usageLimits:
      maxAssistantsPerMeeting:
        value: 1000
  phoneDialing:
    availableFor:
      - PRO
      - BUSINESS
    price: 100.00
    unit: /month
    features:
      phoneDialing:
        value: true
  ...
```

Fig. 3. Zoom's pricing excerpt serialized in Pricing2Yaml

Building on this foundation, the study undertaken by [6] represents the first major effort to systematically analyze SaaS pricings. By modeling up to 162 pricings from 30 different SaaS providers -up to 6 different pricing versions between 2019 and 2024 per SaaS- this study not only demonstrated the flexibility of Pricing2Yaml (and, by extension, the adaptability of Pricing4SaaS) but also revealed significant trends in the evolution of pricings.

To perform their analysis and measure such evolution, the authors introduced the concept of *configuration space,* i.e. the set of different subscriptions within a pricing. As an illustration, the size of the configuration space of Zoom's pricing excerpt (see Fig. 2) is 20. Their results revealed an exponential growth of the configuration space, primarily due to the addition of add-ons over the last years, which diversifies available subscriptions and increases customization. Moreover, the study also highlights a widening gap between this configurability and the actual capabilities of feature toggling tools, which often struggle to manage the

increasing variability. This disparity indicates that existing technical support for handling such challenges remains limited, underscoring a need for automated tools that reduce complexity and time-to-market while allowing the software to adapt to evolving customer demands.

2.3 Automated Analysis in Feature Models and SLAs

Automated pricing analysis shares foundational elements with existing paradigms like Feature Models (FM) in Software Product Lines (SPLs) [2] and SLA analysis frameworks such as WS-Agreement [13]. Both FMs and pricings leverage features as their minimal unit of representation, functioning as variability modeling languages (VMLs) [1] that enable configurability and modularity. However, pricings extend beyond the usually technical scope of FMs by incorporating operational and market-driven elements critical to SaaS offerings, such as SLA guarantees, support, discounts, etc. Although most of those elements could be represented through attributed or stateful feature models [16], using DSLs and tools with a syntax and operations more specific to the pricings management problem emerges as a promising, more natural approach. Moreover, pricings must handle dynamic variables, such as usage-based limits, which can vary depending on add-ons (e.g., overage fees) or be customized by the user when subscribing, making their representation as a FM more complex.

Similarly, the automated analysis of iPricings shares conceptual ground with WS-Agreement, a framework for formalizing service agreements through an XML-based serialization, primarily focusing on SLAs [12]. While WS-Agreement supports the evaluation of user-defined constraints (e.g., agreement creation constraints), it lacks the vocabulary to represent pricing-specific elements like plans, add-ons, and usage-based fees. Extending WS-Agreement to encompass iPricings would require substantial semantic modifications, making it verbose and poorly suited to SaaS pricing strategies. Moreover, in contrast, iPricings approaches such as [6,7] embed subscription constraints directly within the pricing artifact (e.g., "select one plan and up to three add-ons" or "if add-on A2 is selected, exclude A7"), enabling a seamless and efficient evaluation of customer configurations without requiring external frameworks or additional layers of complexity.

The automated analysis of iPricings can thus be viewed as a specialized subset of SLA analysis, where the SLA models have variants corresponding with each possible configuration users can subscribe to. In this way, the demonstrated efficacy of CSOPs in the automated analysis of WS-Agreement documents for evaluating SLA compliance [11,12], combined with the conceptual parallels between WS-Agreement and iPricings, has been the driving force behind this work's adoption of CSOPs as its approach to the AA of iPricings. With this alignment, we ensure a structured methodology that leverages proven techniques to address the unique challenges posed by iPricings. Pricing4SaaS builds upon this foundation by categorizing SLA-related constraints, such as guarantees (e.g., TTO) and support, and embedding them within pricings as features and usage limits.

This duality positions iPricings as the intersection of two paradigms: the configurability and modularity inherent to feature models, and the guarantee-

driven focus of WS-Agreement. By combining these dimensions, iPricings not only address the operational intricacies and market-driven complexities of SaaS offerings but also establish themselves as a versatile variability modeling language capable of capturing both functional and service-level aspects of modern software systems.

3 Automated Analysis of iPricings

Given the growing complexity of pricings and their fast pace of change, automating the analysis of these structures is essential. Automation enables real-time validation of pricing specifications, ensuring their consistency and providing critical insights for tasks such as design, management of changes, and the choice of specific subscriptions based on additional user requirements.

In what follows, we present our approach for the automated analysis of Pricing2Yaml specifications (see Fig. 3) using constraint programming, given that, to the best of our knowledge, is the only serialization of pricings proposed in the literature. Along this section, we first present the formal semantics of Pricing2Yaml by explaining how these specifications can be mapped to a constraint satisfaction optimization problem (CSOP). Then, we present a catalog of seven analysis operations of Pricing2Yaml specifications and show how they can be automated using standard constraint programming reasoning operations.

3.1 Formal Semantics of iPricings

The primary objective of formalizing iPricings is to establish a sound basis for automating the analysis, decision-making, and task automation in the context of SaaS pricing models, thus enacting the vision of iPricings. Following the principles defined by [15], we follow a transformational style by translating Pricing2Yaml specifications to a target domain suitable for the automated analysis (*Primary Goal Principle*). Specifically, we propose Constraint Satisfaction Problems (CSPs) as this target domain, enabling pricings to be analyzed using constraint programming tools. This approach aligns with previous efforts to automate the analysis of feature models [2] and service level agreements [11,12].

A CSP is a 3-tuple (V, D, C), where V is a set of variables, D represents the set of domains for these variables, and C is a set of constraints among the variables. A solution to a CSP is an assignment of values from each domain in D to the variables in V that satisfies all constraints in C. Solving a CSP is finding one or more solutions that meet all the constraints. Constraint Satisfaction Optimization Problems (CSOPs) extend the CSP formulation by incorporating an objective function. Formally, a CSOP can be represented as (V, D, C, O), where (V, D, C) defines the underlying CSP, and O is an objective function that assigns a value (often to be minimized or maximized) to each possible solution. Thus, instead of merely satisfying the given constraints, a CSOP seeks a solution that also optimizes O.

Our proposal formulates pricings as CSOPs. We introduce a generic model π that captures the common structure and constraints of all pricing scenarios, as well as a mapping from Pricing2Yaml to a data file specifying the parameters of a given instance. This clear separation between the model —defining the structure of the pricing problem— and the data —specifying a particular instance— enables the creation of a single, reusable model for multiple datasets, simplifying the formulation, analysis, and optimization of pricings as CSOPs.

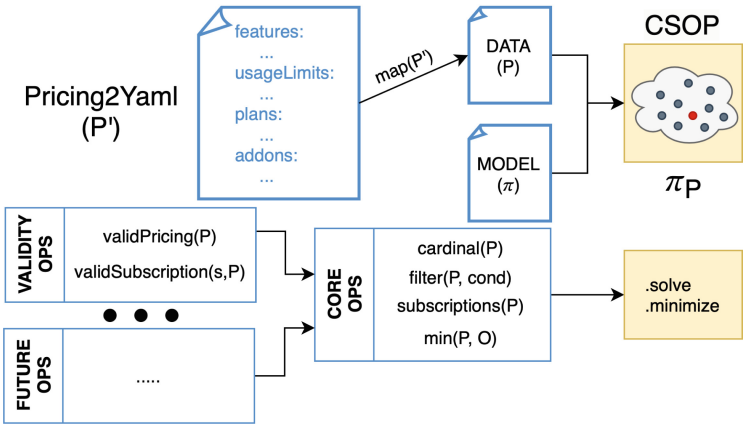


Fig. 4. Outline of our approach to automate the analysis of Pricing2Yaml serializations

In our approach (see Fig. 4), the model π represents a generic pricing scenario, with P_p plans and P_A add-ons. It encapsulates fundamental validity criteria for pricings that stem directly from the semantics of the pricing concepts (plans, add-ons, etc.). More specifically, π enforces the following constraints:

1. A pricing P cannot be empty, it must at least contain a plan or an add-on

$$notEmpty(P) \iff (P_p \neq \emptyset \vee P_A \neq \emptyset)$$

2. For each plan $p \in P_p$, if usage limits are defined, the features affected by those usage limits must be included within the same plan.

$$containsLinkedFeatures(p) \iff \begin{cases} p.usageLimits \neq \emptyset \implies \\ \forall u \in p.usageLimits, \\ u.linkedFeatures \subseteq p.features \end{cases}$$

3. If the pricing includes both plans and add-ons, each add-on $a \in P_A$ must be available for at least one plan.

$$isAvailable(a) \iff ((P_p \neq \emptyset \wedge P_A \neq \emptyset) \implies a.availableFor \neq \emptyset)$$

In a very similar way, the validity criteria for a subscription S comprises the following constraints:

1. A subscription cannot be empty; it must include either a plan p with zero or more add-ons (the set of selected add-ons is denoted as A_S) if the pricing includes plans, or at least one add-on if the pricing consists solely of add-ons.

$$notEmptySubscription(p, A_S) \iff (p \neq \emptyset \vee A_S \neq \emptyset)$$

2. If the pricing includes both plans and add-ons, and a subscription includes add-ons, every add-on must be available for the specific plan selected in that subscription.

$$addonsAvailable(p, A_S) \iff \begin{cases} p \neq \emptyset \wedge A_S \neq \emptyset \implies \\ \forall a \in A_S, \\ p \in a.availableFor \end{cases}$$

3. If a subscription includes add-ons with dependencies on other add-ons, all required dependencies must also be part of the subscription.

$$dependencyAware(A_S) \iff \begin{cases} A_S \neq \emptyset \implies \\ \forall a \in A_S, \forall a' \in a.dependsOn, \\ a' \in A_S \end{cases}$$

4. If a subscription includes add-ons that exclude others, all excluded add-ons cannot be included within the subscription.

$$exclusionAware(A_S) \iff \begin{cases} A_S \neq \emptyset \implies \\ \forall a \in A_S, \forall a' \in a.excludes, \\ a' \notin A_S \end{cases}$$

Then, we denote every model instance as π_P, created by pairing the model π with the particular data of a pricing P. Solutions to π_P —i.e. $solve(\pi_P)$— thus represent the set of feasible configurations, for a pricing P, where all defined constraints are satisfied.

3.2 Analysis Operations

Building on the formalization outlined in the previous section, where iPricings were mapped to CSOPs, we now turn our attention to the analytical operations that this formalization enables. These operations are key to extracting meaningful, latent insights from iPricing specifications. To provide clarity and continuity, each operation is illustrated using the Zoom's pricing excerpt (see Fig. 2) as a running example.

Number of Configurations. One of the most important metrics derived from a pricing is the cardinality of its configuration space, representing the number of different configurations that can be generated from it. A higher cardinality indicates increased flexibility for the SaaS, but also adds complexity to its maintenance.

Definition 1 (Cardinality). *Let P be a pricing, the number of configurations of P, hereinafter cardinal, is equal to the number of solutions of its equivalent CSOP π_P.*

$$cardinal(P) = |solve(\pi_P)|$$

In the example of Zoom's pricing excerpt, $cardinal(Zoom) = 20$. However, simply by adding a new plan, the cardinality increases to 28. In contrast, introducing an additional add-on doubles the cardinality, resulting in $cardinal(Zoom) = 40$. As demonstrated in [6], each new add-on leads to an exponential increase in cardinality, whereas adding a new plan results in only a linear increase.

Filter. This is particularly useful for customers that are looking for a configuration with specific features and usage limits, i.e. they are not interested in all possible configurations but those that tailor to their needs —i.e. pass the filter.

Definition 2 (Filter). *Let P be a pricing and F a constraint representing a filter, containing the desired features and usage limits' values within the subscription. The filtered pricing of π_P, hereinafter filter, is equal to π_P with the added constraint F.*

$$filter(P, F) \iff (\pi_P \wedge F)$$

A possible filter for zoom's pricing excerpt would be to ask for all configurations with the "administrator portal" feature and, at least, a 200 assistants capacity in meetings. By applying this filter to the pricing, the number of potential configurations decreases from 20 to 8.

$$F = (administratorPortal = true \wedge maxAssistantsPerMeeting >= 200)$$

$$cardinal(filter(Zoom, F)) = 8$$

Subscriptions. Once π_P is defined, there should be a way to get all the solutions of the CSOP, i.e. possible configurations of the pricing.

Definition 3 (Subscriptions). *Let P be a pricing, the potential configurations of the pricing, hereinafter subscriptions, are equal to the solutions of the equivalent CSOP π_P.*

$$subscriptions(P) = \{s \in solve(\pi_P)\}$$

Within zoom, we would like to get all possible subscriptions that include the "records" feature. In this way, $P = filter(Zoom, records = true)$ and $subscriptions(P) = \{s \in solve(\pi_P \wedge records = true)\}$.

Subscription Cost. In order to perform cost optimization operations, it is necessary to define how the cost of a subscription is computed.

Definition 4 (Subscription Cost). *Let s be a subscription of the pricing P. The cost associated with s is calculated as the sum of the selected plan's price, if it exists, and the prices of the selected add-ons, if any.*

$$cost(s) = s.plan.price + \sum_{a \in s.addons} a.price$$

Within the zoom's pricing excerpt, we would like to get the cost of a subscription that includes the plan "PRO", and the add-on "Huge Meetings", i.e., $s = (PRO, \{HugeMeetings\})$. Thus, $cost(s) = 65.99$.

Pricing Validation. A pricing is valid when it has at least one configuration that can be selected. That is, a model where π_P has at least one solution.

Definition 5 (Valid Pricing). *A pricing P is valid if its equivalent CSOP π_P is satisfiable.*

$$valid(P) \iff subscriptions(P) \neq \emptyset$$

As an illustration, let $CircularConstraints$ be a pricing that consists solely of three add-ons $P_A = \{a_1, a_2, a_3\}$, that share the following restrictions among them: i) a_1 depends on a_2, ii) a_2 depends on a_3, and iii) a_3 excludes a_1. Then:

$$valid(CircularConstraints) = false$$

Subscription Validation. Since many different subscriptions can be generated from a pricing, it should be possible to contrast whether a subscription is valid according to a pricing.

Definition 6 (Valid Subscription). *Let P be a pricing and s be a subscription, that is considered valid if and only if it is a potential solution of π_P.*

$$valid(s, P) \iff s \in subscriptions(P)$$

Taking the excerpt of zoom's pricing as an illustration, a subscription that includes the usage limit "maxAssistantsPerMeeting = 1200" is not valid, as the maximum permitted value for this usage limit is 1000, achievable only by purchasing the "Huge Meetings" add-on.

Optimum Subscription. Identifying the optimum subscription based on a specific criterion is essential in pricing for both customers and providers. Customers can determine the minimum price for a defined set of features and usage limits, while providers gain insight into the price range at which a particular set of features and usage limits is offered.

Definition 7 (Optimum). *Let P be pricing and O an objective function for the subscription cost, then the optimum set of subscriptions, hereinafter max and min, is equal to the optimum space of π_P.*

$$max(P, O) = max(\pi_P, O)$$

$$min(P, O) = min(\pi_P, O)$$

It is also possible to apply a filter to the zoom's pricing excerpt and then ask for an optimal subscription. Thus, a possible optimum criterion for our running example would be to ask for all subscriptions with $record = true \wedge cloudStorage >= 5$, and the minimum value for the subscription cost. In this case, optimum subscriptions S_{opt} are:

$$S_{opt} = min(filter(Zoom, record = true \wedge cloudStorage >= 5), cost)$$

4 Tooling Support and Testing

This section presents the implementation of the proposed CSOP-based approach for automating iPricing analysis using MiniZinc [14], as well as its testing with real-world and synthetic datasets. Due to space limitations, the specific mapping of each individual constraint from Sect. 3.2 to this solver, as well as the definition of the involved variables and domains, are defined in the technical report of this paper, that is available as part of the laboratory package [9].

At the core of this implementation is MiniZinc, a medium-level declarative modeling language recognized for its flexibility and compatibility with a wide range of constraint programming (CP) solvers. By separating CSPs into two components —the model, which defines the constraints and structure of a class of problems— and the data —which specifies the parameters for a particular instance— MiniZinc allows a single model to be reused across datasets, thus enhancing scalability and modularity. Moreover, its ability to translate CSPs into the FlatZinc format ensures solver compatibility [14], bridging the gap between high-level problem descriptions and low-level solver implementations.

Leveraging MiniZinc's modularity, we developed a hierarchical formalization for iPricings, structured using a bottom-up methodology. We begin with a foundational model that establishes the core parameters and variables of the CSOP (PricingModel.mzn). The parameters, drawn from input data, encapsulate the details of a specific pricing, while the variables define the subscription attributes

that constitute the solutions of the CSOP. This foundation enables us to incrementally add layers of constraints that are tailored to specific analytical tasks, making it easier to extend the set of AA operations the approach can support.

Building on this foundation, we introduce models that ensure the integrity of the input pricing (valid-pricing.mzn) and the correctness of derived subscriptions (valid-subscription.mzn). On top of these, we built specialized analysis models to perform the operations described in Sect. 3. Each layer refines and extends the constraints set by its predecessor, ultimately producing a fully tailored CSOP for a specific task. To instantiate an analysis, the chosen model is paired with a corresponding *.dzn* data file —automatically generated from a Pricing2Yaml serialization— ensuring a clear separation between the generic problem structure and the pricing-specific details.

To validate the proposed framework, two datasets were utilized. The first one, derived from the benchmark presented in [6], encompasses 162 real-world SaaS pricings spanning 30 different services over a five-year period (see Table 1). The initial set of services was taken from the repository presented in [7], and subsequently expanded based on three main criteria: (i) snapshot availability via the Wayback Machine, (ii) clear feature listings within each snapshot's pricing page, and (iii) appropriate temporal spacing between snapshots (6 to 12 months). Although the resulting dataset captures services of various sizes and key real-world aspects such as frequent pricing updates and usage-based offerings, the exclusive use of Pricing2Yaml serialization causes inherent limitations in representing more complex scenarios, such as bundled offerings or discounts.

Analysis of this dataset revealed inconsistencies in 35 out of 162 pricings (21.6% of the cases): i) 25 linked feature mismatches, where non-zero usage limits were assigned to unavailable features; ii) eight incorrect feature definitions, where features were incorrectly represented as numerical values instead of usage limits; iii) one temporal error, with a pricing having a future creation date; and iv) one pricing without specific prices (Trustmary 2020), where all plans required contacting sales. Notably, 34 out of 35 inconsistencies stemmed from errors during the manual transition from web pricings to iPricings, underscoring the risk of human error in this process. To address this, we corrected all 34 inconsistencies originating from human errors, ensuring the dataset aligns with the intended specifications.[3] Additionally, to mitigate similar issues in the future, we have developed a Pricing2Yaml editor within SPHERE, a platform for intelligent pricing-driven solutions.[4] The editor provides real-time error detection during the modeling process (using the validity constraints defined in this paper) and offers a live rendering of the pricing as it is constructed/edited, streamlining the development of reliable iPricings.

The second dataset contains synthetic pricings intentionally designed to include inconsistencies. Although most cases adhere to the syntax rules of Pricing2Yaml, they exhibit errors that rendered them invalid within the iPricings

[3] The corrected version of the real-world pricings dataset is included both in the laboratory package [9] and in SPHERE.

[4] https://sphere.score.us.es/editor.

Table 1. Benchmark overview from [6]. Each stat indicates the number of: snapshots (S), features (F), plans (P), add-ons (A), configuration space size (C); in 2024

SaaS	S	F	P	A	C	SaaS	S	F	P	A	C
Salesforce	6	111	3	14	12544	Buffer	6	76	4	3	7
GitHub	6	81	3	14	8960	Jira	6	60	4	1	7
Postman	5	100	4	12	1412	Notion	4	58	4	1	7
Databox	6	62	5	8	786	Figma	6	90	6	0	6
OpenPhone	5	48	3	6	192	Box	6	50	5	0	5
Wrike	6	78	5	5	85	Canva	6	92	4	0	4
Tableau	6	41	3	7	48	Dropbox	4	82	4	0	4
Zapier	5	51	4	4	40	Evernote	6	32	4	0	4
Slack	4	44	4	4	21	Hypercontext	4	63	4	0	4
MailChimp	6	90	4	5	15	Pumble	4	34	4	0	4
ClickUp	6	135	4	2	13	UserGuiding	5	59	3	1	4
Planable	6	41	4	2	13	Crowdcast	5	16	3	0	3
Clockify	6	72	6	4	10	Deskera	4	100	3	0	3
Microsoft 365	6	60	4	1	8	Overleaf	6	16	3	0	3
Trustmary	5	45	4	1	8	Quip	6	15	3	0	3

domain. Thus, inconsistencies range from syntax issues (e.g., add-ons linked to nonexistent plans) to structural flaws, where pricings failed to behave as expected. Structural flaws include cases where the expected configuration space is not generated or "dead" pricing elements are present —e.g., plans with identical features and usage limits but different prices. By applying the "valid subscription" operation across all cases, our framework successfully identified the majority of inconsistencies, while cases involving unaddressed analysis operations highlight the need for extending the approach to handle more complex scenarios like dead elements,[5] paving the way for future advancements in pricing analysis.

As a final test, we applied our approach to replicate the analytical operations described in [6] using their proposed benchmark. The results obtained with our method matched the manually extracted values reported in [6], demonstrating the accuracy and applicability of the proposed catalog of operations.

In summary, the proposed framework demonstrates its robustness and versatility in automating iPricing analysis, addressing both real-world scenarios and deliberately challenging synthetic cases. By leveraging the modularity and scalability of MiniZinc, coupled with the support of tools like the Pricing2Yaml editor, the approach not only helps to reduce human errors but also provides an efficient pathway for analyzing pricings. The results highlight the potential of constraint programming to drive innovation in this domain, setting the foundation for future enhancements and broader applications in dynamic pricing ecosystems.

[5] For further details, we invite the reader to consult the technical report [9].

5 Conclusions and Future Work

In this work, we first formalized iPricings as Constraint Satisfaction Optimization Problems (CSOPs), establishing a structured foundation for their analysis. Building on this, we defined a set of analysis operations to partially automate and streamline key pricing management tasks. For instance, evaluating the cardinality of the configuration space can aid decision-making during pricing design, while optimizing subscriptions based on cost or feature requirements can assist users in selecting the most suitable subscription.

As part of our testing efforts, we i) identified and corrected inconsistencies in a real-world SaaS pricing dataset [6], enhancing its reliability for future studies; and ii) developed a synthetic dataset to benchmark the detection of pricing inconsistencies, offering a robust foundation for evaluating and improving future contributions in this domain; iii) introduced a Pricing2Yaml editor within SPHERE, designed to improve the transition from web pricings to iPricings by reducing human error through real-time error detection and live rendering of pricings during their construction; and iv) also integrated within this platform a solution based on our approach: pricing cards[6].

Looking ahead, this work opens several promising avenues for future research in automated analysis of SaaS pricings. Key opportunities include:

1. Improving the explainability of the framework's insights for developers and pricing decision-makers is critical. Given MiniZinc's inherent limitations in providing explanations, exploring alternative solvers with built-in diagnostic capabilities could significantly enhance usability. However, such alternatives would require additional implementation effort to replicate the modularity that MiniZinc currently offers natively.
2. Since our proposed framework is inherently tied to the Pricing2Yaml serialization—and thus, indirectly, to the Pricing4SaaS metamodel—it naturally inherits the limitations identified in prior research [6], such as the representation of bundles and discount mechanisms. A promising future direction is, therefore, to first address and extend these identified limitations in the Pricing4SaaS metamodel itself, before incorporating the corresponding improvements into our framework. This would allow the framework to accurately capture more intricate and realistic pricing strategies encountered in real-world SaaS offerings.
3. The framework currently assumes that users express their requirements using the provider's terminology, which is rarely the case. Developing an automated mapping process capable of translating user-defined requirements into the provider's pricing terminology would significantly enhance the intuitiveness and practical applicability of the optimization operation of the framework.
4. Manual translation from user-oriented pricings to iPricings introduces significant error risk, as highlighted by our results (see Sect. 4). Therefore, further automation in this process, possibly leveraging large language models (LLMs), shows promise for significantly reducing human errors.

[6] Sample pricing card can be found here.

By addressing these challenges, we hope that future research can push the boundaries of Pricing-driven DevOps, making automated pricing analysis not only scalable and efficient but also intuitive, adaptable, and truly aligned with the evolving needs of SaaS ecosystems.

Acknowledgments. This work has been partially supported by grants PID2021-126227NB-C21 and PID2021-126227NB-C22 funded by MCIN / AEI / 10.13039 / 501100011033/FEDER, UE, and grants TED2021-131023B-C21 and TED2021-131023B-C22 funded by MCIN/AEI/10.13039/501100011033 and by European Union "NextGenerationEU"/PRTR.

References

1. Benavides, D., Trinidad, P., Ruiz-Cortés, A.: Automated reasoning on feature models. In: Pastor, O., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 491–503. Springer, Heidelberg (2005). https://doi.org/10.1007/11431855_34
2. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: a literature review. Inf. Syst. **35**(6), 615–636 (2010)
3. Cavero, F.J., Alonso, J.C., Ruiz-Cortés, A.: From Static to Intelligent: Evolving SaaS Pricing with LLMs. In: Proceeding of the International Conference on Service Oriented Computing, (ICSOC). 1st International Workshop on Service-Oriented Computing for Artificial Intelligence, (SOC4AI), Springer LNCS (2024)
4. Fowler, M.: Feature toggles (aka feature flags) (nd). https://martinfowler.com/articles/feature-toggles.html, Accessed December 2023
5. García, J.M., Martín-Díaz, O., Fernandez, P., Müller, C., Ruiz-Cortés, A.: A flexible billing life cycle for cloud services using augmented customer agreements. IEEE Access **9**, 44374–44389 (2021)
6. García-Fernández, A., Parejo, J.A., Cavero, F.J., Ruiz-Cortés, A.: Racing the market: an industry support analysis for pricing-driven devops in saas. In: Service-Oriented Computing, pp. 260–275 (2024)
7. García-Fernández, A., Parejo, J.A., Ruiz-Cortés, A.: Pricing4SaaS: towards a pricing model to drive the operation of SaaS. In: Intelligent Information Systems, - CAiSE Forum, Proceedings. LNBIP, vol. 520, pp. 47–54. Springer (2024). https://doi.org/10.1007/978-3-031-61000-4_6
8. García-Fernández, A., Parejo, J.A., Trinidad, P., Ruiz-Cortés, A.: wards Pricing4SaaS: a framework for pricing-driven feature toggling in SaaS. In: Stefanidis, K., Systä, K., Matera, M., Heil, S., Kondylakis, H., Quintarelli, E. (eds.) Web Engineering. ICWE 2024. LNCS, vol. 14629. Springer, Cham. https://doi.org/10.1007/978-3-031-62362-2_30
9. García-Fernández, A., Parejo, J.A., Trinidad, P., Ruiz-Cortés, A.: Automated Analysis of Pricings in SaaS-based Information Systems - Supplementary Material (2025). https://doi.org/10.5281/zenodo.14254341
10. Jiang, Z., Sun, W., Tang, K., Snowdon, J., Zhang, X.: A pattern-based design approach for subscription management of software as a service. In: 2009 Congress on Services - I, pp. 678–685 (2009)
11. Müller, C., Gutierrez Fernandez, A.M., Fernandez, P., Martin-Diaz, O., Resinas, M., Ruiz-Cortes, A.: Automated validation of compensable SLAs. IEEE Trans. Serv. Comput. (2018)

12. Müller, C., Resinas, M., Ruiz-Cortés, A.: Automated analysis of conflicts in WS-agreement. IEEE Trans. Serv. Comput. **7**(4), 530–544 (2014)
13. Müller Cejás, C., Resinas Arias de Reyna, M., Ruiz Cortés, A.: Automated analysis of conflicts in ws-agreement documents. IEEE Trans. Services Comput. **7**(4), 530-544 (2013)
14. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: towards a standard CP modelling language. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 529–543. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74970-7_38
15. Ter Hofstede, A., Proper, H.A.: How to formalize it? formalization principles for information system development methods. Inf. Softw. Technol. **40**, 519–540 (1998)
16. Trinidad, P., Ruiz-Cortés, A., Benavides, D.: Automated analysis of stateful feature models. In: Seminal Contributions to Information Systems Engineering: 25 Years of CAiSE, pp. 375–380 (2013). https://doi.org/10.1007/978-3-642-36926-1_30

Extending Process Modelling

Modeling and Monitoring Business Constraints of Non-conformant Choreographed Business Processes

Giovanni Meroni[1], Pierluigi Plebani[2(✉)], Simone Tagliente[2], and Marco Montali[3]

[1] Technical University of Denmark, Kgs. Lyngby, Denmark
giom@dtu.dk
[2] Politecnico di Milano, Milan, Italy
pierluigi.plebani@polimi.it, simone.tagliente@mail.polimi.it
[3] University of Bozen-Bolzano, Bolzano, Italy
montali@inf.unibz.it

Abstract. A choreographed process defines the agreement between organizations with respect to the protocol that should be respected when the involved organizations interact with each other through their business processes. The resulting BPMN Choreographed Process that models this protocol can formalize business constraints.

Although there are solutions that can monitor whether these constraints are satisfied, situations of non-conformance that have an impact on monitoring can arise. This happens especially when the involved parties are not equipped with a BPMS that can force business processes to align with the protocol.

The purpose of this paper is twofold. First of all, it proposes the use of an extension of timed commitments to express business constraints. This extension takes into account situations of non-conformance that are not acceptable by the protocol, but are compatible with business objectives. In addition, it proposes a monitoring application solution that can support the proposed extension.

Keywords: Commitments · Process Monitoring · Multi-party business processes

1 Introduction

The increasing possibilities of interconnection between heterogeneous systems, not only belonging to a single organization, but also to separate organizations, has allowed an ever greater interconnection even between business processes. From a modeling point of view, the definition of choreographed processes has made it possible to define the protocols that govern the interaction between these systems. Among the different approaches proposed, today it is possible to affirm that BPMN has made it possible to define these protocols with a

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 243–258, 2025.
https://doi.org/10.1007/978-3-031-94571-7_14

notation widely shared by academics and practitioners. This can be done with choreography models that emphasize the interaction among the parties, or with collaborative process models where information about how internally the actors manage the interaction are also specified.

Although BPMN allows the modeling of functional aspects of this interaction protocol, the same cannot be said for non-functional aspects related to business constraints. Often, constraints such as the lead time, the status of the resources used or exchanged, and the execution time of one or more specific tasks are indicated with accompanying documents without a precise and shared formalism.

In this area, the use of commitments has gained considerable interest over the years, starting from seminal work in the agent-based systems [13], which has then been applied to business process management [6,12]. With a specific emphasis on the definition of constraints that exist for choreographed processes, commitments allow you to define constraints specifying who is the debtor and who is the creditor, as well as the nature of the constraint. In this context, Timed Commitments [2,11] are an extension of the classic commitments, where the notion of time is introduced to be able to also consider the specific nature of a process.

Nevertheless, timed commitment might struggle to capture and adapt to situations where the monitored process does not necessarily adhere to the initially choreographed model. Thus, possible non-conformance can affect the ability of the timed commitment to effectively monitor the status of the process as it reduces the ability of the timed commitment to really understand which is the actual flow of operations.

The goal of this work is twofold. On the one hand, we propose a commitment notation, compatible with the BPMN Choreography meta-model, that can define business constraints also considering possible non-conformance in the choreography protocol. On the other hand, we provide a software solution capable of monitoring a choreographed process for the verification of the satisfaction, or violation, of the commitments indicated at the model level.

The rest of the paper is organized as follows. Section 2 provides an overview of the literature related to the objective of this work. Section 3 summarizes the main aspects of time commitments that are here adopted. After introducing a running example in Sect. 4, Sect. 5 provides details of the notation introduced to express non functional constraints. Based on this notation, Sect. 6 describes the monitoring approach able to keep working also in case of non-conformance, and Sect. 7 provides some information about the software application implementing the proposed approach. Finally, Sect. 8 concludes the work outlining possible future directions.

2 Related Work

Compliance checking is related to semantic correctness, i.e., to comply with imposed rules stemming from regulations, standards, and laws [8,10]. Most of the approaches concerning compliance are focused on the orchestrated process [7],

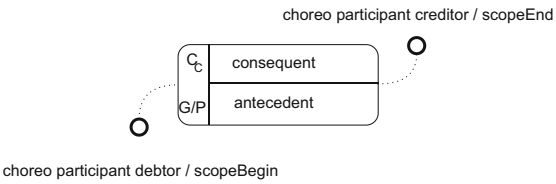


Fig. 1. Timed commitment element

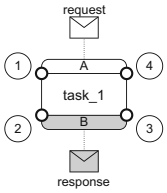


Fig. 2. Connection points

while in this paper the focus is shifted toward the choreography protocol. Yet, in this case, the study on methods to verify the compliance of choreographed protocols have been usually investigated considering the structure of the process, i.e., the control flow [5], while less attention is paid to the business constraints that predicates on elements that are related to the process model but where the process model alone is enough to define them. Examples of these business constraints are related to the time constraints [3] or security constraints [9].

About the modeling of business requirements, a survey of the proposed extensions to address this issue using BPMN is introduced in [14]. Among others, the approach in [4] supports designers in modeling temporal requirements directly within BPMN workflows. Yet, [1] introduces a framework for integrating non-functional requirements (NFRs) into business process models, adapting methodologies typically seen in software engineering for use within business process improvement contexts. As a peculiar aspect of our work, we rely on a formal element, i.e., commitment [12,13] initially introduced in the agent-based systems literature, to express these requirements and, in particular, on a variant proposed in [11] to also capture time dimensions.

3 Timed Commitments

The characteristics of a timed commitment are described here using the notation proposed in [2] that extends the classic notation of a commitment to also be able to define the timeframe within which the commitment should hold.

A timed commitment represents an obligation between a *debtor* (who is responsible for ensuring the fulfillment of the commitment) and a *creditor* (who has an interest in the satisfaction of the commitment). Maintaining its original graphic representation as proposed in [12], a commitment[1] is represented as a rectangle with rounded borders (see Fig. 1) labeled with the name of the commitment (e.g., C_c). Within the commitment are specified two boolean expressions that when evaluated true, indicate the *antecedent*, i.e., the condition to generate the commitment, and the *consequent*, the condition that must apply to make the commitment satisfied.

[1] For the sake of conciseness, from now on, we use the generic term "commitment" to indicate a timed commitment.

Compared to the original notation, the notation here adopted provides for: (i) the presence of two connectors, one identifying the debtor and the other the creditor, that can be attached to a BPMN choreography task to also indicate the *scopeBegin* and *scopeEnd* of the commitment; (ii) the distinction between *goal* commitment or *persistent* commitment.

Regarding the connector (see Fig. 2), if the debtor (or creditor) side of the connector is attached to ① or ④, it indicates that participant A holds the role of a debtor (or creditor) of the commitment. On the contrary, if the debtor (or creditor) side of the connector is attached to ② or ③ position, then participant B will hold the debtor (or creditor) role. In addition to defining the role of involved participants, the connector is also the tool with which it is possible to define the time scope of the commitment concerning the structure of the choreographed process. In fact, a connector associated with the debtor (or creditor) side to the position ① or ③ provides for the beginning (or the end) of the scope associated with when the message is sent by the participant A (in case of ①) or B (in case of ③). Conversely, a connector associated with the debtor (or credit) side to the ② or ④ position provides for the beginning (or end) of the scope when participant A (in case of ④) or B (in case of ②) receives the message, respectively. In this way, the activation state of a commitment, previously linked only to the antecedent and consequent conditions, is now included within the *scopeBegin* and *scopeEnd*.

The introduction of the explicit *scopeBegin* and *scopeEnd* allow us to extend the expressiveness of the commitments. In fact, in the original version, the antecedent corresponds to a condition resulting in the generation of the commitment, while the consequent states whether the commitment is satisfied or not and upholds the dismission of the commitment. Conversely, a timed commitment separates the lifetime of the commitment (corresponding to the interval between *scopeBegin* to *scopeEnd*) from the time frame in which the commitment should be verified, and this time frame depends on the type of commitment which can be: a *goal* commitment (letter G is placed in the lower left position) or a *persistence* commitment (with letter P). In the former case, assuming that the antecedent has become true at any moment after the scopeBegin[2], the consequent is evaluated when the *scopeEnd* is reached. This behavior corresponds to the case covered by the original commitment. In the latter case, the consequent is constantly evaluated from when the antecedent is true to the *scopeEnd*.

4 Running Example

For a more comprehensive understanding of the utilization of timed commitments, the BPMN Choreography in Fig. 3 is used. This is related to a fish shipping process performed by the Sea co., in collaboration with local and, if required, long-distance couriers. Aligned with the definition provided in the previous section, the process diagram is enriched with three commitments:

[2] It is worth noticing that it is not mandatory to specify the antecedent. If so, the commitment considers the antecedent as evaluated as true.

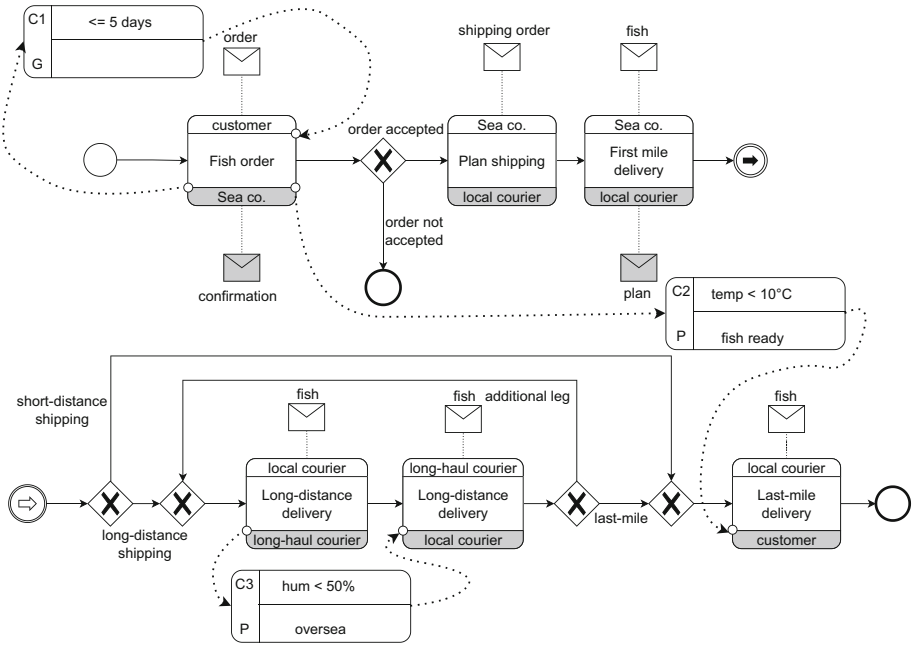


Fig. 3. Example of timed commitment in a BPMN Choreography diagram

- C_1: this commitment involves a single task, the scopeBegin is when the order is received by the Sea co, and the scopeEnd when the confirmation is received by the customer. The debtor is Sea Co. and the creditor is the customer. This commitment maps the requirement of the customer to receive a confirmation from the company in less than five days from when the order is sent. The commitment is a goal commitment, thus the consequent is evaluated only at the scopeEnd, i.e., the confirmation is sent. Finally, since any antecedent is not defined, the commitment is considered active from the time in which the order is received by the Sea co.
- C_2: this commitment extends to many tasks where the scopeBegin corresponds to the confirmation sent by Sea co. to the customer, while the scopeEnd is when the customer receives the fish. Again, the debtor is Sea Co. and the creditor is the customer. This commitment checks whether, during the shipping, the temperature of the fish is maintained below 10°C. This commitment is a persistent commitment, thus the consequent must be valid from when the fish is ready (i.e., the antecedent) to when it is delivered to the customer.
- C_3: this commitment ensures that for long-distance deliveries, the goods are kept below a certain humidity threshold only when the travel implies an oversea leg. Here, the debtor is the long-haul courier, and the creditor is the local courier. Also in this case, due to the nature of the constraint, the

comittment is a persiting one as the humidity threshold must be respects along the whole scope.

Although the participants in the choreography have shared this collaboration protocol and do everything to respect it, deviations from the expected behavior may happen. For example, the customer, after sending the order, could send an integration to that order. This often leads to discussions with Sea co. to determine if the limit of 5 days for sending the confirmation must start from the initial submission or the integration. In this paper, the objective is to provide a modeling tool capable of capturing these non-conformances and monitoring the execution of the protocol in accordance with these decisions in order to verify compliance with the non-functional constraints expressed by the commitment.

5 Non-conformant Choreographed Processes with Timed Commitments

5.1 Types of Non-conformance

Given a choreographed process, possible non-conformance can grouped into two main categories: (i) deviations in the execution flow of the activities carried out, and (ii) deviations in the number of instances of the messages exchanged.

The first case includes situations in which the interactions between the participants do not respect the order dependencies indicated in the model. For example, sending the shipping order by Sea co to the local courier only after Sea co has sent the fish. This corresponds to a switch between the tasks *Plan shipping* and *first-mile delivery*. This type of non-conformance often has a significant impact on the feasibility of the whole process and could lead to deadlocks. In this paper, this type of non-conformance is not taken into account, but it will be the subject of analysis in future developments.

A second category of non-conformance, which is the one considered in this work, concerns the cardinality of the messages exchanged: given a message attached to a choreography task, although a regular execution implies that this message is sent only once, more than one instances of this message may be sent. Although this behavior violates the choreography protocol, there are cases in which it can be tolerated. At the same time, the possible re-sending of the message could, however, violate or, worse, make the interpretation of the commitments ambiguous.

5.2 Timed Commitment Notation Extension

To model the accepted deviations in the number of instances of the messages exchanged, we extend the semantics of the connection points that characterize a timed commitment.

In particular, the connection point that defines the *scopeBegin* can be further characterized by an icon that defines the behavior to be followed in case there is a non-conformance due to a greater number of messages than what is defined

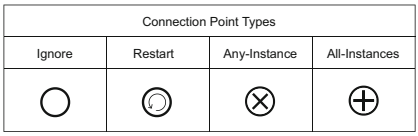


Fig. 4. Connection Point Types classification

in the choreography task. As shown in Fig. 4, we have four situations that have been identified based on our experience and the analysis of common situations in real business processes:

- *Ignore*: the commitment, generated when the first message associated with the *scopeBegin* arrived, expects only one message. Any other messages are simply ignored.
- *Restart*: although the message associated with the *scopeBegin* has already arrived, in case of a new message of the same type, the running commitment is canceled and a new one is created according to the new message.
- *any-instance*: any instance of the message associated to the *scopeBegin* generates a new commitment instance. Concerning the previous cases, having the possibility to have more than one commitment instance, it is fundamental to define how to evaluate the respect of the non-functional requirement associated with the commitment. In the case of *any-instance*, we assume that the receiving of the first message associated to the *scopeEnd* triggers the cancellation of all the running instances. The commitment is satisfied if the consequent for the regularly closed commitment is evaluated true.
- *all-instances*: as per the *any-instance* a new commitment is generated for each upcoming message related to the *scopeBegin*. Conversely, the commitment is evaluated only when all the *scopeEnd* messages associated to all the running instances are received. Here, the commitment is considered satisfied if the consequents of all the commitment instances are evaluated as true.

Referring to the previously introduced running example, Fig. 5 shows the use of the connection points. In particular, the commitment C_1 is defined as *restart*. In this way, it becomes clear that a possible integration of the order cancels the existing commitment and generates a new commitment. So the times are recalculated starting from the sending of the new message. This solves the ambiguity previously discussed in Sect. 4.

Otherwise, the commitment C_2, which is of type *ignore*, is activated by sending the first confirmation. This is to ensure the maintenance of the cold chain starting from the fish prepared upon receipt of the first order. A possible second order by the customer requires a new confirmation but this confirmation does not cancel the commitment, and therefore not even the temperature monitoring started previously.

Finally, considering C_3, if the local courier delivers the fish in batches to the long-haul courier, each of them generates a new commitment (and therefore a

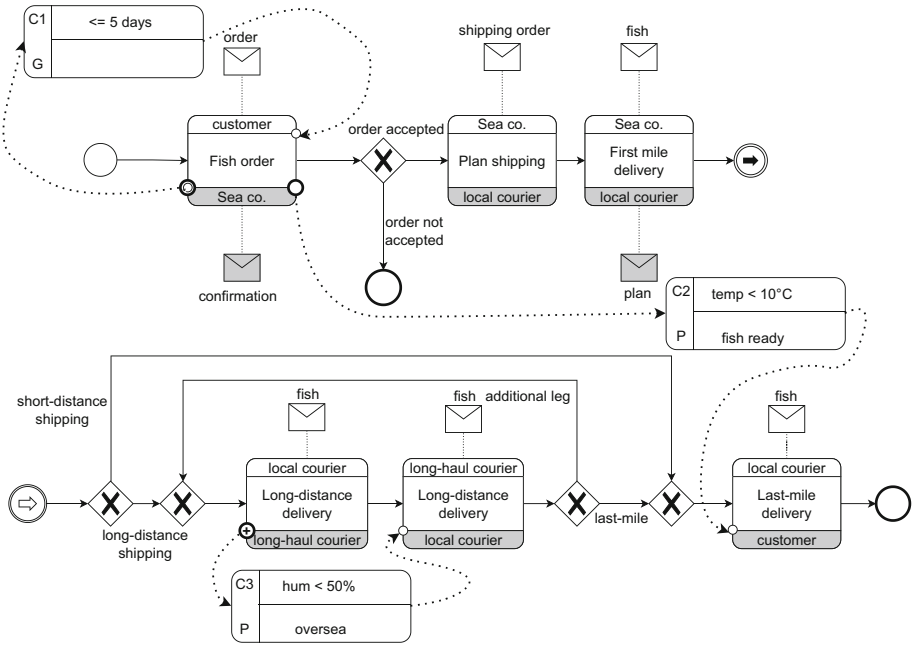


Fig. 5. Running example enriched with connection points

new monitoring instance specific to that batch). Being of type *all-instance*, it is expected that all fish will then be returned to a local courier and the constraint on humidity will be respected for all deliveries.

It is worth noticing that the introduced connection points are related to the beginning of the scope of the commitment, while the scope ending is still represented in the usual way, i.e., with a generic circle. This is because the effect in the evaluation of the commitment, which is actually also dependent on the non-conformance, is already considered in the type of connection point in the *scopeBegin*. Considering the any-instance or all-instance cases, the indication in the *scopeBegin* of these cases also indicates the behavior to be taken in evaluating the commitment at the *scopeEnd*.

It is worth mentioning that the multi-instance choreography tasks are not yet fully considered in the approach, and their effect is under investigation. This type of task has a significant influence on how the commitment is created. Thus, this will be reflected in the way the non-conformances can occur. For instance, the semantics associated with a commitment that starts from a single-instance task and ends in a multi-instance one has not been considered yet. Also, parallel multi-instance and sequential multi-instance tasks may require different approaches.

5.3 Extended Timed Commitment BPMN Compliant Metamodel

The Timed Commitment metamodel has been extended to integrate new connection point types to enhance the model's adaptability and flexibility. This metamodel is fully compliant with the BPMN specification. In this way, the proposed extension can be also accommodated in the BPMNI format.

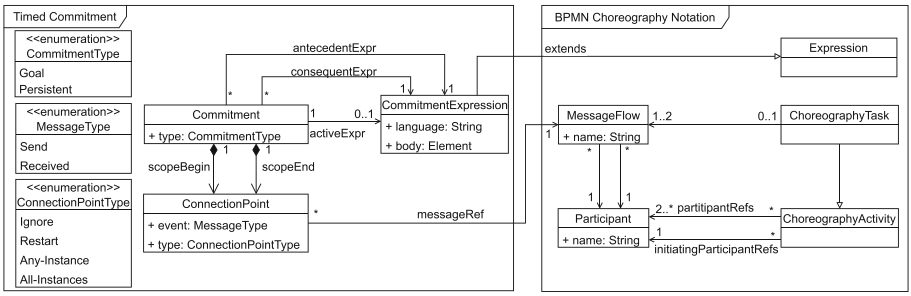


Fig. 6. Timed Commitment model with Connection Point Types

Notably, as illustrated in Fig. 6, a new attribute named *type*, of type *ConnectionPointType*, has been added to the ConnectionPoint class. This type is an enumeration that contains the introduced set of Connection Point types.

6 Monitoring Solution

In addition to the choreography model enriched with commitments, a monitoring system has been designed to verify the satisfaction of these requirements. To create a robust monitoring solution, which can cope with incomplete adherence to the choreography protocol, the proposed system exploits the information defined by the connection points to be able to continue in the monitoring phase even in case of non-conformance with the protocol itself.

The proposed solution is totally distributed and partially transparent to the systems that are in charge of the execution of the processes on the participant side and is composed of two main elements:

Commitment Supervisor (a.k.a. supervisor): Component capable of managing the life cycle of a commitment. For each commitment present in the model, the instantiation of a supervisor connected to it is provided for the period relating to the scope of the commitment.

Participant Process Sidecar (a.k.a. sidecar): Component that mediates between the internal process of a participant and the commitment supervisor.

Figure 7 shows on the right how these elements are used in the proposed solution referring to a generic scenario reported on the left. First of all, the sidecar processes are assumed to be able to intercept all the messages exchanged by the parties. This is a basic assumption that limits the transparency of the

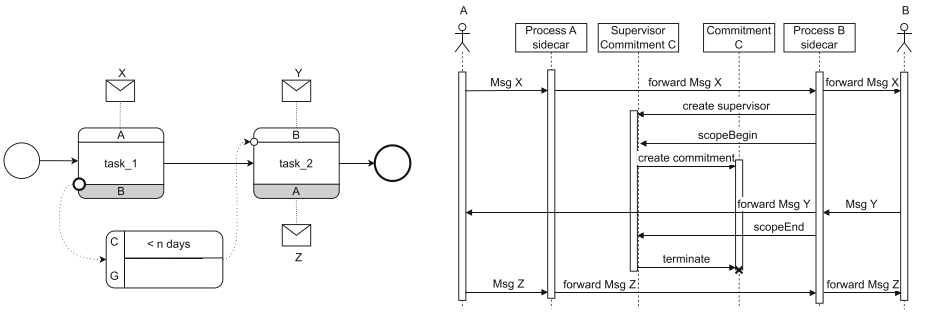


Fig. 7. Interaction between sidecar processes, supervisor commitment, and commitment

proposed method but it should be not so difficult to satisfy. Depending on the role of the message in the commitment life cycle, the sidecar process can decide, in addition to forwarding the message to the final destination, to interact with the supervisor. As shown in the figure, the scope of the commitment C starts when the message X is received by B and ends when the message Y is sent by B. Conversely, the message Z does not have any role.

On this basis, the sidecar process of B realizes, when receiving X, that the commitment has to start. Thus, it instantiates the supervisor and informs it about the *scopeBegin* which, in turn, will instantiate the commitment. While the commitment is active, the commitment is properly configured [2,11] to obtain the data required to evaluate the conditions defined. Finally, when B sends the message Y, the sidecar process of B reacts by sending the *scopeEnd* command to the supervisor, which causes the termination of the commitment. Conversely, the sidecar process of A does not react because the connector of the commitment is not related to the receiving of the message by A, thus the sidecar simply forwards the message to A. Finally, being the message Z not related to the commitment, it does not imply any action of any of the sidecars.

While the supervisor is closely linked to the concept of commitment and is therefore independent of the process, sidecars are automatically generated starting from the choreography model. As discussed in the next paragraphs, the generation of the sidecars takes into account the type of connection point as the reaction of the sidecar process depends on the semantics of the connection point.

6.1 Commitment Supervisor

Before describing the characteristics of the commitment supervisor it is useful to recall the life cycle of a commitment. As shown in Fig. 8, given a commitment included in a choreography, at the time of its instantiation it passes into a state of *conditional*. The commitment remains in this state until the antecedent expression is true, changing the state to *detached*, either it is brought to the *terminated* state if the commitment is *cancelled* or the *scopeEnd* is reached. In the detached state, depending on the nature of the commitment, i.e., persistent

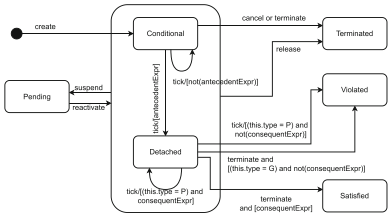


Fig. 8. Timed commitment state machine

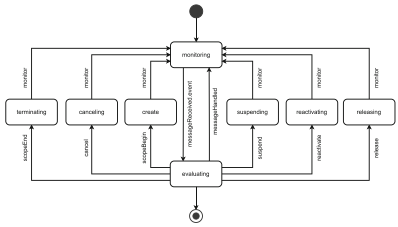


Fig. 9. Commitment supervisor state machine

or goal, the state can be moved to *violated* if the consequent is violated at least once during the time scope (for the persistence case) or when the *scopeEnd* is reached the consequent is false (for the goal case). On the contrary, the commitment ends in a state of *satisfied* if the consequent is true when the *scopeEnd* is reached. Finally, regardless of whether the commitment is in a conditional or detached state, any suspension requests bring the commitment to a pending state from which it is possible to return to the previous state with a *reactivate* event. Likewise, the commitment could also be *released* and therefore its status brought into terminated without any verification of its satisfaction or violation.

Based on this state machine, the supervisor has the role of obtaining from sidecar processes information about sending or receiving messages that define the *scopeBegin* and *scopeEnd*. In addition, the information that can always be obtained from messages or artifacts allows you to obtain the data necessary to process the various conditions expressed in the antecedent and consequent, to evaluate the satisfaction or violation of the commitment.

This behavior is summarized in the diagram of Fig. 9. Once a supervisor is created, it directly moves to a *monitoring* state, therefore listening to the messages exchanged between the parties. Not all messages are relevant, but only those that define the scopeBegin and the *scopeEnd* and the messages that contain information useful for evaluating the conditions expressed. While for the messages of the first type, the proposed solution obtains information directly from the choreography model, for the messages of the second type the configuration is still manual and it is part of the future development the definition of a mechanism that is able to obtain them from the model.

The *evaluating* state represents the capability of the supervisor, once a relevant message is captured in the monitoring state, to determine to which extent the managed commitment is affected. In particular, the commitment can be involved in two main ways:

– the message contains data relevant for the evaluation of the antecedent and consequent. Thus, the commitment reacts by changing, if requested, the current state accordingly (e.g., if the antecedent becomes true then the commitment moves from a *conditional* to a *detached* state). At the same time, the supervisor goes back to the monitoring state to analyze the next messages.

– the message contains data concerning the need to *create* a new commitment instance, i.e., when the received message is associated with the *scopeBegin*, or the need to *terminate* a commitment, i.e., when the received message is associated to the *scopeEnd*. In addition, the supervisor could also receive messages not included in the choreography model. These are out-of-bound messages that can be raised by the participants to *suspend* or *reactivate* the monitoring, or to force the termination of the process via a *release* to a *cancel*. These out-of-bound messages can be also related to cases in which the *scopeEnd* of a commitment cannot be reached because the executed branch does not involve the execution of the tasks determining the end of the commitment scope[3].

The second type of message is not represented in the choreography model but it can be used by the participants, during the execution of the choreography, to manage some particular situations that could require forcing the behavior of the commitment. Generally speaking, these messages are the starting point for extending the capabilities of the monitoring system to deal also with other non-conformances that are not related to the process but to the commitment itself, thus in addition to the cases discussed in Sect. 5.1. For instance, the participants can agree – at run-time – that a commitment that has been indicated in the model does not apply to the specific case. Thus it is reasonable to terminate. Similarly, due to some external events, the participants can decide to stop monitoring the temperature of the fish if the transport occurs during freezing days to save energy for the sensors used to capture this data.

6.2 Sidecar Process

The sidecar process has a central role in managing the behavior of the monitoring system in case of non-conformance. As discussed above, the sidecar process is generated automatically considering the connection point used to represent the commitment and the messages identifying the *scopeEnd* and *scopeBegin*. On this basis, it is possible to derive the behavior of the sidecar processes in the four considered cases as represented in Fig. 10. Each case is reported with an example of commitment usage in a hypothetical choreography, the sidecar process associated with the debtor, i.e., process B, and the sidecar process associated with the creditor, i.e., process A.

As shown the sidecar processes are assumed to automatically start when the main processes start. Moreover, these processes are configured to listen to the messages[4] that are connected to the *scopeBegin* and *scopeEnd* and when they arrive, a message containing the instruction is sent to the supervisor which will

[3] For the sake of simplicity, commitments of type types (e.g., connecting *first-mile delivery* that is always executed, with *long-distance delivery* which execution could be optional) are not discussed in the paper but they are fully supported by the mentioned out-of-bound messages.

[4] To better focus on the meaningful aspects of the proposed approach, the processes do not represent the portion of the flow that intercepts and forwards the non-relevant messages as discussed in Fig. 7.

act accordingly. The same processes are valid also in case the scope starts or ends with the sending of a message (while in the provided examples are not connected to the receiving). In fact, although it is relevant to react when, for instance A, sends the message X, from a sidecar perspective this will be mapped with e received event as it corresponds to the interception of the message while moving towards B. Moreover, all the sidecar processes have a concurrent branch used to manage situations in which the process ends even if the commitment is still active. This is possible, as mentioned above, because there could be process execution traces that do not ensure that the task related to the *scopeEnd* is reached. In this case, the commitment is simply terminated without any evaluation.

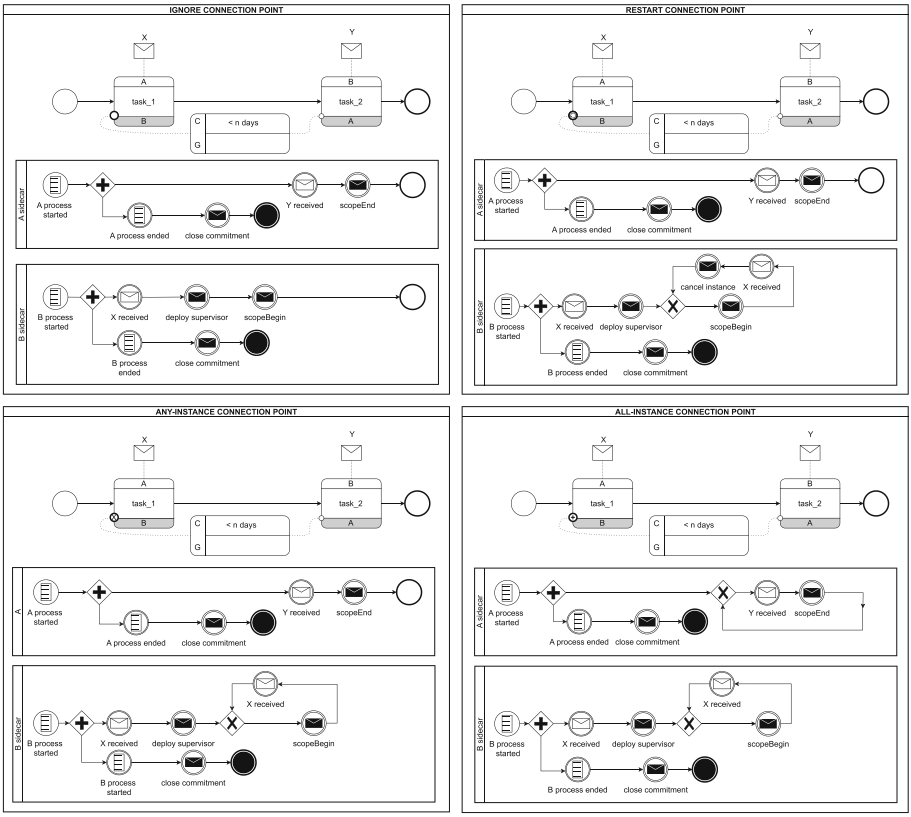


Fig. 10. Sidecar processes related to the considered non-conformities

Entering in detail, it is possible to notice the different behaviors in the four cases. Notably, in the *ignore* the first time X is received, the supervisor of the considered commitment is created and informed about the need to create an instance of the commit (this is a portion in common with all the other situations). Then, the sidecar process B will ignore any other messages X. Conversely, the

sidecar process of A will wait for Y to arrive and, if so, inform the supervisor that the commitment should be closed.

In the *restart* case, after receiving the first message X, B is allowed to manage additional messages of the same type. If this happens, the sidecar will inform the supervisor to cancel the current instance of the commitment and create a new one, thus restarting the obligation. On the other side, the sidecar of A remains the same as before.

In the *any-instance* and *all-instance* cases the behavior of sidecar B is the same, i.e., when a second X is received a new *scopeBegin* message is sent to the supervisor to indicate that a new instance of the commitment must be created. Conversely, the behavior is different for the A sidecar. In fact, in the *any-instance* case, it is enough to receive one Y message to close all the commitments, while in the *all-instance* case, each X message should be followed by the same number of Y messages to allow the correct closing of all the commitment. The need to check the consistency between the number of *scopeEnd* and *scopeBegin*, as well as, the possibility to close the commitment with a single *scopeEnd* is not evident in the sidecar processes as it is a task for the supervisor that, once created, it is informed about the policy to follow in managing this situation.

For the *all-instance* case, it is worth highlighting that the supervisor and the sidecars are not in charge of deciding whether the commitment is satisfied or not. In fact, after closing all the commitments some of them might be evaluated as violated and some others as satisfied. Thus, at the system level, it remains unclear the final decision. For this reason, we assume (although it is not yet integrated into the proposed notation) that the modeler will define which is the decision policy in these cases as it is definitely domain-dependent.

7 Validation

A software solution able to implement the discussed monitoring systems has been developed and it is currently available on a public code repository[5].

The monitoring platform assumes that one of the participants in the choreography has deployed their process within a BPMS. In the same BPMS, a set of sidecar processes related to the commitments for which the process is either debtor or creditor are also deployed. Figure 11 illustrates the situation of Sea co. which is involved in the commitments $C1$ and $C2$.

Sidecar processes are subscribers of messages from external participants for whom they are creditors or debtors. This allows the evolution of these sidecar processes according to the respective logic defined in Fig. 10, which may also require interaction with the *Commitment Supervisor* which has been offered as a container exposing a REST service. This permits a simple deployment in different environments. Being the supervisor associated with a commitment, depending on the agreement between the debtor and the creditor, as well as the technical aspects related to where the data needed to evaluate the expressions in the

[5] https://github.com/ISGroup-Polimi/choreo-timed-commitment.

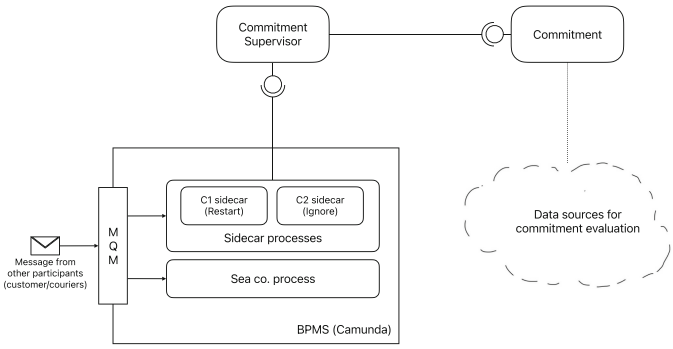


Fig. 11. Overall architecture of the monitoring platform

commitment, few assumptions can be made concerning which location and platform will host the supervisor. The API interface of the supervisor offers all the methods to map the events reported in the state machine in Fig. 9 as well as the request for information about the number of managed commitment instances, their current status, and other management operations.

For validation purposes, these processes have been deployed and tested on Camunda, although any BPMS can be used for this purpose. It might be also possible, that no BPMS is used but, on the contrary, ad hoc implementation of the provided sidecar processes models is developed. Using the reference example adopted in this paper, we simulate the execution of the process involving the different participants by introducing the considered non-conformances to check whether the behavior of the overall system is able to continuously monitor the execution of the commitments.

8 Concluding Remarks

This paper presented an approach for modeling business requirements using timed commitments that can be inserted directly into a BPMN choreography model. The particularity of the approach also lies in the possibility of defining the behavior of the monitoring system in case of non-conformance, starting from instructions inserted directly in the model. The approach was implemented with a distributed application solution based on containerized services for supervising commitments and orchestrating processes capable of handling non-conformance.

Future work, as also described in the paper, will focus mainly on the possibility of managing further types of non-conformances related not only to messages but also to the availability of data useful for evaluating commitments. Other aspects that require necessary in-depth studies concern the possibility and methods of collecting the data necessary to evaluate the expressions of the antecedent and the consequent of the commitments. A deployment that gives visibility of all the information necessary for the correct evolution of the commitment state machines is also subject for future studies.

Acknowledgements. This work has been partially funded by the MICS Extended Partnership financed by the EU - NextGeneration EU - PNRR program.

References

1. Aburub, F., Odeh, M., Beeson, I.: Modelling non-functional requirements of business processes. Inf. Softw. Technol. **49**(11), 1162–1171 (2007). https://doi.org/10.1016/j.infsof.2006.12.002
2. Bertolini, M., Meroni, G., Plebani, P.: Trusted compliance checking on blockchain with commitments: a model-driven approach. In: BPM 2023 Forum, Proceedings. LNBIP, vol. 490, pp. 3–19. Springer. https://doi.org/10.1007/978-3-031-41623-1_1
3. Capel, M.I., Mendoza, L.E.: Choreography modeling compliance for timed business models. In: EOMAS 2014, Held at CAiSE 2014 Selected Papers. LNBIP, vol. 191, pp. 202–218. Springer (2014). https://doi.org/10.1007/978-3-662-44860-1_12
4. Combi, C., Oliboni, B., Zerbato, F.: A modular approach to the specification and management of time duration constraints in BPMN. Inf. Syst. **84**, 111–144 (2019). https://doi.org/10.1016/j.is.2019.04.010
5. Fdhila, W., Knuplesch, D., Rinderle-Ma, S., Reichert, M.: Verifying compliance in process choreographies: Foundations, algorithms, and implementation. Inf. Syst. **108**, 101983 (2022). https://doi.org/10.1016/j.is.2022.101983
6. Governatori, G., Rotolo, A.: Norm compliance in business process modeling. J. Bus. Process Anal. **22**(1), 78–102 (2014)
7. Hashmi, M., Governatori, G., Lam, H.-P., Wynn, M.T.: Are we done with business process compliance: state of the art and challenges ahead. Knowl. Inf. Syst. **57**(1), 79–133 (2018). https://doi.org/10.1007/s10115-017-1142-1
8. Knuplesch, D., Reichert, M., Mangler, J., Rinderle-Ma, S., Fdhila, W.: Towards compliance of cross-organizational processes and their changes (2012). https://doi.org/10.1007/978-3-642-36285-9_65
9. Köpke, J., Meroni, G., Salnitri, M.: Designing secure business processes for blockchains with SecBPMN2BC. Future Gener. Comput. Syst. **141**, 382–398 (2023). https://doi.org/10.1016/j.future.2022.11.013
10. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.: Compliance monitoring in business processes: Functionalities, application, and tool-support. Inf. Syst. **54** (2015). https://doi.org/10.1016/j.is.2015.02.007
11. Montali, M., Plebani, P.: IoT-based compliance checking of multi-party business processes modeled with commitments. In: Service-Oriented and Cloud Computing, pp. 179–195. Springer International Publishing, Cham (2017)
12. Telang, P.R., Singh, M.P.: Specifying and verifying cross-organizational business models: an agent-oriented approach. IEEE Trans. Serv. Comput. **5**(3) (2012). https://doi.org/10.1109/TSC.2011.4
13. Verdicchio, M., Colombetti, M.: Commitments for agent-based supply chain management. ACM SIGecom Exchanges **3**(1), 18–29 (2002)
14. Zarour, K., Benmerzoug, D., Guermouche, N., Drira, K.: A systematic literature review on BPMN extensions. Bus. Process. Manag. J. **26**(6), 1473–1503 (2020). https://doi.org/10.1108/BPMJ-01-2019-0040

A Unified View on Data Object States

Maximilian König(✉), Raban Gießler, William Brandt, Anjo Seidel, and Mathias Weske

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
{maximilian.koenig,anjo.seidel,mathias.weske}@hpi.de,
{raban.giessler,william.brandt}@student.hpi.de

Abstract. Process models play a key role in business process management. While control flow aspects are well understood, data aspects received less attention. In process models, data is commonly represented by data objects, and data object states are employed as abstractions from concrete properties of data objects. However, there is little agreement on how those states can actually be defined. This paper proposes a framework for the definition of data object states. The framework generalizes and consolidates the dimensions that make up data object states in related work. It supports concurrent states to capture different perspectives on a given object. The work is evaluated by (i) mapping the state definitions of existing approaches to the framework to demonstrate its unifying nature, and (ii) a prototypical implementation based on decision tables showcasing the applicability of the framework.

Keywords: Data in Business Processes · Process Modeling · Object-centric Processes · Data Object States · Concurrent States

1 Introduction

Business process management supports organizations in managing business processes along their entire lifecycle [28]. Business process models are a core concept to represent relevant activities, events, participants, and data. Traditionally, the main focus has been on the control flow, i.e., the ordering of activities and events [15]. Data is tightly coupled to the process execution, since data availability serves as a precondition for activities which manipulate data. To provide a glue between process and data, various process modeling approaches agree on the concept of data objects, serving as an abstraction for the real-world business objects involved in processes [14,18,23]. Further, to capture the evolving nature of business objects, the notion of data object states is employed. Especially in the light of the currently emerging object-centric paradigm, where a system is described as the interaction of different objects with individual behavior, these concepts have become more prevalent [7].

While many approaches rely on data objects and states, their perspective on an object's state varies significantly, resulting in different dimensions contributing to object state definitions. Some works propose state definitions as logical

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 259–276, 2025.
https://doi.org/10.1007/978-3-031-94571-7_15

formulae based solely on an object's attribute values [8,21]. Other approaches suggest additional factors influencing an object's current state, namely its links to other objects and the history of activities and events interacting with the object [5,7,23]. Since concurrent branches of a given process may operate on the same data object, a notion of concurrency in state definitions is discussed [23]. In summary, all these approaches share a common concept, data object states, though their interpretations of the concept differ significantly. Therefore, this paper explores the following research question: (1) What are the properties that define the state of a data object in current modeling approaches, and how does each approach cover them? Based thereon, assuming there are overlaps between these approaches that allow to relate them, we investigate (2) whether these properties can be integrated into a unified definition of data object states, aiming to establish a starting point for the comparability of the approaches.

To address these research questions, we analyze existing literature to identify the dimensions constituting data object state definitions and introduce a formal definition that captures all these dimensions. Thereby, we lay the foundation for future endeavors to derive general analysis methods on state definitions across different approaches. The approach is evaluated by mapping the state definitions of a selection of the analyzed approaches to the unified definition and a prototypical implementation showcasing its technical feasibility.

The remainder of the paper is structured as follows: In Sect. 2, we motivate the dimensions constituting data object states in related work alongside an example process. Based thereon, we analyze process modeling approaches that support data objects regarding their coverage of the identified dimensions in Sect. 3. Afterward, Sect. 4 details the formalization of our unified data object state definition. Section 5 then presents a two-fold evaluation of that definition before Sect. 6 discusses and concludes the paper.

2 Motivation

Using data objects and their states as an abstraction of concrete data is a common concept in business process modeling. In the following, we analyze existing process modeling approaches regarding the dimensions constituting an object's state. These dimensions will be illustrated by an order-to-cash (O2C) process. In the O2C process, the following classes of objects are involved: *order*, *item*, *invoice*, and *payment*. Their structure, i.e., attributes, and associations are represented in a UML class diagram [19] shown in Fig. 1.

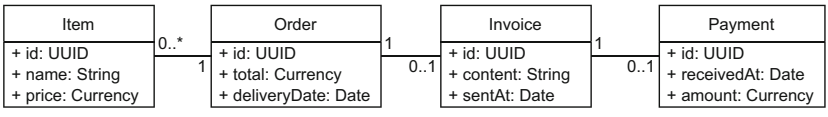


Fig. 1. UML class diagram of the data object types in the order-to-cash process.

Data models define the domain for the first dimension constituting an object's state: constraints on the attribute values of an object at runtime [2,8,13,14,21]. In the simplest form, e.g., for an order to be in state *valid*, its *total* attribute must have a value greater than 0. Such a state definition can then be evaluated on concrete objects at runtime. To visualize that, a database snapshot including a set of objects that have been created during an execution of the O2C process is shown in Fig. 2. Since the order *O1* has a *total* of 150, it is in state *valid*.

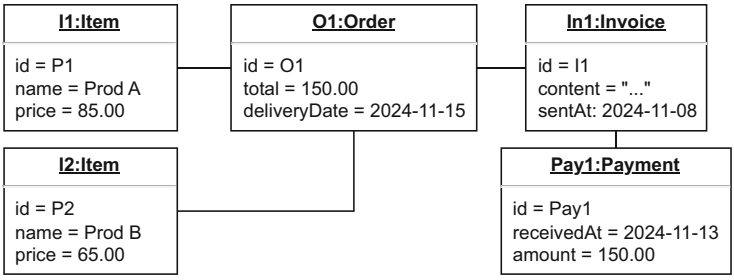


Fig. 2. UML object diagram showing an order *O1* and its related objects.

This also allows us to observe a second dimension relevant to data object states. Data objects are not processed in isolation, but typically have dependencies on each other, represented as associations in the class diagram and links in the object diagram. Links, often captured using foreign key attributes, indicate that these objects are related. Therefore, state definitions in related work include conditions on the existence of links to objects of other classes, and potentially the properties or state of these linked objects as well [5–7,14]. In the O2C process, a linked invoice object might be required for an order to be in state *invoiced*.

Table 1. Tabular view of the event log in the information system leading to the database snapshot in Fig. 2.

Event ID	Event Type	Timestamp	Related Objects
e1	Create and send invoice	2024-11-08 05:00	O1, In1
e2	Manufacture items	2024-11-13 09:00	O1, I1, I2
e3	Receive payment	2024-11-13 14:00	Pay1
e4	Deliver order	2024-11-15 14:00	O1, I1, I2
e5	Process payment	2024-11-16 11:00	Pay1, O1, In1

Data objects change over time. In an information system, this can be captured through the events that interact with an object by, e.g., reading its contents, manipulating its attribute values or linking it to another object. A data

object's event history constitutes the third dimension employed in related work to define its state [1,4,5,16,23]. Approaches vary in the depth of events considered, ranging from only the last related event to the entire event history. In the O2C process, an order being in state *archived* might depend on an event *Archive order* to have occurred. We assume for every event to have a unique identifier, belong to an event type, have a timestamp, and specify the objects it interacts with. For example, a selection of events for the O2C process is depicted in Table 1.

In summary, we identify three dimensions in related work that can determine a data object's state: the current attribute values, its links to other objects and their respective states, and the history of event occurrences.

In an organization, multiple processes or concurrent branches in the same process may operate on an overlapping set of objects simultaneously. For example, in the O2C process, procuring and delivering items may happen concurrently to invoicing and payment, as shown in the BPMN process model in Fig. 3. Both concurrent branches process the same order. To track progress, they use states for the order that are only relevant in the context of their respective branch. For example, procuring the ordered items does not depend on whether an invoice has been sent yet. This leads to the observation that a given data object might be in several states at the same time, depending on the perspective it is viewed from, as discussed in [13,21,23].

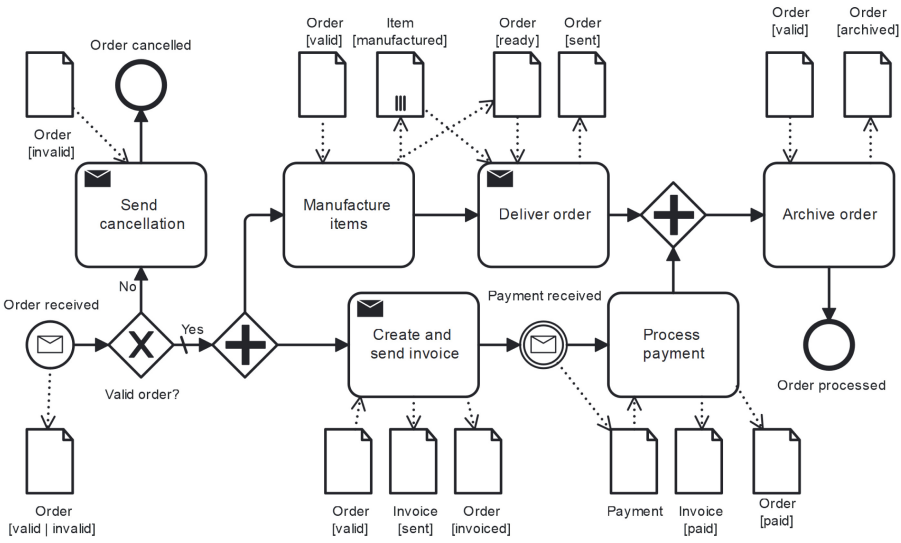


Fig. 3. Simplified BPMN process model of the order-to-cash process, where payment processing and item dispatching occur concurrently. Both require the order to be in specific states (e.g., *paid* and *sent*), highlighting the need for concurrent states. Note that the data object visualizations do not follow standard BPMN semantics but are mainly used for visualization purposes.

This complex state behavior as well as the fundamental dimensions constituting a data object's state have yet to be brought together. Existing approaches are silos with widely differing state definitions that currently cannot be related to each other. However, data objects and their states are the basis for harmonizing process and data flow. Therefore, this paper aims to introduce a unifying framework as a first step towards bridging these silos and enabling comparability between approaches. To be compatible with existing approaches, it must cover all identified dimensions, namely (i) attribute values, (ii) object links, and (iii) objects' event history, and (iv) permit overlapping state definitions to capture concurrent processing of objects.

3 Data Object States in Related Work

The previous section introduces the dimensions constituting data object state definitions in related approaches. To address research question (1), this section provides an overview of the reviewed literature and the dimensions covered by each approach. A summary of the analysis can be found in Table 2.

For the analysis, we only consider approaches that support an explicit representation of data objects in different states. For each approach, we evaluate which of the identified dimensions in Sect. 2 *explicitly* contributes to defining data object states. If an approach can capture a dimension in general, but the employed state definitions do not allow for constraints on that dimension, it will be classified as not explicit, and hence, as not supported. Full support for attribute values implies that a state definition may include constraints on one or more attribute values. Object links are partially supported if an object state may depend on the existence of one or multiple linked objects. For full support, additional constraints on the linked objects' state must be allowed. The event history dimension is fully supported if states may depend on a sequence of occurred events, while partial support indicates restrictions on these dependencies, e.g., by only allowing the last event to be considered.

Many process modeling approaches follow a multi-model paradigm to represent different perspectives: A data model describes the structure of relevant objects, their attributes, and their relationships, a lifecycle model per class represents the internal behavior of its objects, and an inter-object behavior model specifies the interaction between objects of different classes. Examples of this kind of framework are MERODE [23,24], BAUML [4], and fragment-based case management (fCM) [12]. While all of these approaches capture attribute values and object links, a state in their lifecycle models depends solely on the history of events it interacted with and neither on attribute values nor on object links. Of this selection, only MERODE provides a means for an object to be in multiple states at the same time, using hierarchical state charts [10]. PHILharmonicFlows [14] uses similar model types. However, instead of occurred events, object states primarily depend on attributes values, specified as stages within the state of an object's lifecycle, and on other objects being in a specific state, as defined in a so-called coordination process. Concurrent states are not supported.

Table 2. Comparison of process modeling approaches including a data representation regarding the dimensions constituting a data object's state. ✓ indicates full support, ∼ is used for partial support, and − for no support.

	Attr. Values	Object Links	Event History	Concur. States
MERODE [23]	−	−	✓	✓
PHILharmonicFlows [14]	✓	✓	∼	−
BAUML [4]	−	−	✓	−
fCM [12]	−	−	✓	−
GSM [13]	✓	∼	✓	✓
OCPN [1]	−	∼	✓	✓
Synchronous Proclets [6]	−	∼	✓	✓
OPID [7]	−	∼	✓	✓
ISML [22]	−	∼	✓	✓
DB-nets [17]	✓	∼	✓	✓
OC-DCR [5]	−	✓	✓	✓
Pérez-Álvarez et al. [21]	✓	∼	−	✓
Gómez-López et al. [8]	✓	∼	−	−
Meyer et al. [16]	−	−	✓	−
Bano et al. [2]	✓	∼	−	−
BPMN [18], UML AD [19]	−	−	∼	−

Artifact-centric modeling approaches follow a similar multi-model structure. Instead of an overarching data model, each artifact consists of an information model next to its lifecycle, describing its data structure [26]. A representative of this kind of framework is the Guard-Stage-Milestone (GSM) approach [13], where an object's state is dependent on its attribute values, including foreign keys to check the existence of related objects, and occurred events. Since the lifecycles are divided into stages that can be processed concurrently, an object can be in multiple states simultaneously.

Another perspective on the state of an object can be found in a number of recent Petri net-based object-centric process modeling approaches, prominently OCPNs [1], OPIDs [7], and synchronous proclets [6]. A Petri net is composed of places that can hold tokens and transitions that consume tokens from incoming places and produce tokens to outgoing places. In those approaches, each place references an object type and, thus, can hold objects as tokens. The place specifies the state in which an object may be. Hence, the state of an object depends primarily on the transitions it interacted with. In addition, transitions may require tokens of different types, whereby a notion of object relations is employed. Since there may be multiple tokens carrying the same object's ID on different places, concurrent states are supported. ISML [22] builds on Petri nets with identifiers (PNIDs) [11] and introduces persistence by linking transitions to abstract transactions on an information model that supports entity types and

relations, but not attributes. DB-nets [17] are a data-aware approach connecting Petri nets to a persistence layer by assigning data update actions to transitions and introducing additional view places defined as queries on the data model, i.e., attribute values and link existence.

Christfort et al. introduce an object-centric extension to DCR graphs [5], where objects can be explicitly modeled, including states relying on occurred events and dependencies on other objects' states.

Activity-centric modeling languages such as BPMN 2.0 [18] and UML activity diagrams (AD) [19] also support the notion of data objects with states. Considering these model types in isolation, the state solely depends on executed activities. However, both can be combined with other model types to cover additional aspects in their state definition, as demonstrated by MERODE (BPMN) [24] and BAUML (UML AD) [4].

Based on BPMN process models, Pérez-Álvarez et al. use a supplementary data model to define the structure of data objects and propose a domain-specific language (DSL) for state definitions based on object attributes [21]. As a means of verification, they investigate the relations between state definitions resulting from their approach, which may be disjoint, overlapping, or hierarchical. Another DSL has been presented by Gómez-López et al., where attributes, including foreign keys to capture object relations, form the basis for state definitions [8].

Instead of defining data object states manually, their automated derivation from different sources has been investigated. Meyer et al. analyze activity labels of process models to derive the objects and their states [16]. Since no additional data model is utilized, the states depend solely on the control flow. Based on attribute manipulations observed from an event log, Bano et al. enrich discovered process models with data objects and their behavior [2].

This overview of existing approaches shows that the concept of data object states is widely used. Its main purpose is to discretize properties an object must fulfill to enable the next process steps. However, there is no agreement on the constituents of such a state definition, as existing works cover different subsets of dimensions. To bridge these perspectives, this paper presents a unifying framework for defining data object states based on the identified dimensions.

4 State Definition

As elaborated in the prior sections, data object states can be based on (i) attribute values, (ii) object links, and (iii) the object's event history. Additionally, (iv) an object may be in multiple states concurrently. Based on these requirements, this section provides a unifying definition of data object states.

First, a deeper understanding of the relationship between potentially concurrent data object states is required. If concurrent processes operate on the same object, they need individual perspectives on its state. Hence, the same object can be in different states at the same time. This can be realized by constraints that are evaluated on the object's properties independently. As a result, these state definitions can be overlapping or even hierarchically structured.

Hierarchical states can be observed wherever the constraints defined in one state definition are fully contained by the constraints of another state definition. In the O2C example discussed earlier, the *valid* state for orders is of a very general nature, defining the validity of an order based on its *total* attribute being greater than 0. Other states, such as *invoiced*, require at least the same constraints to be fulfilled while adding further conditions such as a sent invoice. Therefore, *invoiced* is a sub-state of *valid*, creating a hierarchy of states. This means, an order in state *invoiced* is also always in state *valid*.

With partially overlapping constraints in different state definitions, data objects can be in either combination of these states at the same time. Consider again the state *invoiced* of an order, which depends on the existence of a linked invoice object. Another state, *sent*, solely depends on the occurrence of a *Deliver order* event for the same order. Since these two definitions do not exclude each other, the same order can be in both states at the same time if both conditions are fulfilled. If only one constraint is met, it is only in the respective state.

Based on these considerations, we introduce a formalization allowing for overlapping state definitions while also covering all identified dimensions. In order to do so, we first need to define the environment in which objects' state definitions are interpreted, which we call a system model:

Definition 1 (System Model). *Let A be the set of attributes, where each $a \in A$ has a name and a data type $dom(a)$, ET be the set of event types, and C be the set of data classes. A* system model *is a tuple $M = (ET, C)$, where each class $c = (id, A_c, Z_c) \in C$ is a tuple of an identifier id, a set of the class's attributes $A_c \subseteq A$, and the states Z_c that an object of the class can assume.*

To evaluate a data object's state in the context of a system model, a snapshot of a system at a certain point in time is required. We call that a system instance:

Definition 2 (System Instance). *Let $\mathcal{I}_M$ be the set of all system instances of a system model $M = (ET, C)$, such that each system instance $i \in \mathcal{I}_M$ is defined as a tuple $i = (O, L, E, ts_i)$ consisting of:*

- *a set of objects O where each object $o \in O$ is defined as a tuple $o = (id, c, V)$ being identified through an id, belonging to an object class $c = (id_c, A_c, Z_c) \in C$, and having a set of attribute values $V : A_c \rightarrow dom(A_c)$,*
- *an irreflexive relation of links between objects $L \subseteq O \times O$,*
- *an event history E where each event $e \in E$ is defined as a tuple $e = (ts, et, O_e)$ with a unique time stamp ts so the event history is totally ordered, an event type $et \in ET$, and a set of related objects $O_e \subseteq O$, and*
- *a timestamp of the system instance ts_i s.t. $\forall e \in E : e.ts \leq ts_i$.*

Note that we assume unique timestamps per event, i.e., that no two events occur at the exact same time, and that multiple links between the same two objects are consolidated in a single link. An example for a system instance can

be found in Sect. 2. Figure 2 describes the objects and links, while Table 1 lists the occurred events.

As explained, an object can be in multiple states simultaneously, if it is processed by multiple processes concurrently. To capture these concurrent states, we formalize data object state definitions as individual functions $f_{c[z]} \in F_c$ for each state $z \in Z_c$ of class $c \in C$. These states can be evaluated independently. An object $o \in O_c$ is in state z iff the state function $f_{c[z]}$ evaluates to true.

Definition 3 (Data Object State Definition). *A data object state $z \in Z_c$ for a class $c \in C$ in a system model $M = (ET, C)$ is defined by the function $f_{c[z]} : \mathcal{I}_M \times O_c \rightarrow Boolean$. The function describes the conditions an object $o \in O_c$ has to satisfy to be in state z given a system instance $i \in \mathcal{I}_M$ of system model M as a first-order logic expression. F_c denotes the set of all state definitions for class c.*

Since the above definition is very generic, the following paragraphs show its application to the three dimensions according to the O2C process.

The first dimension covers conditions on the attributes of the specific class. In the motivating example, the *valid* state of the order class can be specified based solely on its *total* attribute, such that $f_{order[valid]}(i, o) := o.total > 0$.

The scope of influence on an object's state extends beyond the object itself. For example, the existence of a payment associated with an invoice affects its state through the object link: once a payment is received, the invoice is settled, transitioning it to state *paid*. Note that this interdependence becomes evident only when considering object links. The following support functions are defined to facilitate presentation.

$$l(i, o_1, o_2) := (o_1, o_2) \in i.L \vee (o_2, o_1) \in i.L$$
$$rL(i, c, o) := \{o_l \in i.O \mid l(i, o, o_l) \wedge o_l.c = c\}$$

The function $l(i, o_1, o_2)$ checks whether a link exists between two objects o_1 and o_2 in a given system instance i. Utilizing that, $rL(i, c, o)$ returns the set of objects of class c with which a given object o has a link in system instance i. Now we can express the desired constraint that an invoice is in state *paid* iff it is linked to exactly one payment: $f_{invoice[paid]}(i, o) := |\{p \in rL(i, payment, o)\}| = 1$.

In the O2C process, the state of an order changes to *paid* when its associated invoice is in state *paid*. Hence, the state depends not only on the object link but also on the state of the related object. To model this, a helper function $s(i, z, o)$ is introduced, which verifies whether a given object o satisfies a specific state z by evaluating the corresponding state definition for that object in the current system instance i, such that $s(i, z, o) := f_{o.c[z]}(i, o) = true$. Using this helper function, the state *paid* for the order class can be specified as $f_{order[paid]}(i, o) := |\{o_l \in rL(i, invoice, o) \mid s(i, paid, o_l) = true\}| = 1$.

Besides the data, events can also influence the state of a data object. For example, *Archive order* is an activity that documents an administrative step in which the order is printed and stored. This operation does not directly affect the attributes of the order object or influence object links. Instead, an order is in

state *archived* as soon as an event e of type $e.et =$ "*Archive order*" occurs which interacts with the respective order data object, such that $f_{order[archived]}(i, o) := \exists e \in i.E : o \in e.O_e \wedge e.et =$ "*Archive order*".

5 Evaluation

To evaluate the framework introduced, we show how different existing modeling approaches can be compared with respect to their definitions of data states before reporting on a prototypical implementation of the framework.

5.1 Evaluation of Unification

This section discusses the unifying nature of the framework by mapping a selection of analyzed approaches from Sect. 3 to the state definition and showing how those approaches can be compared to each other. First, a mapping to object-centric Petri nets with identifiers (OPIDs) [7] is presented formally. Thereafter, we provide the intuition for mapping two additional approaches based on the O2C process introduced in Sect. 2. The selected approaches, MERODE [23] and PHILharmonicFlows [14], support a mostly disjoint set of dimensions for their state definitions, whereby in total all four dimensions are covered (cf. Table 2).

Figure 4 shows an OPID as representative for Petri net-based modeling approaches. Places have a data type, called color, and may hold tokens to capture the state of the net. Each token is assigned an identifier representing an object. Hence, the token distribution for one identifier determines the state of the according object. The color of a place may involve multiple object types, signaling that only tokens holding tuples of the respective object types are allowed on that place. For example, in Fig. 4, tokens in yellow places represent orders while those in orange places refer to $Order \times Item$ links. Double-lined arcs indicate variable flows that can carry multiple tokens at once.

We simplify the definition of OPIDs by Gianola et al. [7] as follows. Let $\mathcal{A}$ be a set of activities, OT be the set of object types, and $\mathcal{C}$ be the set of colors consisting of all $ot_1 \times \cdots \times ot_m$ such that $m \geq 1$ and $ot_i \in OT$ for all $1 \leq i \leq m$.

Definition 4 (OPID). *An object-centric Petri net with identifiers is defined as a tuple* $N = (P, T, OT, \texttt{color}, \texttt{label}, F)$ *consisting of:*

- *finite sets of places* P *and transitions* T *with* $P \cap T = \emptyset$,
- *a function* $\texttt{color} : P \to \mathcal{C}$ *assigning each place a color,*
- *a function* $\texttt{label} : T \to \mathcal{A} \cup \{\tau\}$ *labeling all transitions, and*
- *a flow relation* $F \subseteq (P \times T) \cup (T \times P)$.

To access the preceding and succeeding transitions of a given place, respectively, the functions $\bullet p = \{t \mid (t, p) \in F\}$ and $p\bullet = \{t \mid (p, t) \in F\}$ are used.

Based on this formalization, OPIDs are mapped to the state definition presented in this paper. Given an OPID $N = (P, T, OT, \texttt{color}, \texttt{label}, F)$, we derive

a system model $M = (ET, C)$ with a set of event types $ET = \mathcal{A}$ corresponding to the set of transition labels and a set of classes C containing an element $c \in C$ for each $ot \in OT$, such that $c = (ot, A_c, Z_c)$. Since OPIDs do not support attributes, for every class $c \in C$, the set of attributes A_c is empty.

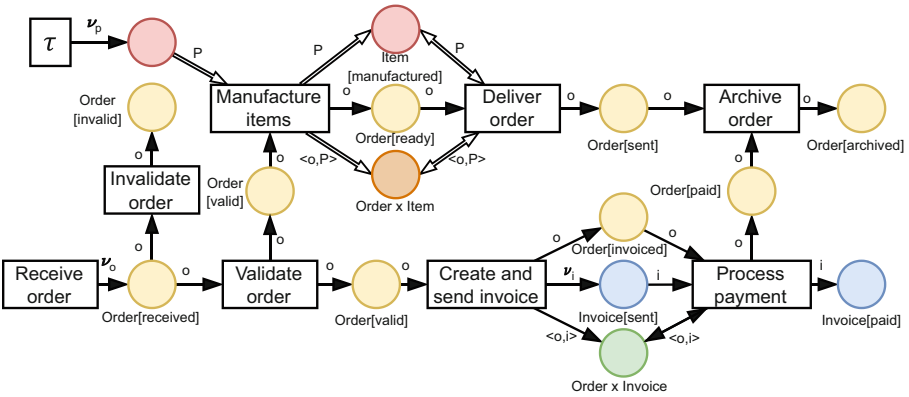


Fig. 4. O2C example in Fig. 3 modeled as an OPID. The color of a place indicates its data type: Order (yellow), Item (red), Invoice (blue), and object links $Order \times Item$ (orange) and $Order \times Invoice$ (green).

Before we can introduce the set of states Z_c, the definition of a system instance $i \in \mathcal{I}_M$ is required. According to Definition 2, a set of objects O, links L, events E, and the timestamp ts_i are required. O is the set of tokens in places whose `color` is a single object type, each represented as a tuple $(id, c, \emptyset) \in O$, where id is its identifier, c its object type, and $\emptyset$ the empty set of attribute values. L is the set of tokens in places whose `color` comprises exactly two object types. This interpretation is limited to binary links. However, arbitrary n-ary ($n \geq 3$) links can simply be refactored to binary links using association classes. For E, we assume that the trace leading to the current state is available, including the transition bindings. If timestamps are not available, we use logical timestamps that preserve event ordering.

The state of an individual object depends on which places hold a token with the respective object identifier. Hence, if we encode each place into a state, we can capture the overall state. To do so, we create a separate state definition for each place $p \in P$. We differentiate between places whose `color` comprises either one or two object types, as will be discussed below.

For any place $p \in P$ where $|\texttt{color}(p)| = 1$, we consider an object o to be in state $c[p]$ iff there is an occurrence of any directly preceding transition of p that is not eventually followed by any directly succeeding transition of p. Formally:

$$f_{c[p]}(i,o) := \exists (ts, et, O_e) \in i.E : \; o \in O_e \wedge et \in \bigcup_{t \in {}^\bullet p} \texttt{label}(t) \wedge \nexists (ts', et', O'_e) \in i.E : \; o \in O'_e \wedge et' \in \bigcup_{t \in p^\bullet} \texttt{label}(t) \wedge et' \notin \bigcup_{t \in {}^\bullet p} \texttt{label}(t) \wedge ts < ts'$$

To visualize this definition, consider place *Order[ready]* in Fig. 4. In a given system instance, we would look for an occurrence of *Manufacture items* that is not eventually followed by *Deliver order*. In that case, a token would be in place *Order[ready]* and the state definition would be fulfilled. Since we define every place as a separate state, our definition also covers tokens with the same identifier in multiple places, as they are evaluated independently.

For any place $p \in P$ where $|\texttt{color}(p)| = 2$, the place holds tokens representing links between two objects, e.g., $Order \times Item$. Therefore, the state definitions for these places require at least one such link to be established. For every $ot \in \texttt{color}(p)$ we define a new state $f_{c[\texttt{color}(p)]}$ for the corresponding class $c \in C$ as:

$$f_{c[\texttt{color}(p)]}(i,o) := \exists o' \in i.O : o'.c.id \in \texttt{color}(p) \wedge o'.c \neq c \wedge l(i,o,o')$$

As defined in Sect. 4, $l(i, o_1, o_2)$ checks whether a link exists between two given objects. For place $Order \times Item$, above definition requires an *order* to be linked to at least one *item* to be in state $[Order \times Item]$ and vice versa. Based on these state definitions, compound state definitions depending on events and links can be created by combining the definitions for multiple places.

The above mapping demonstrates that our framework can be formally applied to existing approaches. In the following, the intuition for two additional mappings to different approaches is outlined.

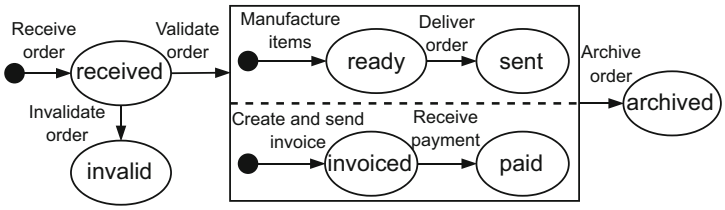


Fig. 5. Excerpt of a Harel statechart as employed by MERODE representing the lifecycle of an order from the example in Sect. 2. We use the activities and events in Fig. 3 as transition labels.

In MERODE [23], object lifecycles are defined by statecharts [10]. The state transitions are annotated with the triggering methods/events, which are further specified using a constraint language inspired by the UML object constraint language [19]. An example lifecycle for an order in the O2C process can be found in Fig. 5. The transition labels refer to the activities and events in Fig. 3. We added state *received* to capture the decision on validity that otherwise depends on an order's attribute values, which are not included in the lifecycle. Verbruggen et al. introduce a means to extract data-aware object-centric event

logs (DOCEL) [9] from MERODE applications [27]. A DOCEL contains object types with attributes, objects with attribute values, and events with timestamps and involved objects. Thereby, they include the required information for system models and system instances as defined in Sect. 4. Based thereon, we can derive the state definitions from the object lifecycles. A lifecycle, defined as a statechart, only includes information on the events triggering state transitions, without considering attributes or links for the state definition. Hence, we can follow a similar pattern to the OPID mapping and define a state to be active if one of its triggering events has occurred that is not eventually followed by an event involving the same object that would leave the state. For example, given a system instance i, state *ready* for an *Order* in Fig. 5 would be defined as:

$$f_{order[ready]}(i,o) := \exists (ts, \text{“}Manufacture\ items'', O_e) \in i.E : o \in O_e \wedge \nexists (ts', \text{“}Deliver\ order'', O'_e) \in i.E : o \in O'_e \wedge ts > ts'$$

In contrast to MERODE, state definitions in PHILharmonicFlows [14] primarily depend on attribute values and object links. A model excerpt for the O2C process is provided in Fig. 6, depicting the order lifecycle (top) and a coordination process (bottom). Note that, since the approach cannot capture concurrent object states, we assume the payment to happen before the delivery to reduce model complexity. In the order's lifecycle, states may depend on attribute values. For example, state *valid* requires $total \geq 0$. The coordination process depicts the interaction between different classes, meaning that, e.g., an invoice must be *sent* before an order can be *invoiced*. We omitted the annotation of semantic relationship types [25] for readability reasons. PHILharmonicFlows models employ a custom data model specifying classes of objects, attributes, and associations, serving as the foundation for the system model. System instances then correspond to the concrete objects, links, and recorded events in a running instance of the model. For each data object state, its respective definition is the conjunction of the attribute requirements specified in the lifecycle (e.g., $total > 0$ for *valid*), the directly preceding nodes in the coordination process (e.g., a linked *Invoice[paid]* before the *Order* can be in state *paid*), and event occurrences.

By comparing the application of our framework to these three approaches, we can identify differences and overlaps in their state definitions. While OPIDs and MERODE both rely on event dependencies and thus have similar state definitions, only PHILharmonicFlows considers attribute values. This showcases that our formal framework is applicable and unifies the notion of data object states across different approaches, thus affirming research question (2).

5.2 Implementation

To strengthen the engineering contribution of the approach, this section reports on a prototypical implementation that allows to model and evaluate data object states in concrete data formats according to the proposed framework.

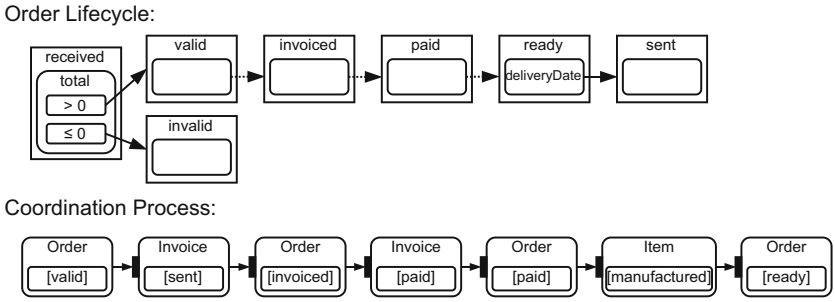


Fig. 6. Excerpt of a PHILharmonicFlows model representing the O2C process. An order lifecycle shows the object-internal behavior, the coordination process depicts dependencies between different classes.

Determining the state of a data object requires its attribute values, links to other objects, and relevant events as defined in Sect. 2. A suitable data structure to capture this information is OCEL 2.0 as introduced in [3]. OCEL logs consist of events, each with an event type, a timestamp, and a set of objects it interacts with, and information about objects, including attribute values and links at a point in time. Hence, an OCEL can provide information on all dimensions required for a system instance as defined in Definition 2, meaning that it serves as an adequate basis for the implementation.

Defining the conditions for each data object state as logical formulae would pose a significant hurdle in the adoption of the approach. Since state definitions resemble decisions repeatedly evaluated for an object in a system, we use a tabular format inspired by DMN decision tables [20]. For each data class, a separate table is created. The columns define the relevant dimensions for the class' state definitions, i.e., attribute values, links, and the event history, while the rows represent the individual state definitions. If all conditions in a row are satisfied, it returns the state specified in its output column. Empty cells indicate unspecified conditions, which are treated as *true*. If a state definition includes a disjunction, multiple rows may be required with the same state as output. We use the DMN concept of a multi-hit policy, *collect*, to support concurrent states by returning the set of all states whose rows evaluated to *true*. An example showing the state definitions presented in Sect. 4 can be found in Fig. 7.

The implementation does not yet support the full extent of the framework, but limits constraints on links and the event history to expressions with a predefined set of operators. For links, these include checks on the number of linked objects (*amount()*) and their states (*amount([state])*), for the event history we currently only support checking the number of event occurrences per event type (*amount([event type])*). The implementation is openly accessible[1]. Even with the limited operator set, it is possible to accidentally create cyclic dependencies

[1] https://github.com/WilliamBrandt/Data-Object-State-Abstraction.

between state definitions, since they can depend on each other. Our implementation detects these dependencies and asks the user to resolve them.

order — Hit policy: Collect

	When <<attribute>> id (string)	And <<attribute>> total (string)	And <<link>> invoice (Any)	And <<link>> item (Any)	And events (Any)	Then state (string)
1	-	-	-	-	-	invalid
2	!= None	> 0	-	-	-	valid
3	-	-	amount() == 1	-	-	invoiced
4	-	-	amount("paid") == 1	-	-	paid
5	-	-	-	amount() >= 1	-	ready
6	-	-	-	-	amount("Deliver order") == 1	sent
7	-	-	-	-	amount("Archive order") == 1	archived

Fig. 7. Decision table defining the states of an order in the O2C process. An order is considered *valid* if its total is greater than zero, and it is considered *paid* if there exists exactly one linked invoice that itself is in state *paid*. Additionally, an order is defined to be *archived* if there is an event of type *Archive order* linked to it.

To illustrate the approach, consider the evaluation of the order object $O1$ as described by the system state in Fig. 2 and Table 1. Applying the table in Fig. 7 reveals that $O1$ is in states *valid*, as its *total* is greater than 0, and *sent*, as event *Deliver order* has occurred (e4). Assuming that invoice $In1$ is in state *paid* since it is linked to a payment, $O1$ is also in state *paid*.

In addition to this simple example, we applied the tool to the *order management* log from the official OCEL website[2] by defining a set of states for the order class and evaluating them on the whole log. Further details can be found in the repository.

6 Discussion and Conclusion

This work analyzes existing process modeling approaches regarding their definition of data object states. We have discussed that different notions of data object states exist. While some approaches have precise definitions of data object states, other works use states in a more abstract manner, leaving the actual state definition implicit. The analysis shows the dimensions that contribute to state definitions of a data object: (i) its attribute values, (ii) its links to other objects, and (iii) the executed activities for the object, while (iv) allowing overlapping state definitions. Based thereon, we propose a unifying framework for data object states. The paper argues that this definition subsumes the existing modeling approaches and allows a unified view on different existing notions of data object states, even for procedural and declarative process models. A prototypical implementation using OCEL and decision tables shows the technical feasibility of modeling our state definitions and applying them to system instances.

[2] https://www.ocel-standard.org/event-logs/overview.

Due to space constraints, we showcase only one mapping of a modeling approach to the unifying state definition formally, while providing an intuition for two others. Future studies need to provide accurate mapping rules for additional approaches. The framework specifically aims to capture the identified dimensions of object states. Hence, the formal state definition operates only on current attribute values of objects. OCEL 2.0 also captures prior value changes, which can be integrated into our approach in the future. Another extension should introduce support for different types of links between the same objects. As discussed for the prototypical implementation, our definition allows for cyclic specifications of data object states. However, we show that they can be detected before their evaluation. The prototypical state modeler utilizes decision tables with a limited set of operators to evaluate constraints on object dependencies and event histories. Hence, it is less powerful than the provided definitions and needs to be extended and, finally, integrated with process modeling and execution tools. Still, the framework itself is not limited to decision tables and can be implemented using other representations of state specifications.

In the future, this paper allows investigating the combination of multiple process modeling approaches by ensuring a consistent notion of data object states. It needs to be considered that the unifying definition of states lifts the assumption of many approaches that one object is in exactly one state at one point in time. This allows for more flexibility, but may affect the execution semantics of the process models. Therefore, future work needs to allow for a formal analysis of states to identify disjoint and overlapping state definitions. In the context of process mining, explicit state definitions can be used to support process discovery. Extending the work in [2] to discover state definitions could reduce the manual effort required to devise state definitions. Furthermore, the compliance of object-centric event logs to state definitions can be checked. Empirical investigations should apply the provided approach to real-world use cases.

In summary, this work provides the formal basis for a unified view on the states of data objects in information systems, allowing to bridge the silos of the existing modeling approaches.

References

1. van der Aalst, W., Berti, A.: Discovering object-centric petri nets. Fundam. Informaticae **175**(1–4), 1–40 (2020)
2. Bano, D., Zerbato, F., Weber, B., Weske, M.: Enhancing discovered process models with data object lifecycles. In: EDOC, pp. 124–133. IEEE (2021). https://doi.org/10.1109/EDOC52215.2021.00023
3. Berti, A., et al.: OCEL (object-centric event log) 2.0 specification. CoRR abs/2403.01975 (2024). https://doi.org/10.48550/ARXIV.2403.01975
4. Calvanese, D., Montali, M., Estañol, M., Teniente, E.: Verifiable UML artifact-centric business process models. In: Li, J., Wang, X.S., Garofalakis, M.N., Soboroff, I., Suel, T., Wang, M. (eds.) ACM CIKM, pp. 1289–1298. ACM (2014). https://doi.org/10.1145/2661829.2662050

5. Christfort, A.K.F., Rivkin, A., Fahland, D., Hildebrandt, T.T., Slaats, T.: Discovery of object-centric declarative models. In: ICPM, pp. 121–128. IEEE (2024). https://doi.org/10.1109/ICPM63005.2024.10680680
6. Fahland, D.: Describing behavior of processes with many-to-many interactions. In: Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings. LNCS, vol. 11522. Springer (2019). https://doi.org/10.1007/978-3-030-21571-2_1
7. Gianola, A., Montali, M., Winkler, S.: Object-centric conformance alignments with synchronization. In: CAiSE 2024. LNCS, vol. 14663, pp. 3–19. Springer (2024)
8. Gómez-López, M.T., Borrego, D., Gasca, R.M.: Data state description for the migration to activity-centric business process model maintaining legacy databases. In: Abramowicz, W., Kokkinaki, A.I. (eds.) BIS 2014. LNBIP, vol. 176, pp. 86–97. Springer (2014). https://doi.org/10.1007/978-3-319-06695-0_8
9. Goossens, A., Smedt, J.D., Vanthienen, J., van der Aalst, W.M.P.: Enhancing data-awareness of object-centric event logs. In: Montali, M., Senderovich, A., Weidlich, M. (eds.) Process Mining Workshops - ICPM 2022 International Workshops, Bozen-Bolzano, Italy, October 23-28, 2022, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 468, pp. 18–30. Springer (2022). https://doi.org/10.1007/978-3-031-27815-0_2
10. Harel, D.: Statecharts: a visual formalism for complex systems. Sci. Comput. Program. **8**(3), 231–274 (1987). https://doi.org/10.1016/0167-6423(87)90035-9
11. van Hee, K.M., Sidorova, N., Voorhoeve, M., van der Werf, J.: Generation of database transactions with petri nets. Fundam. Informaticae **93**(1–3), 171–184 (2009). https://doi.org/10.3233/FI-2009-0095
12. Hewelt, M., Weske, M.: A hybrid approach for flexible case modeling and execution. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNBIP, vol. 260, pp. 38–54. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45468-9_3
13. Hull, R., et al.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: Bravetti, M., Bultan, T. (eds.) Workshop on Web Services and Formal Methods. LNCS, vol. 6551, pp. 1–24. Springer (2010). https://doi.org/10.1007/978-3-642-19589-1_1
14. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. J. Softw. Maintenance Res. Pract. **23**(4), 205–244 (2011). https://doi.org/10.1002/smr.524
15. Meyer, A., Smirnov, S., Weske, M.: Data in business processes. EMISA Forum **31**, 5–31 (2011)
16. Meyer, A., Weske, M.: Extracting data objects and their states from process models. In: Gasevic, D., Hatala, M., Nezhad, H.R.M., Reichert, M. (eds.) EDOC, pp. 27–36. IEEE Computer Society (2013). https://doi.org/10.1109/EDOC.2013.13
17. Montali, M., Rivkin, A.: DB-Nets: on the marriage of colored petri nets and relational databases. Trans. Petri Nets Other Model. Concurr. **12**, 91–118 (2017). https://doi.org/10.1007/978-3-662-55862-1_5
18. OMG: Business Process Model and Notation (BPMN), Version 2.0.2. Tech. rep., Object Management Group (2014). https://www.omg.org/spec/BPMN/2.0.2
19. OMG: Unified Modeling Language (OMG UML), Version 2.5.1. Tech. rep., Object Management Group (2017). https://www.omg.org/spec/UML/2.5.1/PDF
20. OMG: Decision Model and Notation (DMN) Version 1.6 beta. Tech. rep., Object Management Group (2024). https://www.omg.org/spec/DMN/1.6/Beta1
21. Pérez-Álvarez, J.M., Gómez-López, M.T., Eshuis, R., Montali, M., Gasca, R.M.: Verifying the manipulation of data objects according to business process and

data models. Knowl. Inf. Syst. **62**(7), 2653–2683 (2020). https://doi.org/10.1007/s10115-019-01431-5
22. Polyvyanyy, A., van der Werf, J.M.E.M., Overbeek, S., Brouwers, R.: Information systems modeling: language, verification, and tool support. In: Giorgini, P., Weber, B. (eds.) CAiSE 2019. Lecture Notes in Computer Science, vol. 11483, pp. 194–212. Springer (2019). https://doi.org/10.1007/978-3-030-21290-2_13
23. Enterprise Information Systems Engineering. TEES, Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10145-3
24. Snoeck, M., Verbruggen, C., Smedt, J.D., Weerdt, J.D.: Supporting data-aware processes with MERODE. Softw. Syst. Model. **22**(6), 1779–1802 (2023). https://doi.org/10.1007/S10270-023-01095-4
25. Steinau, S., Andrews, K., Reichert, M.: Enacting coordination processes. CoRR abs/2012.08409 (2020). https://doi.org/10.48550/arXiv.2012.08409
26. Steinau, S., Marrella, A., Andrews, K., Leotta, F., Mecella, M., Reichert, M.: DALEC: a framework for the systematic evaluation of data-centric approaches to process management software. Softw. Syst. Model. **18**(4), 2679–2716 (2019). https://doi.org/10.1007/S10270-018-0695-0
27. Verbruggen, C., Goossens, A., Smedt, J.D., Vanthienen, J., Snoeck, M.: iDOCEM: defining a common terminology for object-centric event logging and data-centric process modelling. Softw. Syst. Model. **24**(1), 9–33 (2025). https://doi.org/10.1007/S10270-024-01191-Z
28. Weske, M.: Business process management - concepts, languages, architectures. Fourth Edition. Springer (2024). https://doi.org/10.1007/978-3-662-69518-0

Declarative Process Specifications over Discrete/Continuous Event Data

Carl Corea[1(✉)], Anti Alman[2], Fabrizio Maria Maggi[3], and Paul Hermann Wittlinger[3]

[1] University of Koblenz, Koblenz, Germany
ccorea@uni-koblenz.de
[2] University of Tartu, Tartu, Estonia
anti.alman@ut.ee
[3] Free University of Bozen-Bolzano, Bolzano, Italy
maggi@inf.unibz.it, pwittlinger@unibz.it

Abstract. We investigate declarative process specifications over event data containing both discrete events and *continuous* data streams (e.g., from IoT sensors). For example, it might be desirable to check LTL-like properties not only over event traces but also over sensor data streams, e.g., a CO_2 value not staying over a certain threshold for some period. As existing LTL approaches are based on *discrete* state systems, they are not suitable for this hybrid/continuous setting. Therefore, we propose using Signal Time Logic (STL) for verifying temporal properties over a mix of discrete and continuous behaviors. We show how important topics from conformance checking can be expressed in terms of results from signaling and thus be used to check temporal constraints over discrete/continuous systems. In this context, we also introduce a catalogue of Declare templates over STL, as counterpart to the traditional Declare but for continuous settings.

Keywords: Declarative Process Specification · Signal Time Logic · Conformance Checking · Continuous Data · Hybrid Traces

1 Introduction

Sustained interest in research fields such as Industry 4.0 [19] or the Internet of Things (IoT) [17] has led to an increasing integration of various *sensors* into disciplines such as business process management (BPM) or process mining [1,4,17,25]. As opposed to more traditional settings of BPM systems, where events are assumed to occur at discrete times and in discrete steps, a novel aspect of sensors is that they may exhibit *continuous* behaviors – for example, a continuous stream from a temperature sensor. Especially for Cyber-Physical Systems [27] or Digital Twin settings [13], the integration of such continuous data will—as we believe—play a pivotal role in the research and development of the recently proposed AI-augmented Business Process Management Systems (ABPMSs) [11].

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 277–294, 2025.
https://doi.org/10.1007/978-3-031-94571-7_16

Any ABPMS should be capable of (semi-)autonomously managing the processes it supports, while working within predefined boundaries (referred to as *process frames*). Here, the ABPMS should have mechanisms for detecting and responding to situations where these boundaries would be violated [11]. Especially the latter requires the ABPMS to be aware of, and to reason over, not only the discrete states of the processes it supports, but also the state of the surrounding environment and the relevant artifacts, both of which may be undergoing *continuous* change. For example, given a temperature sensor, an ABPMS should be able to "*detect the gradually increasing temperature in a cooling truck and predict that it will have to undergo maintenance operations within the next weeks*" [11]. Other domain examples could be a continuous signal from a patient monitor, with the need to verify properties such as *"if the oxygen saturation drops below 90%, the doctor should be notified if it remains low for more than 2 min"*.

Clearly, such questions are fundamental and can impact the system behavior as a whole. For example, if the temperature exceeds certain thresholds, other subprocesses may have to be initiated. But verifying such properties over continuous data cannot be handled well with current approaches such as Linear Temporal Logic (LTL), as they are inherently reliant on automata with discrete state systems and are thus not well-suited for capturing the nature of continuous signals. Therefore, in this work, we place continuous data at the forefront by using *Signal Temporal Logic* (STL) [21] for checking temporal properties over continuous data. For example, we envisage to specify rules such as

$\neg \mathbf{F}_{[0,100]}(x > 5)$ *("for the first 100 s, x will not exceed 5")*

i.e., rules that are real-timed and real-valued. We integrate the notion of discrete events by faithfully encoding them as boolean signals, therefore enabling seamless modeling of, and reasoning over, both discrete events and continuous sensor streams. More specifically, we:

- present a **syntax** and **semantics** for a temporal logic on *hybrid* traces, which allows specifying LTL-like constraints over discrete/continuous behaviors; furthermore; introduce a corresponding set of modeling patterns, inspired by the DECLARE modeling language [24], but with STL semantics;
- show how important tasks from **conformance checking** can be expressed with results from the field of signaling. In particular; show how the conformance of a hybrid trace with an STL specification can be checked both in a binary and quantitative way, and;
- report on conformance checking **experiments** on a real-life IoT dataset containing continuous data for CO_2, temperature, humidity, and light intensity.

The rest of the paper is structured as follows. Section 2 recalls crucial preliminary notions. Sections 3 to 5 define, in order, hybrid continuous traces, the corresponding temporal logic together with modeling templates, and the conformance checking task. Section 6 reports on conformance checking experiments. Section 7 highlights related works. And finally, Sect. 8 concludes the paper.

2 Preliminaries

In the following, we assume basic familiarity with LTL and the DECLARE modeling language. We refer the reader to [8] otherwise. As it will be referred to later, we briefly recall that LTL includes temporal operators to specify when certain formulas should hold, in particular, **X** (at the neXt time point), **F** (at some future time point) and **G** (at all following time points). Also, we recall two concrete DECLARE templates that will be referenced in this paper, namely $\text{RESPONSE}(a, b) \equiv \mathbf{G}(a \rightarrow \mathbf{F}b)$ and $\text{CHAINRESPONSE}(a, b) \equiv \mathbf{G}(a \rightarrow \mathbf{X}b)$.

In a general sense, any temporal system is defined over a set of state variables $V = \{V_1, ..., V_n\}$, each ranging over a domain DO_i. Given a linear time domain T, the behavior of the system is then a function that assigns every variable a value at each instant in time. In our setting, the intuition is that we have two different *types* of variables, namely discrete variables (for standard traces), and continuous variables (producing continuous, real-valued signals, such as from sensor data). The aim of this work is then to combine both types of variables to allow reasoning over, what we call, hybrid behaviors.

In our setting, we are interested in modeling behaviors in real-time, e.g., to model real-time behavior of sensor streams. In turn, we consider a finite, continuous time domain T of the form $[0, m] \subset \mathbb{R}$, where 0 is the first instant in time, m is the last instant in time, $m > 0$, and $\forall i \in \{0 \leq i < m\}, t_i < t_{i+1}$. The discrete and continuous state variables then evolve over this continuous time domain together. To better specify the interplay of these variables, we define the notion of *hybrid continuous traces*, which we introduce next.

3 Defining Hybrid Continuous Traces

We define the set of state variables as a pair $V = (D, C)$, where D is a set of discrete variables stemming from traces (via traditional event logs) and C is a set of continuous variables stemming from data streams such as IoT sensor data (see below). We then combine these two variables into, what we call, *hybrid continuous traces*, which we explain in the following (cf. Fig. 1).

As a starting point, we consider a standard event log (1) and fix D as the distinct activities in the log (2). From the log, we extract, for a case of interest, a *timed trace* (3) of the form $tr = \langle (d_1, t_1), ..., (d_n, t_n) \rangle$, where each $d_i \in D$ and each t_i is a point in the time domain T. In other words, (d_i, t_i) represents a timed event that witnesses the occurrence of d_i at time t_i.

For the continuous state variables, we consider as input a set of *data streams*, which are continuous flows of data (think of CO_2 data measured continuously by a sensor). We fix C as the set of streams (4), i.e., each data stream can be thought of to be emitted by a continuous variable. In this sense, we consider every data stream to produce a continuous output over the time domain T (5).[1]

[1] Streams, in an IT context, are usually not truly continuous in a mathematical sense due to hardware limitations or sampling techniques. Yet, we will refer to such streams as "continuous" to distinguish from the traditional LTL setting.

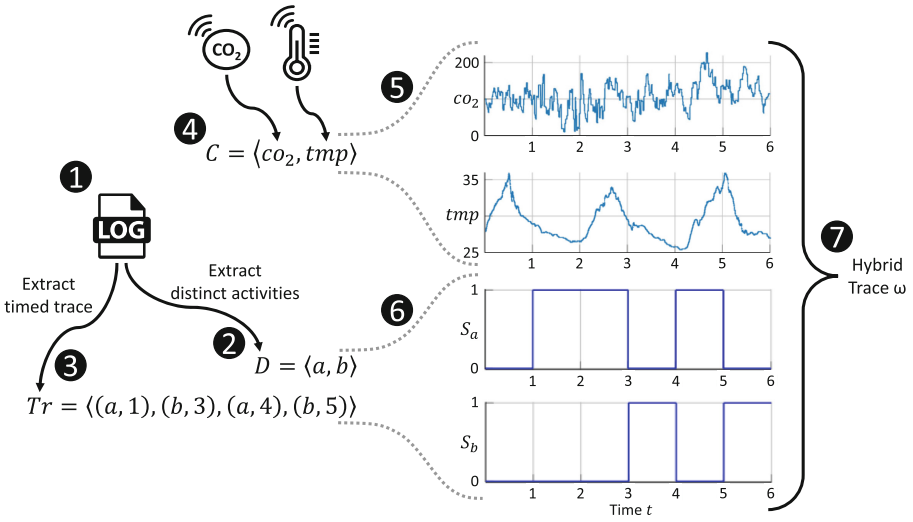


Fig. 1. Construction of hybrid continuous traces.

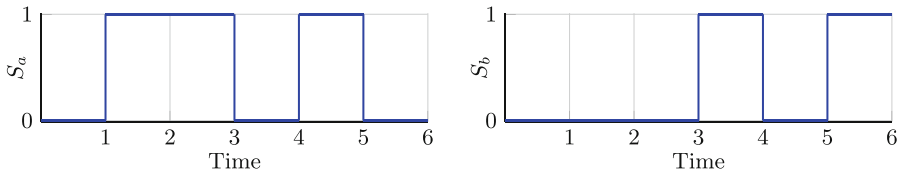
A problem with discrete events is that they lack a continuous representation, making it difficult to evaluate them over continuous time. Here, we propose to construct, for every discrete variable, a *boolean* signal, which is a continuous signal that remains *true* when the corresponding event is the last event that occurred in a trace, and *false* otherwise.

Definition 1 (Augmented boolean signal for discrete variables). *Let D be a set of (discrete) state variables. For a timed trace tr over D, the augmented boolean signal of each variable $d_i \in D$ is defined as $S_{d_i} : T \to \{0, 1\}$, where*

$$S_{d_i}(t) = \begin{cases} 1 & \text{if } d_i \text{ is the most recent event in the trace up to time } t \\ 0 & \text{otherwise} \end{cases}$$

In other words, whenever an event witnessing d_i occurs in the trace, the signal S_{d_i} is set to 1, and all other signals are set to 0. The intuition is that each discrete variable, resp. boolean signal, can be seen as a system *state* that remains active over the time domain T until a different event (or state) is witnessed. This is in line with the DECLARE-assumption, which states that the occurrence of events (or states) is mutually exclusive for any given time point [14].

Example 1. Consider a timed trace $tr = \langle (a, 1), (b, 3), (a, 4), (b, 5) \rangle$ (as shown in Fig. 1 - (3)). Then, the corresponding augmented boolean signals S_a, S_b are:



The proposed transformation approach allows for a uniform representation that aligns the different variable types. Subsequently, as shown in Fig. 1, the augmented boolean signals of the discrete variables (6) are combined with the continuous data streams as a *hybrid continuous trace* (7).

Definition 2 (Hybrid Continuous Trace). *Given a set of state variables $V = (D, C)$, a hybrid continuous trace ω over V is a function $\omega : T \to \mathbb{R}^{|D|+|C|}$.*

That is, a hybrid continuous trace assigns to each variable (discrete or continuous) a value over each time instant of a continuous time domain (so the time-axes of all signals are exactly aligned). Regarding the name, the considered traces are *hybrid*, as they consider discrete and continuous state variables, and *continuous*, as they are viewed over a continuous (real-timed) time domain.[2] For any $v \in C \cup D$ and a time $t \in T$, we denote $v[t]$ as the value of v at time t.

Remark 1. Observe that the discrete events are considered mutually exclusive via our construction mechanism (and so is the "true"-value of their boolean signals), but the data streams over the continuous variables can produce non-zero values simultaneously (e.g., two IoT sensors capturing data simultaneously).

Based on the notion of hybrid continuous traces, we will now present a logic for specifying temporal constraints on such traces.

4 Declarative Process Specifications over Hybrid Data

In this section, we introduce a temporal logic for hybrid continuous traces, which is a variant of Signal Temporal Logic (STL). In general, STL is a logic for defining properties over real-valued signals. In our setting, we lift STL for expressing constraints over a combination of discrete/continuous event data. We then propose an extension of the DECLARE modeling language for hybrid traces, where the constraints can take predicates, time intervals, and real values as parameters.

4.1 A Temporal Logic for Hybrid Continuous Traces Based on STL

Syntax. The proposed signal time logic defines linear-time properties of hybrid continuous traces. For this, let $V = (D, C)$ be a set of state variables over a time domain T (with T of form $[0, m] \subset \mathbb{R}$, and $\forall i \in \{0 \leq i < m\}, t_i < t_{i+1}$). The STL syntax is then defined via the following grammar:

$$\varphi ::= \mathbf{true} \mid c[t]\, op\ k \mid \mathsf{dis}(d[t]) \mid (\neg\varphi) \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1\, \mathbf{U}_I\, \varphi_2) \tag{1}$$

where $c \in C$ is a data stream, $op \in \{<, \leq, =, \geq, >\}$, k is a constant in $\mathbb{Q}$, d is a discrete variable in D, and I is a time interval in T. The term $c[t]\, op\, k$ is used to define conditions over individual data streams, e.g., "c must be larger than 100". The term $\mathsf{dis}(d[t])$ is used to refer to discrete events. The interval I in the $\mathbf{U}$ operator is used to express time bounds and will be explained below.

[2] The term *hybrid* in this context has been coined by the authors in [3] (cf. Section 7).

Remark 2. For the remainder, any value shown in a time interval is meant to be in *seconds*. For any interval $I = [a, b]$ we assume that $0 \leq a < b \leq m$.

W.r.t. the grammar, we can define the usual boolean connectives $(\vee, \rightarrow)$ and further usual abbreviations, in particular $\mathbf{F}$ (Future) with $\mathbf{F}_I\varphi \equiv \mathbf{true}\ \mathbf{U}_I\varphi$, and $\mathbf{G}$ (Globally) with $\mathbf{G}_I \equiv \neg\mathbf{F}_I\neg\varphi$. A remark here is that the $\mathbf{X}$ (neXt) operator from LTL is not suitable for our setting as the time scale is continuous and a "next" point in time is therefore not intuitive. $\mathbf{F}_I$ can be used instead.[3]

We continue with a short illustration of exemplary STL rules.

Example 2. Some exemplary STL rules are shown below (over a discrete variable d and a continuous variable c).

- $\mathbf{G}_{[0,10]}\ (c[t] < 100)$ (For the first 10 s, c is never above 100).
- $\mathbf{G}_{[0,60]}(c[t] > 50 \rightarrow \mathbf{F}_{[0,5]}\mathsf{dis}(d))$ (For the first 60 s, whenever the value of c exceeds 50, somewhere within 5 s thereafter d needs to occur).

We allow to specify intervals of the form $[0, m]$ (using a placeholder m), meaning "until the last instant in time". For intervals of this form, we drop the interval for readability, e.g., $\mathbf{U} \equiv \mathbf{U}_{[0,m]}$.

Example 3. (Continued from Example 2)

- $\mathbf{G}\ (c[t] < 100)$ (c is never above 100 at any point in time).

Semantics. The satisfaction of an STL formula ϕ by a hybrid continuous trace ω is evaluated w.r.t. a time instant $t \in T$, denoted $\omega, t \models \phi$. For any t, satisfaction is inductively defined as follows:

$$
\begin{aligned}
&\omega, t \models \mathbf{true}\\
&\omega, t \models c[t] op\ k \text{ iff } c[t] op\ k\\
&\omega, t \models \mathsf{dis}(d[t]) \text{ iff } d[t] = 1\\
&\omega, t \models \neg\varphi \text{ iff } \omega, t \not\models \varphi\\
&\omega, t \models \varphi_1 \wedge \phi_2 \text{ iff } \omega, t \models \varphi_1 \text{ and } \omega, t \models \varphi_2\\
&\omega, t \models \varphi_1 \mathbf{U}_{[a,b]} \varphi_2 \text{ iff } t + b \leq m \text{ and}\\
&\qquad\qquad \exists t' \in [t + a, t + b] \text{ s.t. } \omega, t' \models \varphi_2 \wedge\\
&\qquad\qquad\quad \forall t'' \in [t, t'] : \omega, t'' \models \varphi_1
\end{aligned}
$$

For continuous variables, the evaluation for any $c[t]$ is simply the truth value of the comparison with a constant k. This is used to verify whether the value of c at t satisfies necessary conditions. For discrete variables, it is checked whether the system is currently in the corresponding state, i.e., whether the discrete variable is set to *true* at t. A further feature is a real-timed $\mathbf{U}$ operator. The evaluation of $\mathbf{U}$ allows to specify a (strong) "until" condition with a given time-frame, in the

[3] At a technical level, "next" could be used to reason on consecutive discrete samples of a continuous data stream. However, this is out of the scope of the paper.

sense that $\varphi_1 \mathbf{U}_I \varphi_2$ needs to hold only within the interval I. By definition, this also applies to $\mathbf{F}_I$ and $\mathbf{G}_I$ – e.g., $\mathbf{F}_I \varphi$ implies φ must occur eventually within I. Regarding the $\mathbf{U}$ operator, note that the shown semantics is meant for cases where a, b of the interval are provided as numerical values. If the shorthand notation [0,m] is used (see above), the semantics is to be understood s.t. t' is anywhere between $[t + a, m]$ (and $t + a < m$).

We say that a hybrid continuous trace ω satisfies a formula ϕ, denoted $\omega \models \phi$, if ϕ is true at time 0. Any ω satisfies a set of formulas Φ if $\omega \models \phi$ for all $\phi \in \Phi$.

It is worth noting that STL formulas may become unsatisfiable if they reference times "beyond" the considered time domain [0,m], i.e., formulas can only be interpreted in a well-defined way if their *future reach* does not exceed m.[4]

Example 4. Consider a hybrid continuous trace ω from [0,100] containing a continuous variable c, where c produces a constant output of 1 over all time points (a straight line). Then consider the formulas $\phi_1 - \phi_3$ with

$$\phi_1 = \mathbf{G}(c[t] < 2) \qquad \phi_2 = \mathbf{G}_{[20,50]}(c[t] < 2) \qquad \phi_3 = \mathbf{G}_{[0,200]}(c[t] < 2)$$

Then we have that $\omega \models \phi_1$, $\omega \models \phi_2$ (α remains < 2 from 0–100 s), but $\omega \not\models \phi_3$ (the interval in ϕ_3 goes beyond the last considered time point of ω).

4.2 A DECLARE Extension for Hybrid Continuous Traces

We now propose an extension of the DECLARE modeling language for hybrid continuous traces. The general idea is—as in DECLARE—to provide a set of high-level templates, but with an underlying STL semantics. For this, some template extensions are necessary. Take for example the DECLARE constraint $\mathsf{Response}(a, b)$, which is designed for an LTL context. Here, a and b are assigned to the Response template, where a is considered as the activation of the constraint and b is considered the reaction. In our STL setting, further information is needed for such a constraint, namely:

a) information on the values of a and b that should hold (as we deal with real-valued inputs)
b) an interval for how soon the response b is expected to occur after a
c) an interval for how long the constraint is expected to hold in general

So the intuition of our DECLARE extension for STL is that templates can receive real-valued *conditions* over the (discrete/continuous) state variables, as well as multiple intervals specifying the real-time aspects.

Definition 3 (DECLARE specification over hybrid continuous traces). *A* DECLARE *process specification over a hybrid continuous trace is a tuple* $M = (\boldsymbol{P}, \boldsymbol{V}, \boldsymbol{C})$*, where*

[4] The future reach $f(\phi)$ of an STL formula ϕ is inductively defined via $f(\mathbf{true}) = 0$, $f(e_i[t] op\, k) = 0$, $f(\neg\phi) = f(\phi)$, $f(\phi_1 \wedge \phi_2) = max(f(\phi_1), f(\phi_2))$, and $f(\phi_1 \mathbf{U}_I \phi_2) = sup(I) + max(f(\phi_1), f(\phi_2))$.

- $\boldsymbol{P}$ *is a finite set of predefined templates (i.e., predicates) of the form* $P_{i_1,...,i_n}(x_1,...,x_m)$*—over intervals* $i_1,...,i_n$ *and variables* $x_1,...,x_m$*.*
- $\boldsymbol{V} = (D, C)$ *is a finite set of discrete and continuous state variables. Furthermore, define* $\Theta_{\boldsymbol{V}}$ *as a set of terms over* $\boldsymbol{V}$*, where every term* Θ_i *is of the form* $\Theta_i ::= c[t]\, op\ k \mid \mathsf{dis}(d[t])$*.*
- $\boldsymbol{C}$ *is a finite set of constraints in form of tuples* $(P_{i_1,...,i_n}(x_1,...,x_m), \tau, \alpha)$*, where* $P_{i_1,...,i_n}(x_1,...,x_m)$ *is a template from* $\boldsymbol{P}$*;* τ *is a mapping assigning every interval* i_j *a time interval* $[a, b] \subset \mathbb{R}$ *(with* $a, b > 0$*); and* α *is a mapping that assigns every variable* x_i *with a term* Θ_i *in* $\Theta_{\boldsymbol{V}}$*. We compactly denote such a constraint with* $P_{i_1,...,i_n}(\Theta_1,...,\Theta_m)$*.*

In other words, a constraint is constructed by instantiating templates with elements in $\Theta_{\mathbf{V}}$ (which are conditions over the discrete/continuous variables), as well as time intervals for activations and reactions of the constraints (cf. [8,28]).

Based on prominent DECLARE templates from the literature [8], we propose an inital catalogue of templates, shown in Table 1.

Table 1. Modeling templates for hybrid event data.

Template	STL Semantics
$\textsc{Existence}_I(\Theta)$	$\mathbf{F}_I(\Theta)$
$\textsc{PeriodExistence}_{I_a,I_r}(\Theta)$	$\mathbf{F}_{I_a}(\mathbf{G}_{I_r}\Theta)$
$\textsc{Response}_{I_a,I_r}(\Theta_1, \Theta_2)$	$\mathbf{G}_{I_a}(\Theta_1 \rightarrow \mathbf{F}_{I_r}(\Theta_2))$
$\textsc{RespondedExistence}_{I_a,I_r}(\Theta_1, \Theta_2)$	$\mathbf{F}_{I_a}(\Theta_1) \rightarrow \mathbf{F}_{I_r}(\Theta_2)$
$\textsc{CoExistence}_I(\Theta_1, \Theta_2)$	$\mathbf{F}_I(\Theta_1) \leftrightarrow \mathbf{F}_I(\Theta_2)$
$\textsc{Choice}_I(\Theta_1, \Theta_2)$	$\mathbf{F}_I(\Theta_1) \vee \mathbf{F}_I(\Theta_2)$
$\textsc{Absence}_I(\Theta)$	$\neg\mathbf{F}_I(\Theta)$
$\textsc{NotPeriodExistence}_{I_a,I_r}(\Theta)$	$\neg\mathbf{F}_{I_a}(\mathbf{G}_{I_r}\Theta)$
$\textsc{NotResponse}_{I_a,I_r}(\Theta_1, \Theta_1)$	$\mathbf{G}_{I_a}(\Theta_1 \rightarrow \neg\mathbf{F}_{I_r}(\Theta_2))$
$\textsc{NotCoExistence}_I(\Theta_1, \Theta_2)$	$\neg(\mathbf{F}_I(\Theta_1) \wedge \mathbf{F}_I(\Theta_2))$

Example 5. Some exemplary constraints based on Table 1 are shown below (over discrete event d and data streams a, b).

- $\textsc{Existence}_{[0,10]}(a[t] > 100)$ (within the first 10 s, a will go over 100).
- $\textsc{PeriodExistence}_{[0,m][0,10]}(a[t] > 100)$ (there is eventually a period where a remains over 100 for 10 s).
- $\textsc{Response}_{[0,m][0,5]}(\mathsf{dis}(d[t]), b[t] < 50)$ (whenever d occurs, b goes under 50 within 5 s thereafter).
- $\textsc{Absence}_{[0,m]}(a[t] > 100)$ (a never goes over 100).
- $\textsc{NotPeriodExistence}_{[0,m][0,10]}(a[t] > 100)$ (a never remains over 100 for stretches of 10 s (or more)).

- $\text{NotCoExistence}_{[0,m]}(a[t] > 100, b[t] > 100)$ (a and b never go over 100 together).

Remark 3. Based on the STL syntax, it is immediate that many templates based on the **X** operator cannot be plausibly lifted to STL. For example, it is not straightforward how to implement a ChainResponse constraint, which for LTL is well-defined. For our setting, a suitable interval has to be chosen, and the change between ChainResponse and Response is more gradual.

An important remark is that arbitrary combinations of discrete variables and data streams can be applied as parameters, exemplified in Example 6.

Example 6. (over discrete event d, e and streams a, b).

- $\text{Response}_{[0,m],[0,5]}(\mathsf{dis}(d[t]), \mathsf{dis}(e[t]))$ (whenever d occurs, e occurs within 5 s thereafter).
- $\text{Response}_{[0,m],[0,5]}(a[t] < 50, \mathsf{dis}(d[t]))$ (whenever a goes under 50, d occurs within 5 s thereafter).
- $\text{Response}_{[0,m],[0,5]}(a[t] < 50, b[t] > 100)$ (whenever a goes under 50, b goes over 100 within 5 s thereafter).

The presented catalogue of templates allows to define many forms of constraints, with the complexity of STL hidden from the user. For example, recalling the hospital example from Sect. 1, the constraint of interest there can be expressed as follows (over the oxygen saturation ox):

- $\text{NotPeriodExistence}_{[0,m][0,120]}(ox[t] < 90)$ (oxygen saturation should never be below 90 for more than 2 min).

We comment on possible extensions to the catalogue in Sect. 8.

Based on the introduced specification techniques, we now show how to verify whether a hybrid continuous trace satisfies such a specification.

5 Conformance Checking over Hybrid Continuous Traces

In this section, we focus on checking whether a given hybrid continuous trace satisfies an STL specification. A direct question for conformance checking is verifying whether/when a hybrid continuous trace satisfies an STL formula ϕ. This is referred to as a *monitor*. For reference, we define the following problem:

(ϕ-Conformance) *Input:* Hybrid trace ω, STL formula ϕ *Output:* $\omega \models \phi$?

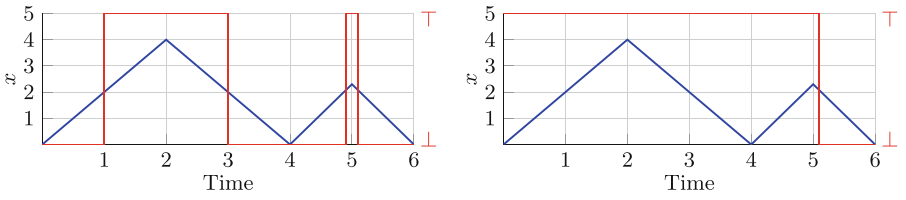
A difficulty in our "continuous" setting is that standard automata-based techniques are not suitable to determine *ϕ-conformance*, as they rely on discrete state representations (and the signals may be too granular for being represented with such techniques). However, results from signaling can be used to define monitors via *secondary boolean signals*. The intuition is that an auxiliary boolean signal

$\mathbb{B}(\phi, \omega, t)$ can be defined "on top of" the raw trace signal (in a dependent manner), where the boolean signal transduces the signal of the continuous trace at t into $\{\top, \bot\}$ (true (1), false (0)), depending on whether $\omega, t \models \phi$ is true. In this way, the boolean signal is a direct monitor for the continuous trace signal. For a hybrid continuous trace ω, this auxiliary signal w.r.t. t can be defined via the following satisfaction function $\mathbb{B}$.[5]

Definition 4 (Boolean satisfaction function). *Let a hybrid continuous trace ω over a set of state variables $V = (D, C)$ as before, a set of operators (as before) and a constant $k \in \mathbb{Q}$. Then, the boolean satisfaction function $\mathbb{B}$ of an STL formula ϕ w.r.t. ω, t is inductively defined as*

- $\mathbb{B}(\textbf{\textit{true}}, \omega, t) = \top$
- $\mathbb{B}(c[t]\ op\ k, \omega, t) = (c[t]\ op\ k)$
- $\mathbb{B}(\textit{dis}(d[t]), \omega, t) = (d[t] = 1)$
- $\mathbb{B}(\neg\phi, \omega, t) = -(\mathbb{B}(\phi, \omega, t))$
- $\mathbb{B}(\phi_1 \wedge \phi_2, \omega, t) = \min(\mathbb{B}(\phi_1, \omega, t), \mathbb{B}(\phi_2, \omega, t))$
- $\mathbb{B}(\phi_1\, \boldsymbol{U}_I \phi_2, \omega, t) = \max_{t' \in t+I}(\min(\mathbb{B}(\phi_2, \omega, t'), \min_{t'' \in [t,t']}(\mathbb{B}(\phi_1, \omega, t''))))$

In other words, the boolean signal is such that $\omega, t \models \phi \leftrightarrow \mathbb{B}(\phi, \omega, t) \models \top$. Fig. 2 shows an exemplary trace ω over a signal x, and two boolean monitors for the formulas $x[t] \geq 2$, and $\mathbf{F}x[t] \geq 2$. Here, the (red) boolean satisfaction signal is a dependent signal of x (over the same time), where the boolean signal oscillates between $\bot$ and $\top$, corresponding to those points in time t where ω satisfies the respective formulas. A boolean satisfaction signal for a *set* of STL constraints can be obtained by checking their conjunction over ω.



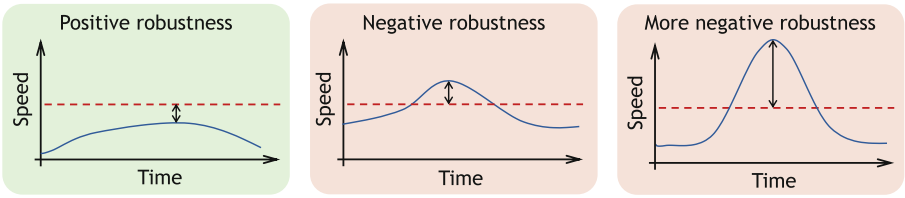
(a) Boolean monitor for $x[t] \geq 2$ (red). (b) Boolean monitor for $\mathbf{F}x[t] \geq 2$ (red).

Fig. 2. Two examples: The same signal over x (blue) and two corresponding boolean monitors (red) for two different STL formulas. (Color figure online)

The boolean satisfaction signals directly yields a monitor for conformance checking (cf. e.g. [21,22]). However, in our setting, such a binary conformance assessment may be too coarse, explained as follows: In our setting, we are dealing with data streams, which may be naturally prone to noise. In this sense, one

[5] For this, for every time point t in T, let $t + I$ denote the set $\{t + t' \mid t' \in I\}$.

may need to account for this noise to obtain a more *meaningful* assessment of satisfaction. Consider for example Fig. 3, which shows three continuous signals over a data stream of an accelerometer (speed value). Assume a simple STL rule ϕ_{speed}, stating that the speed must never go over 100 (dashed line). Then we have that: (a) clearly satisfies ϕ_{speed}, but (b) went over the threshold. However, this deviation may be due to noise of the sensor data and still within a threshold, so (b) may still satisfy ϕ_{speed} "to some degree". For example, it seems intuitive that (c) violates ϕ_{speed} "more" than (b). Thus, the given noisy setting may require a more quantitative approach to determine satisfaction, which we introduce in the following. This is also referred to as *robust* monitoring, i.e., a satisfaction degree.



(a) Trace fully satisfying ϕ_{speed}.

(b) Trace "slightly" violating ϕ_{speed}.

(c) Trace violating ϕ_{speed} "more" than trace (b).

Fig. 3. Three exemplary traces (a)-(c) and their (quantitative) conformance to the rule $speed[t] < 100$. (Images taken from [23]).

A quantitative semantics can be defined as an extension to the boolean satisfaction signal. The idea is that again we define an auxiliary signal dependent on the raw data, but this time the signal value is not boolean satisfaction, but the *distance* (Δ) between the given and the expected value at every instant t.

Definition 5 (Quantitative satisfaction function). *Let a hybrid continuous trace ω over a set of state variables $V = (D, C)$ as before, a set of operators $op = \{<, \leq, =, \geq, >\}$ and a constant $k \in \mathbb{Q}$. Then, the quantitative satisfaction function Δ of an STL formula ϕ w.r.t. ω, t is inductively defined as*

- $\Delta(\textbf{\textit{true}}, \omega, t) = \infty$
- $\Delta(c[t]\ op\ k, \omega, t) = \begin{cases} k - c[t] & \text{if } op = \text{“}<\text{”} | \text{“}\leq\text{”} \\ -(|k - c[t]|) & \text{if } op = \text{“}=\text{”} \\ c[t] - k & \text{if } op = \text{“}\geq\text{”} | \text{“}>\text{”} \end{cases}$
- $\Delta(\textit{dis}(d[t]), \omega, t) = \begin{cases} \infty & \text{if } d[t] = 1 \\ -\infty & \text{otherwise} \end{cases}$
- $\Delta(\neg\phi, \omega, t) = -(\Delta(\phi, \omega, t))$
- $\Delta(\phi_1 \wedge \phi_2, \omega, t) = \min(\Delta(\phi_1, \omega, t), \Delta(\phi_2, \omega, t))$
- $\Delta(\phi_1 \, \boldsymbol{U}_{[a,b]}, \omega, t) = \max_{t' \in [t+a, t+b]} (\min(\Delta(\phi_2, \omega, t'), \min_{t'' \in [0, t']} \Delta(\phi_1, \omega, t'')))$

In words, the quantitative satisfaction measures the distance between the expected and actual value at every $t \in T$, where a distance Δ indicates that ω satisfies/violates ϕ at t with degree Δ. An example is shown in Fig. 4, for a signal x and an exemplary constraint $\mathbf{G}_{[0,2]}(x[t] < 3)$.

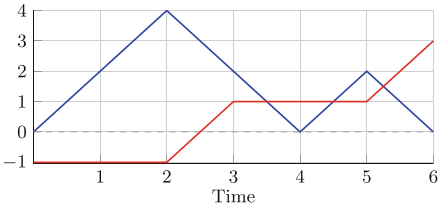


Fig. 4. Signal x (blue) and quant. monitor (red) for the formula $\mathbf{G}_{[0,2]}(x[t] < 3)$. (Color figure online)

Remark 4. It is immediate that the sign of the quantitative semantics coincides with boolean satisfaction (referred to as soundness [22]), in such that for any ϕ, ω, t: $\Delta(\phi, \omega, t) > 0 \rightarrow \omega, t \models \phi$ (and vice-versa).

As we obtain quantitative semantics for every point t, various aggregations can be defined on top of the individual ones to obtain a "global" measure of conformance, for example, the *average* satisfaction degree, or a *boolean* value depending on whether Δ was larger than 0 for all $t \in T$.

We are now able to check the conformance of hybrid continuous traces with STL specifications; and in fact, *ϕ-conformance* can be obtained via the quantitative satisfaction semantics. Beyond conformance checking, a further advantage is that the shape of the satisfaction signal can also be used for visual drift detection [31], i.e., visually identifying gradual drifts or recurring patterns in constraint satisfaction. An important remark is that the considered semantics focus on *space* robustness, i.e., value deviations at time points (so to say, a "vertical" perspective). The semantics do not assess by how much some signal should be shifted "horizontally", referred to as *time* robustness (cf. [5]).

To demonstrate the introduced techniques, we continue with experiments.

6 Experiments

We implemented our approach using the stlcg library [20] and performed experiments on a real-life dataset. Our implementation takes as input a hybrid continuous trace and constraints specified with "DECLARE" as in Table 1, and then, transforms these constraints to STL and computes the *quantitative* conformance of the trace w.r.t. the constraints. All code can be found online[6].

[6] https://github.com/pwittlinger/STL-Experiments.

We consider the data-set by Wibisono et al. [29], who collected data with real IoT sensors. The data-set contains readings from different sensors for temperature (°C), CO_2 value (ppm), humidity (%), and light sensitivity (lux). The data was captured in one-second intervals for a three-month period, yielding approx. 6M data points. After removing outliers, null values and filtering, around 2M samples remained. As the data-set contains no discrete event data, one additional boolean column "breakdown" was added, indicating if the time between two consecutive measurements exceeds a given threshold ($\delta_t > 1000$). The data was further enhanced with the gradient of the CO_2 emissions, allowing us to reason over rapid measurement changes (possibly indicative of a breakdown).

For our experiments, we considered 6 exemplary constraints and measured the time for checking the conformance of the data with these constraints. As specific constraints, we chose 3 of the type Absence and 3 of the type Response (cf. Fig. 5), where the complexity of the constraints was increased incrementally by including additional data streams or nesting additional constraints. As parameters for the experiments, a constraint and a number of observations n_{Obs} (from the data) was provided. As settings, we configured the number of observations n_{Obs} as $[60, 60^2, 60^2 * 24, 60^2 * 24^2, 60^2 * 24^3, 60^2 * 24^5, 60^2 * 24^7, 60^2 * 24^{10}, 10^6]$ (which correspond to 1 min, 1 h, 1 Day, ..., up to 1M observations, as the sampling rate is 1 s). For each setting, we measured the run-time 5 times (repeated 3x), producing 15 measurements per setting. The experiments were conducted in Python 3.11 on a machine with 16GB RAM and an AMD Ryzen 4800U CPU. Note that the stlcg library is based on PyTorch tensors, which includes automatic differentiation of tensors. This option has been turned off.

Figure 5 shows the average run-time (in s) of the 6 constraints. Noticeably, all constraints exhibit a linear trend in regards to the number of observations. The last two Absence constraints (● +●) have a runtime approximately twice of first Absence constraint (●), explained by the additional data-stream. There was no significant difference in runtime between these constraints (p-value of 0.8068 using a two-sample Kolmogorov-Smirnov test). In fact, Leung et al. [20] show that the run-time scales linearly with formula size. Similarly, the first Response constraint (●) requires approximately half the time of the second constraint (●). This could stem from including the discrete signal, the combination of two input

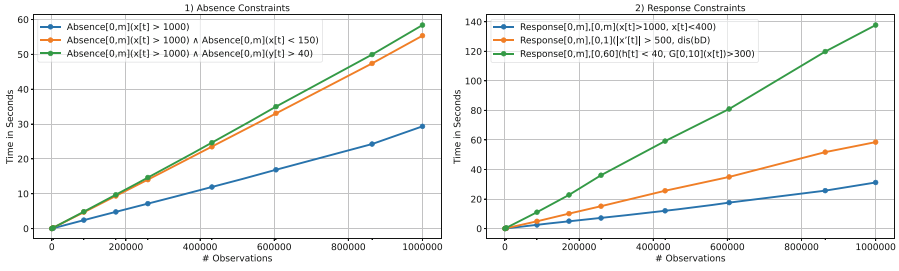


Fig. 5. Runtime performance over various Constraints.

streams, or from the change in intervals. The same applies to the third RESPONSE constraint (●), however, the nested structure likely contributes to the runtime.

Overall, we deem conformance checking over hybrid continuous traces feasible, given that we could monitor up to 1M observations on consumer-grade hardware in a reasonable time-frame. The full details on each experiment can be found in the code repository.

7 Related Work

This work is related to *IoT-aware BPM* [4,6,17]. Given the rising ubiquity of sensors in cyber-physical systems—e.g., smart factories—, there is consensus that integrating sensor data into BPM will be an important step in improving control and analysis of processes [6,17]. So far, most research has been on the integration of sensor concepts into already existing (imperative) modelling languages [4]. In this work, we instead focus on declarative languages, adopting, from the outset, a formalism explicitly designed for continuous data. Leveraging sensor data for new forms of conformance checking, or combining discrete and continuous data, has been called for in [17,26], and is addressed in this work. A novel contribution here is that data is not discretized in our work (cf. e.g. [18]), which allows for new forms of analysis such as checking constraints over IoT data directly.

From a technical perspective, this paper is related to works on declarative process specification considering *time*. Different versions of declarative specifications over timed traces have been proposed in [2,5,28,32], mostly by means of metric temporal logic over finite traces (MTL_f), which is a timed temporal logic over boolean predicates. In this work, we extend timed logic with real-valued predicates, which is needed for reasoning on the real-valued signals from sensors. For example, we can faithfully include the data-domain, other than, e.g., in variants such as MP-DECLARE, where data is a payload of an activity, meaning that data cannot be updated or accessed independently of the activity perspective. Should, e.g., sensor readings be incorporated into a MP-DECLARE model, an additional activity capturing the *sensor measurements* has to be added to the model. The frequency of this artificial activity would be equal to the sample rate of the sensor, which would substantially increase the trace length. Furthermore, even in this case, the data conditions in MP-DECLARE are not designed for continuous data, making it impossible to, for example, represent gradients.

It is also worth noting that none of the above works consider a continuous time domain, i.e., the number of states for formula interpretations in those works is countable. In this work, we propose means for reasoning on formulas over a continuous space in form of satisfaction signals. A related notion of quantitative satisfaction has been proposed in a fuzzy LTL-setting in [9], though for a different formalism and not considering the notion of timed traces.

The temporal logic and novel DECLARE templates considered in this work are based on STL [21]. In this work, we integrate discrete events by encoding them as boolean signals, which enables a uniform representation and comparative analysis across the different variable types. An initial work in this direction is the STL

MX dialect in [12], however that work provides only a logical description of the language, focuses on digital clocks in order to express both dense- and discrete-time properties and allows for the separation of the two time domains by utilizing both a LTL and a STL monitor. A further related work is [3], who extend LTL by incorporating first-order quantifiers, enabling model-checking of systems that combine discrete and continuous dynamics. While [3] offers greater expressiveness, STL as applied in this work is better suited for checking the conformance of real-time signal due to its direct focus on robust satisfaction measures, which are essential for the considered settings where quantitative reasoning and tolerance to perturbations are critical [10]. The idea of defining DECLARE-like constraints over STL (STL MX* dialect) was proposed in the masters thesis by one of the co-authors [30].

8 Conclusion

In this work, we have proposed a framework for checking specifications over discrete-/continuous data using STL, enabling quantitative monitoring of hybrid processes. This approach provides a foundation for improved process analysis in IoT and CPS, supporting monitoring over real-valued and real-timed data streams (and their relationships to discrete events). Furthermore, any DECLARE specification created using the standard DECLARE templates analogous to the ones proposed in this paper (cf. Table 1) represents a subset of the behaviors handled by our approach. For instance, $\textsc{Existence}_I(\Theta)$ used on a discrete variable (i.e., the boolean signal of an activity) over the entire duration of the trace will be equivalent to the EXISTENCE template of standard DECLARE.

However, our approach is not suitable to capture durative actions or consecutive occurrences of the same activity. The latter is a consequence of defining the boolean signals for discrete variables such that a signal remains true from the occurrence of an event until the occurrence of some different event. A potential solution is setting all boolean signals to false in the immediate next time instant after the occurrence of an event. However, this would complicate the formal semantics, as earlier time instants of all signals would need to be checked to determine which of them was the last to be true. This, along with durative actions, will be addressed in future works.

In addition, encoding discrete events as boolean signals in a continuous time domain seems promising for representing events that are not mutually exclusive and *truly* in parallel. Handling such events would also require more complex formal semantics and adjustment of the signals defined in this paper. This would enable the addition of further types of constraints based on interval logic, such as DURING (as e.g. in the HANFORPL [15] modelling language).

In the context of data conditions, we aim to add expressions such as arithmetics. This could be used for expressing constraints such as: $\textsc{Absence}(x[t] > y[t])$ or $\textsc{Absence}((x[t] - y[t]) > 50)$. We also aim to investigate the relationship to hybrid automata [16] (in short: FSAs with discrete and continuous dynamics), which could allow to lift further BPM tasks—such as checking STL specifications for consistency [7]—to hybrid systems.

Acknowledgments. The work of A. Alman was supported by the Estonian Research Council grant PRG1226. The work of F. M. Maggi was partially funded by the NextGenerationEU FAIR PE0000013 project MAIPM (CUP C63C22000770006).

References

1. Bazan, P., Estevez, E.: Industry 4.0 and business process management: state of the art and new challenges. Bus. Process. Manag. J. **28**(1), 62–80 (2022)
2. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Syst. Appl. **65**, 194–211 (2016)
3. Cimatti, A., Roveri, M., Tonetta, S.: Requirements validation for hybrid systems. In: International Conference on Computer Aided Verification, pp. 188–203. Springer (2009)
4. Compagnucci, I., Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F.: A systematic literature review on IoT-aware business process modeling views, requirements and notations. Softw. Syst. Model. **22**(3), 969–1004 (2023)
5. De Giacomo, G., Murano, A., Patrizi, F., Perelli, G., et al.: Timed trace alignment with metric temporal logic over finite traces. In: 18th International Conference on Principles of Knowledge Representation and Reasoning, pp. 227–236 (2021)
6. De Luzi, F., Leotta, F., Marrella, A., Mecella, M.: On the interplay between business process management and Internet-of-Things: a systematic literature review. Business & Information Systems Engineering, pp. 1–24 (2024)
7. Di Ciccio, C., Maggi, F.M., Montali, M., Mendling, J.: Resolving inconsistencies and redundancies in declarative process models. Inf. Syst. **64**, 425–446 (2017)
8. Di Ciccio, C., Montali, M.: Declarative process specifications: reasoning, discovery, monitoring. Process mining handbook **448**, 108–152 (2022)
9. Donadello, I., Felli, P., Innes, C., Maggi, F.M., Montali, M.: Conformance checking of fuzzy logs against declarative temporal specifications. In: International Conference on Business Process Management, pp. 39–56. Springer (2024)
10. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: International Conference on Formal Modeling and Analysis of Timed Systems, pp. 92–106. Springer (2010)
11. Dumas, M., et al.: AI-augmented business process management systems: a research manifesto. ACM Trans. Manag. Inf. Syst. **14**(1), 11:1–11:19 (2023)
12. Ferrère, T., Maler, O., Ničković, D.: Mixed-time signal temporal logic. In: André, É., Stoelinga, M. (eds.) FORMATS 2019. LNCS, vol. 11750, pp. 59–75. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29662-9_4
13. Fornari, F., et al.: Digital twins of business processes: A research manifesto (2024). https://arxiv.org/abs/2410.08219
14. Giacomo, G.D., Masellis, R.D., Montali, M.: Reasoning on LTL on finite traces: insensitivity to infiniteness. In: AAAI, pp. 1027–1033. AAAI Press (2014)
15. Henkel, E., Hauff, N., Langenfeld, V., Eber, L., Podelski, A.: Systematic adaptation and investigation of the understandability of a formal pattern language. Requirements Eng. **29**(1), 3–23 (2024)
16. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science, pp. 278–292. IEEE (1996)

17. Janiesch, C., et al.: The Internet of Things meets business process management: a manifesto. IEEE Syst. Man Cybern. Mag. **6**(4), 34–44 (2020)
18. Janssen, D., Mannhardt, F., Koschmider, A., van Zelst, S.J.: Process model discovery from sensor event data. In: ICPM Workshops. Lecture Notes in Business Information Processing, vol. 406, pp. 69–81. Springer (2020)
19. Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., Hoffmann, M.: Industry 4.0. Bus. Inf. Syst. Eng. **6**(4), 239–242 (2014). https://doi.org/10.1007/s12599-014-0334-4
20. Leung, K., Aréchiga, N., Pavone, M.: Back-propagation through signal temporal logic specifications: infusing logical structure into gradient-based methods. In: Workshop on Algorithmic Foundations of Robotics (2020)
21. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS/FTRTFT - 2004. LNCS, vol. 3253, pp. 152–166. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30206-3_12
22. Maler, O., Nickovic, D., Pnueli, A.: Checking temporal properties of discrete, timed and continuous behaviors. Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday, pp. 475–505 (2008)
23. Pavone, M., Leung, K.: Transforming specifications into robot programs: A survey of formal methods tools for non-experts (2021). https://faculty.washington.edu/kymleung/assets/pdf/Pavone.Leung.FMR.Workshop.IROS21.pdf. Accessed 18 Nov 2024
24. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: full support for loosely-structured processes. In: EDOC, pp. 287–300. IEEE Computer Society (2007)
25. Seiger, R., Huber, S., Heisig, P., Assmann, U.: Enabling Self-adaptive Workflows for Cyber-physical Systems. In: Schmidt, R., Guédria, W., Bider, I., Guerreiro, S. (eds.) BPMDS/EMMSAD - 2016. LNBIP, vol. 248, pp. 3–17. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39429-9_1
26. Seiger, R., Huber, S., Heisig, P., Aßmann, U.: Toward a framework for self-adaptive workflows in cyber-physical systems. Softw. Syst. Model. **18**, 1117–1134 (2019)
27. Song, H.H., Rawat, D.B., Jeschke, S., Brecher, C.: Cyber-physical systems: foundations, principles and applications. Morgan Kaufmann (2016)
28. Westergaard, M., Maggi, F.M.: Looking into the future: using timed automata to provide a priori advice about timed declarative process models. In: OTM Confederated International Conferences on the Move to Meaningful Internet Systems, pp. 250–267. Springer (2012)
29. Wibisono, A., et al.: Dataset of short-term prediction of CO2 concentration based on a wireless sensor network. Data Brief **31**, 105924 (2020)
30. Wittlinger, P.H.: Enriching declare with dense time and real-valued signals: from formalisms to reasoning tasks (2024). https://findit.dtu.dk/en/catalog/65ea65510d5c4117b6ea88ec. Accessed 18 Nov 2024

31. Yeshchenko, A., Di Ciccio, C., Mendling, J., Polyvyanyy, A.: Visual drift detection for event sequence data of business processes. IEEE Trans. Visual Comput. Graph. **28**(8), 3050–3068 (2021)
32. Zaki, N.M., Helal, I., Hassanein, E.E., Awad, A.: Validating temporal compliance patterns: A unified approach with MTL_f over various data models. arXiv preprint arXiv:2406.08530 (2024)

Ontologies and Knowledge Graphs

Restructuring Knowledge Graphs with Conceptual Models: Implications for Machine Learning Predictions in Drug Repurposing

César Bernabé[1(✉)], Rosa Zwart[1], Pablo Perdomo-Quinteiro[2], Annika Jacobsen[1], Tiago Prince Sales[3], Núria Queralt-Rosinach[1], Katherine Wolstencroft[4,5], Luiz Olavo Bonino da Silva Santos[1,3], Barend Mons[1,6], and Marco Roos[1]

[1] Leiden University Medical Center, Leiden, The Netherlands
{c.h.bernabe,a.jacobsen,n.queralt-rosinach,b.mons,m.roos}@lumc.nl, l.o.boninodasilvasantos@utwente.nl, barend@gofair.foundation
[2] Grupo de Aplicación de Telecomunicaciones Visuales, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Madrid, Spain
p.perdomo@upm.es
[3] Semantics, Services and Cybersecurity group, University of Twente, Enschede, The Netherlands
t.princesales@utwente.nl
[4] Life Science Semantics Group, Leiden Institute of Advanced Computer Science, Leiden, The Netherlands
k.j.wolstencroft@amsterdamumc.nl
[5] Advanced Compute and Data Core, Amsterdam University Medical Centre, Amsterdam, The Netherlands
[6] GO FAIR Office, Leiden, The Netherlands

Abstract. This paper investigates the impact of restructuring knowledge graphs (KGs) with well-founded conceptual models to improve machine learning (ML) predictions, particularly in drug repurposing applications. These conceptual models were developed using OntoUML, which is grounded in the Unified Foundational Ontology, and were constructed following an established workflow for data FAIRification–a process aimed at making data more Findable, Accessible, Interoperable, and Reusable. We compared the performance of a Graph Neural Network model trained on original public KGs with models trained on the same restructured KGs. Our results indicate that while the ML model classification performance (measured in terms of accuracy and error metrics) remains similar for both, the models trained on restructured KGs produce more consistent predictions, reducing variability across multiple runs. These findings suggest that restructuring KGs using well-founded conceptual models can enhance the reliability of ML predictions without compromising model performance. We conclude by proposing future research directions to further explore the potential of conceptual models and FAIR principles in improving ML.

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 297–313, 2025.
https://doi.org/10.1007/978-3-031-94571-7_17

Keywords: Conceptual Model · Machine Learning · FAIR Principles · Knowledge Graphs

1 Introduction

Machine learning (ML) models are often trained using large knowledge bases [6]. However, constructing such voluminous datasets is both resource-intensive and time-consuming, as existing data is typically not prepared for reuse [15]. To facilitate data reusability, the FAIR principles were introduced to guide the process of making data and other resources Findable, Accessible, Interoperable, and Reusable [31]. Since their publication in 2016, the principles have gained significant traction across various fields [17]. Similarly, research and applications involving ML models have expanded rapidly in recent years [6]. However, although these areas complement each other, there has been little research on the specific impact of FAIR data on ML methods.

Jacobsen *et al.* [16] proposed a stepwise process of making existing data FAIR (referred to as FAIRification). The generic FAIRification workflow is organised in steps, starting with the identification of FAIRification objectives, followed by the analysis of (meta)data, the design of semantic models for (meta)data, and (meta)data restructuring, linkage, hosting, and assessment. In the semantic modelling phase, a conceptual model [10] of the data elements (e.g. patient, disease) and relationships (e.g. drug treats disease) is constructed. Since FAIR promotes reuse by both humans and machines, the conceptual model is designed to be as accurate a reflection of the data domain as possible [17]. In the data restructuring step, the data to be made FAIR is reorganised to align with the structure of the conceptual model, making it not only more understandable for humans but also more easily integrated with other FAIR data.

In this work, we aim to assess the impact of this conceptual model-based data restructuring on ML models. To conduct this experiment, we build on parts of the pipeline developed by Perdomo-Quinteiro *et al.* [23], which reuses data from public sources to create a knowledge graph (KG) for training a Graph Neural Network (GNN) to predict drugs that can be repurposed to treat symptoms of rare diseases. We replicate the data fetching process of Perdomo-Quinteiro et al.'s pipeline (named *rd-explainer*) to generate an initial KG. Then, we restructure the KG previously generated based on a conceptual model and compare the performance and outputs of the GNN model when trained on both cases.

Our results highlight promising directions for future research, despite being limited to a single domain and ML algorithm. Experimentation shows that models trained on the conceptual-model-based KGs (CM-based KGs) produced more consistent predictions (i.e. less random), with variability in predictions across different runs of the GNN being 29.91% lower compared to those on the original KGs. Furthermore, the predictive performance of the GNN model trained on the CM-based KGs did not show a significant difference from the original KGs in terms of accuracy and error metrics. These findings suggest that further exploration of CM-based KGs could yield valuable insights. Indeed, Perdomo-

Quinteiro *et al.* note that reproducibility is an issue that needs further investigation: "The known reproducibility issue of our explainer that may imply that the explanations are different each time it is used, may reduce the confidence and reliance on the explanations." [23]

For the sake of clarity, it is important to note that the term "model" carries multiple definitions depending on the research field. In the context of machine learning, a "model" is a mathematical representation or algorithm used to make predictions or decisions based on data [22]. Conversely, in conceptual modelling research, a "model" serves as a structured framework that represents the concepts and relationships within a specific domain, thus providing a formalised approach to organising and interpreting information [10]. To maintain clarity, we differentiate between these definitions using the wordings "ML model" and "conceptual model" to define the different interpretations in each case, respectively.

The remainder of this paper is organised as follows: Sect. 2 provides a brief overview of *rd-explainer*. Section 3 describes the method used in our explorations. Section 4 presents our results, followed by a discussion in Sect. 5. Sections 6 and 7 address the limitations of our study and related works, respectively. Finally, Sect. 8 concludes the paper.

2 The *rd-Explainer* use case

The experimentation described in this work builds upon the method presented by Perdomo-Quinteiro *et al.* [23], which developed *rd-explainer*, an interpretable ML method for drug repurposing. Drug repurposing identifies new uses for existing drugs, a cost-effective strategy particularly valuable for rare diseases with limited pharmaceutical interest [24]. This technique has already been successfully employed in many health care domains, such as in cancer therapy (e.g. [2]) and rare diseases themselves (e.g. [26]). *rd-explainer* relies on aggregated data sourced from three key public knowledge bases: the Monarch Initiative [27], Drug Central [29], and the Therapeutic Target Database [7].

During the pipeline execution, data is initially retrieved from Monarch using a disease code (from a ontology) as the starting point for constructing the initial KG. The fetching script uses the disease code to identify the corresponding disease node in Monarch, and subsequently fetches all nodes directly related to it. The KG is then enriched with data from Drug Central and the Therapeutic Target Database, incorporating information about drugs and their associated treated symptoms. The data from these two additional sources is adjusted to conform to the original graph structure defined by Monarch.

Subsequent to generating the disease-specific KG, a GNN model (GraphSAGE [13]) is trained on it. The output of this process is a ranked list of predictions, with each entry representing the probability of a link existing between a drug and the target symptom. The higher the score, the greater the likelihood of an actual edge existing between the two nodes, indicating a stronger potential relationship between the drug and the symptom or disease. For more information on *rd-explainer*, the reader can refer to Perdomo-Quinteiro *et al.* [23].

3 Method

To restructure the KG produced by the *rd-explainer*, we followed relevant FAIRification steps: identification of FAIRification objectives, data analysis, conceptual modelling, and data restructuring. Subsequently, we compared performance and results of the GNN model when trained on the original KG and on the CM-based one. An illustration of the method described in this section, along with the detailed FAIRification objectives, the original KG metamodel, the conceptual model, the data-fetching and training scripts, the performance measurements and resulting predictions are available in the supplementary material [5].

Identification of FAIRification Objectives. The FAIRification objectives [4] identified in this step focus on making data reusable for drug repurposing applications. These include 'identifying existing drugs that can be repurposed to treat the symptoms of rare diseases', as well as the sub-objectives 'identifying drugs that target genes associated (in)directly with a rare disease' and 'identifying drugs known to treat phenotypes associated (in)directly with a rare disease.'

Data Analysis. In this step, the original KG constructed during the execution of *rd-explainer* was analysed, and its schema was extracted to be used as a starting point for conceptual modelling. An excerpt of this schema is illustrated in Fig. 1, and its complete version is available in the supplementary material [5].

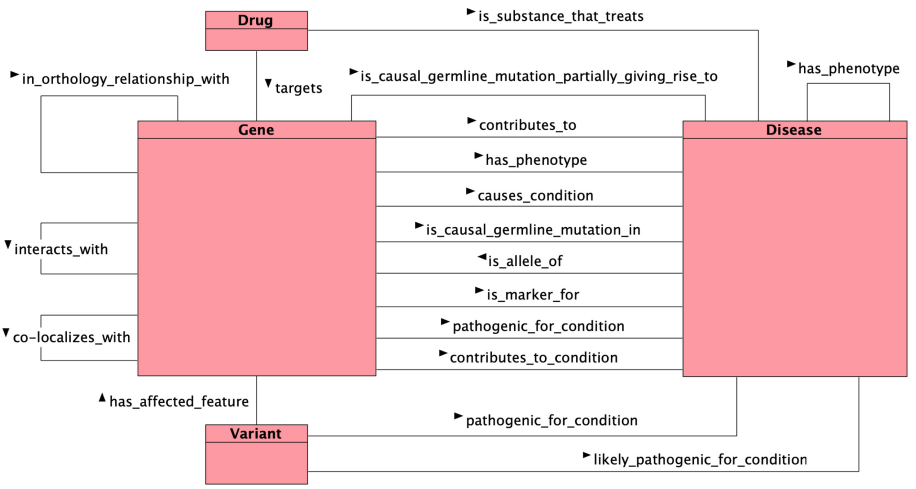


Fig. 1. An excerpt of the original KG schema, showing the concepts Drug, Disease, Gene, and Variant. Rectangles denote node types (rdfs:Class), and lines denote relationship types (rdf:Property).

Conceptual Modelling. The conceptual model for the drug repurposing domain was developed iteratively. This involved designing the domain model using OntoUML [12], a modelling language based on the Unified Foundational Ontology (UFO) [11], which supports the creation of ontologically *well-founded conceptual models*, ensuring semantic clarity in representing real-world phenomena.

The resulting model, designed with FAIRification goals in mind, underwent validation through three rounds of expert review involving bioinformaticians and ontologists. The expert group comprised postdoctoral researchers and PhD candidates, with two university professors also participating in the initial session. All had prior experience in rare disease projects, and most had been involved in drug repurposing efforts for over three years. Each review session included, on average, eight participants. During sessions, the model was presented alongside the FAIRification objectives, and experts were invited to ask clarifying questions and assess the accuracy and relevance of the concepts and relationships. Suggestions were discussed collectively, and those reaching consensus were incorporated into the model. By the third round, the experts agreed the model was sufficiently robust and ready for use.

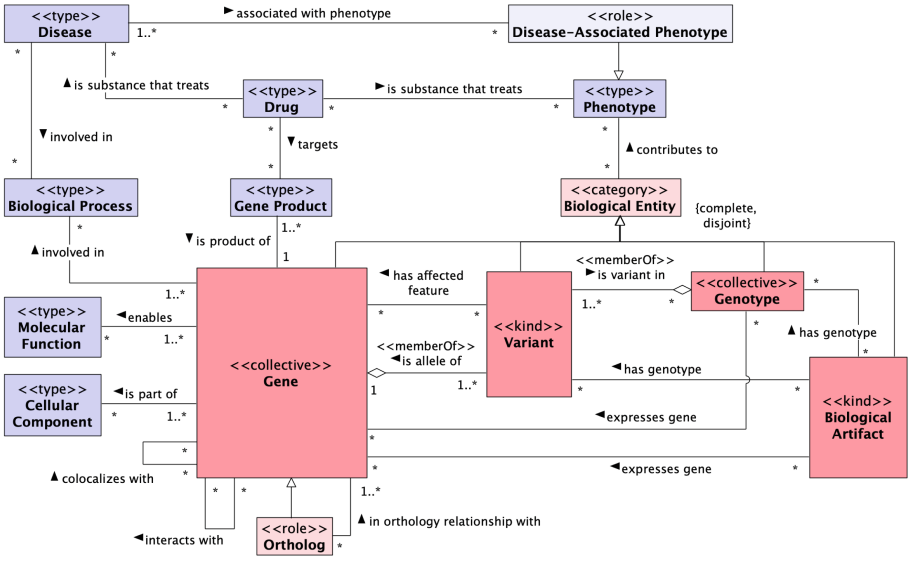


Fig. 2. Conceptual Model for the drug repurposing domain. The resulting CM-based KG mirrors the structure of elements and relationships of the model.

An illustration of the resulting conceptual model is shown in Fig. 2. It represents key biological entities and their relationships within the domain of drug repurposing and (rare) diseases. At the core, the model involves entities such as Gene, Variant, Genotype, and Phenotype, which are fundamental to understanding genetic and phenotypic expressions of diseases. A Gene is a collective

biological entity that may have interactions (e.g. it interacts with or co-localises with other genes). An Ortholog, shown as a subtype of Gene, refers to genes in different species that evolved from a common ancestral gene by speciation and typically retain the same function, making them crucial for studying disease mechanisms across species. Variants are specific alterations in a gene, and they can be part of a Genotype, which represents the complete set of an organism's genetic information. The model shows how Variants and Genotypes *express* Genes, impacting Biological Processes like Molecular Functions and Cellular Components. Additionally, Drugs are connected to Diseases and Phenotypes through their treatment relationships (*is a substance that treats*), thus targeting Gene Products, which are produced by genes and influence disease-related functions.

To exemplify the model in Fig. 2, consider a rare neuromuscular Disease, XYZ syndrome, caused by a specific Variant in the Gene XYZ1. This Variant is part of the patient's overall Genotype. The variant results into an altered Gene Product that is no longer able to reach or function in its intended Cellular Component-the neuromuscular junction-where it normally plays a role in transmitting nerve signals. As a result, the corresponding Molecular Function is disrupted, leading to the Phenotype of *severe muscle weakness*. Research reveals that XYZ1 also has an Ortholog in mouse (xyz1), enabling scientists to model the effects of the variant in vivo and study the Biological Process underlying neuromuscular synapse formation across species. This helps, for instance, identifying a potential Drug through repurposing efforts: it binds to the dysfunctional gene product and partially restores its function, thus mitigating the Disease-Associated Phenotype and offering hope for clinical treatment of XYZ syndrome.

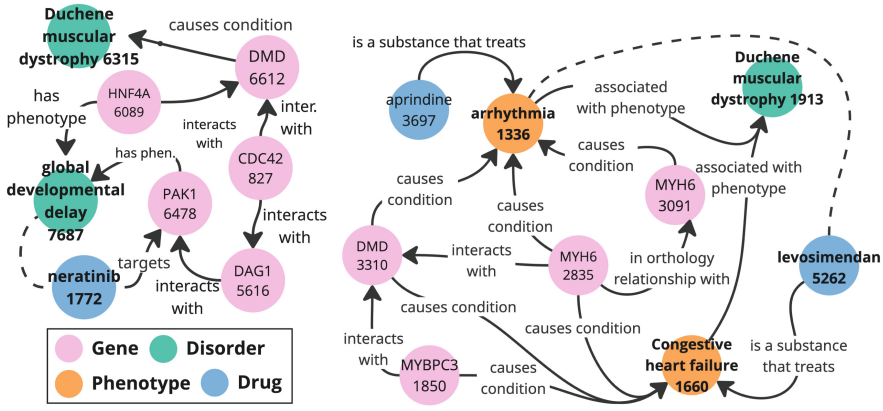
Data Restructuring. Following the conceptual modelling step, the initial KG was reorganised according to the elements and relationships defined in the conceptual model from Fig. 2. To achieve this, the data-fetching script from *rd-explainer* was modified to generate the CM-based KG. The mapping from the original to the CM-based KG was manually reviewed by one of the authors and an external bioinformatician. All data used in this study were retrieved from the September 2021 version of Monarch on 1 May 2024. Data from Drug Central and the Therapeutic Target Database were also collected on 1 May 2024.

GNN Prediction and Performance Assessment. After constructing both the original and CM-based KGs, we proceeded to train separate GNN models on each KG type. This process was repeated ten times per KG type to collect performance metrics that were averaged to ensure a balanced comparison (i.e. AUC-ROC [18], which evaluates a model's capacity to distinguish between classes, where a higher score implies better classification; F1 score [14], which is the harmonic mean of precision and recall; and Cross Entropy Loss [21], which measures the difference between predicted probabilities and true labels, with lower values indicating a more accurate model performance). At the end of this process, 20 prediction lists are generated: 10 lists from the original KG and 10 from the CM-based KG.

Next, to assess the reliability of the ML models, we evaluated the consistency of their predictions. This evaluation was motivated by Perdomo *et al.*'s obser-

vation regarding the challenges of ensuring reproducibility in ML methods. For instance, given the stochastic nature of some components of *rd-explainer* (e.g. edge2vec [9]), running the same process multiple times can lead to different sets of predictions. Therefore, a reliable model should produce consistent results that are not heavily influenced by randomness. To measure this, we performed pairwise comparisons of the prediction lists separately for each of the two groups of 10 iterations from each KG type. For each pair of prediction lists, we calculated the percentage of overlapping predictions, with a lower overlap indicating less consistency across iterations and greater variability in the ML model's outcomes.

Targeting a Rare Disease. As previously mentioned, the process described above requires specifying a target disease (i.e. a disease code) when constructing the KGs, meaning the GNN models are trained on disease-specific KGs. To gather more comprehensive insights from our experiments, we applied our method to three different rare diseases: Duchenne Muscular Dystrophy (DMD) [28], Huntington's Disease (HD) [30], and Osteogenesis Imperfecta (OI) [25]. These diseases have been chosen to maintain consistency in disease characteristics, as each has a single causative gene, in contrast to diseases like Alzheimer's Disease and Amyotrophic Lateral Sclerosis, which were used in Perdomo-Quinteiro *et al.*'s experiments. This process resulted in the construction of six distinct KGs-two for each disease: one original KG and one CM-based KG-and enabled disease-specific comparisons.



(a) original KG (b) CM-based KG

Fig. 3. Illustrations of the knowledge graphs and associated predictions for DMD as target disease. The dashed line represents the prediction itself, while the remaining lines and nodes depict relationships that influence the prediction. Node types are distinguished by colour, as shown in Fig. 3a.

For illustration, Fig. 3 shows an extract of the original and CM-based knowledge graphs generated for DMD as the target disease. It is important to note that assessing the correctness of these predictions is beyond the scope of this paper, as doing so would require proper drug evaluation procedures.

4 Results

A readily applicable outcome of this work is the conceptual model developed during our method (Fig. 2). Moreover, our assessments indicate that the GNN models trained on both types of KGs achieved comparable performance in metrics such as AUC and F1 scores. More significantly, the comparison in terms of output reliability reveals that models trained on CM-based KGs produced more consistent predictions (i.e. similar prediction results). High-resolution versions of the figures presented next and the list of predictions are available in the supplementary material [5].

4.1 The OntoUML-Based Conceptual Model is Reusable

The conceptual model designed in this work can be reused in other ML systems and FAIRification processes, as well as be extended for other applications in related domains. When comparing the original and restructured KGs (Figs. 1 and 2), it can be observed that the number of nodes increased from the original KG to the CM-based KG. In contrast, the number of relationships decreased significantly, as some concepts previously defined as relationships in the original KG were transformed into concepts in the restructured version. For instance, the *has phenotype* relationship in the original KG, which linked Gene and Disease, was transformed into a Phenotype concept in the restructured KG.

4.2 Predictive Performance is Similar

The (averaged) training curves of the GNN models are illustrated in Figs. 4, 5, and 6, for DMD, HD, and OI, respectively. Figures 4a, 5a, and 6a display the training metrics of the GNN models trained on the original KGs, while Figs. 4b, 5b, and 6b show the metrics from the training on the CM-based KGs. Each figure presents the AUC-ROC scores and the Cross Entropy Loss (CEL) of the training processes. When comparing the (a) and (b) versions of each figure, it is important to note that the training curves differ in the total number of epochs as distinct hyperparameter optimisation processes (random search) [3] were performed for each of the six KGs–this is motivated by our aim to evaluate the impact of CM-based restructuring on the entire process. The hyperparameters used in each case are described in the supplementary material [5].

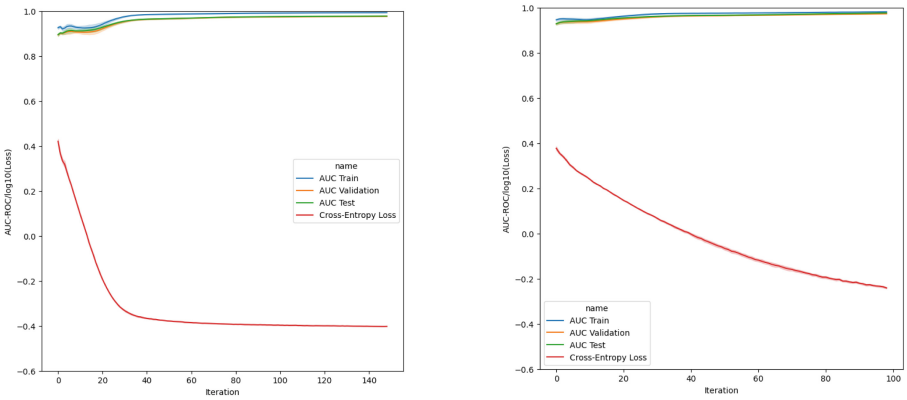
Overall, for all models trained on both the original and CM-based KGs, the training process starts with a high AUC-ROC score for both the training and test sets. However, the improvement from the start to the end of training is minimal. The CEL curve exhibits varying behaviour depending on the target disease. For

DMD, a more rapid decline in the CEL curve is observed when training on the original KG compared to the CM-based KG. In contrast, the opposite trend is seen for HD and OI, where training on the CM-based KG results in a steeper decline.

Table 1 shows a summary of the average AUC-ROC and F1 scores for each case and target rare disease. For DMD, both the AUC-ROC and F1 scores are slightly higher when training the GNN model on the original KG, although the difference is minimal. For HD, training on the original KG resulted in a higher average AUC-ROC, while training on the CM-based KG resulted in a higher F1 score. For OI, the average AUC-ROC and F1 scores were higher when the ML model was trained on the CM-based KG.

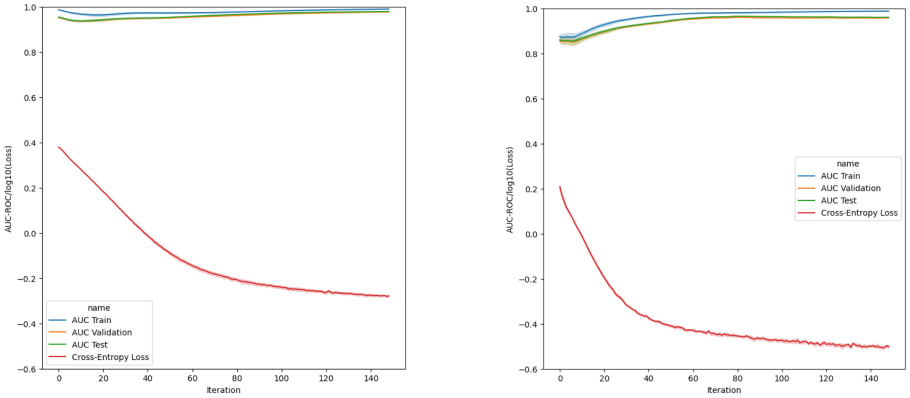
Table 1. AUC-ROC and F1 scores for training on original and CM-based KG, for each target rare disease.

Disease	DMD		HD		OI	
KG Type	Original	CM-based	Original	CM-based	Original	CM-based
AUC-ROC	**0.977**	0.976	**0.978**	0.967	0.9602	**0.974**
F1	**0.933**	0.906	0.896	**0.934**	0.812	**0.915**



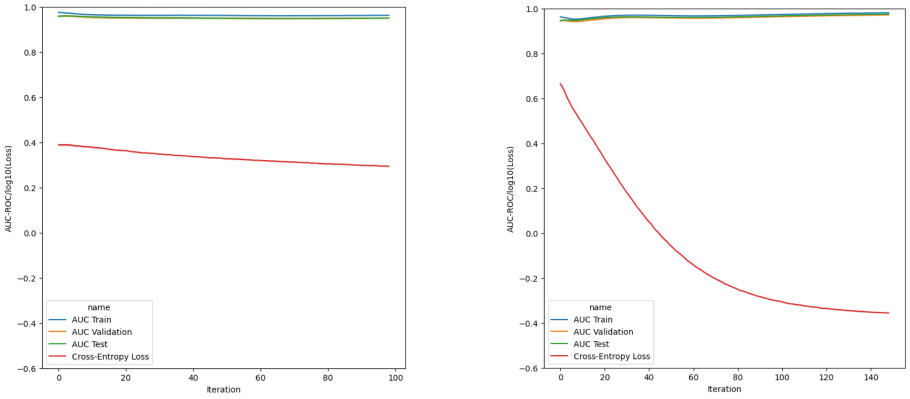
(a) Original KG (b) CM-based KG

Fig. 4. Training curves of the GNN models using the original and CM-based KG as input, for **DMD**. The blue, green and orange lines depict the variations of AUC values among the 10 runs for the train, validation and test sets, respectively. The red line describes the variation in Cross-Entropy Loss.



(a) original KG (b) CM-based KG

Fig. 5. Training curves of the GNN models, for **HD**.

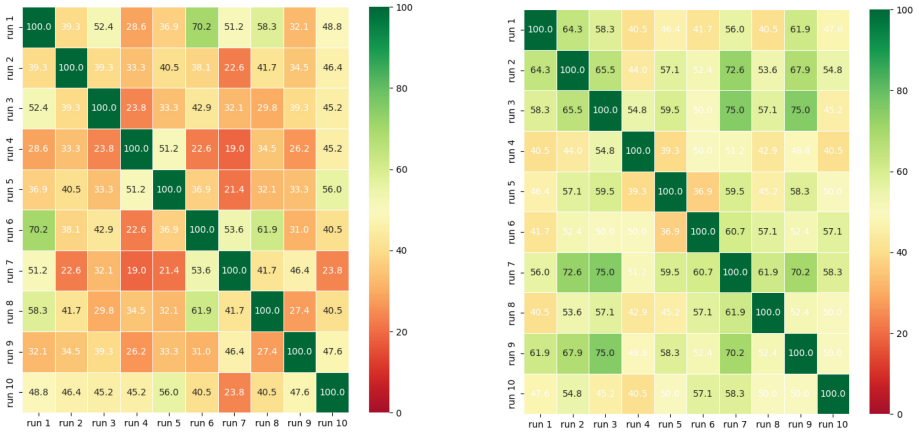


(a) original KG (b) CM-based KG

Fig. 6. Training curves of the GNN models, for **OI**.

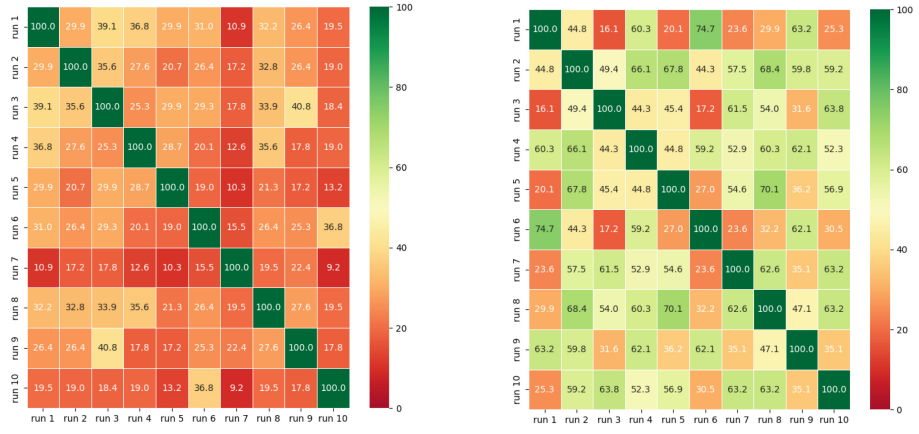
4.3 Predictive Consistency is Higher for CM-Based KGs

Figures 7, 8, and 9, illustrate the degree of overlap (expressed as a percentage) between predicted drug-phenotype pairs across all ten runs for the original and CM-based KG for DMD, HD and OI, respectively. Figures 7a, 8a and 9a are derived from the predictions of the GNN model trained on the original KG, whereas those in Fig. 7b, 8b and 9b are derived from the predictions of the GNN model trained on the CM-based KG. The means and median of the overlaps described in Figs. 7, 8, and 9 are summarised in Table 2. Given the mean and median values of the overlap of predicted drug-phenotype pairs, it can be observed a higher mean and median for when *rd-explainer* is applied to the CM-based KG in the experiments conducted for all target rare diseases.



(a) original KG (b) CM-based KG

Fig. 7. Heat map that shows pairwise overlap of predicted drug-symptom pairs of all ten runs in percentages for each case, for **DMD**.

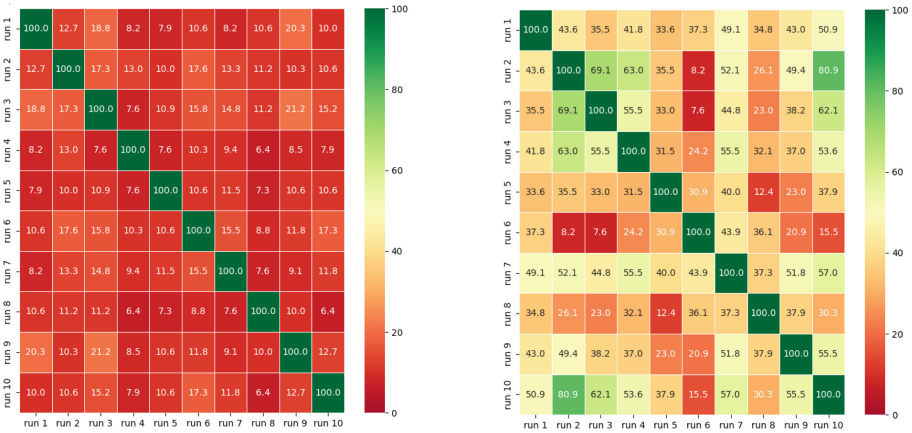


(a) original KG (b) CM-based KG

Fig. 8. Pairwise overlap of predicted drug-symptom pairs of all ten runs in percentages for each case, for **HD**.

Table 2. Summary of the consistency of predictions of all three experiments. The percentages in parenthesis represent the increase of the CM-based values when compared to original KG ones (e.g. increase in mean from original to CM-based).

Disease	DMD		HD		OI	
KG Type	Original	CM-based	Original	CM-based	Original	CM-based
Consistency mean	38.97	**54.1** (+38.82%)	24.27	**48.43** (+99.55%)	11.53	**39.61** (+243.54%)
Consistency median	39.29	**53.57** (+36.35%)	25.29	**52.87** (+109.05%)	10.61	**37.88** (+257.02%)



(a) Original KG (b) CM-based KG

Fig. 9. Pairwise overlap of predicted drug-symptom pairs of all ten runs in percentages for each case, for **OI**.

5 Discussion

We extend our results to propose specific research questions (RQs) to guide future studies. Within the context of FAIR and FAIRification, exploring these RQs will enhance understanding of the benefits of FAIR principles for ML applications. Additionally, addressing these RQs will support gathering more data to make the conclusions of our work more generalisable.

Training Data. When examining the Cross Entropy Loss curve for OI (Fig. 6), it becomes evident that the curve of the ML model trained on the CM-based KG is steeper than that of the ML model trained on the original KG. This may suggest that the ML model trained on the CM-based KG "learned more" when compared to the one trained on the original KG. Consequently, future research should explore whether restructuring training data according to well-founded conceptual models can enhance learning in ML models (e.g. GNN models) that initially do not perform well when trained on current data. Thus, a related RQ could be: *"RQ1 - To what extent do better-structured data improve the predictive performance of ML algorithms that underperform on current data?"*

An example of an experiment to address RQ1 could involve identifying cases where data scientists do not manage to further improve the performance of ML models on specific datasets. In such cases, well-founded conceptual models of the datasets' subjects would be designed and used to restructure those datasets and retrain the ML models.

Reproducibility. When comparing the consistency of predictions from ML models trained on the original and CM-based KGs (Figs. 7, 8, and 9), it is observed that

ML models trained on the latter produce more stable predictions than those trained on the former, as indicated by the average overlap among the 10 lists of predictions generated for each case (Table 2). This suggests that training ML models on data structured according to well-founded conceptual models enhances the consistency of predictions, thereby reducing the randomness of the results. Thus, a research question related to this aspect could be: *"RQ2 - To what extent do conceptual model based data contribute to the consistency of predictions of ML models?"*

To address RQ2, it would be necessary to replicate the experiments conducted in our study in a systematic manner, involving various algorithms and datasets. This approach will ensure that the results are statistically significant and that the conclusions can be generalised and reproduced across different scenarios.

Design of AI Systems. During the application of our method, it was observed that conceptual models improved communication, understanding, and task execution among different stakeholders. A pertinent research question arising from this observation is whether data scientists can enhance their performance in feature engineering, ML model selection, and parameter tuning due to a better understanding of the domain data (provided by conceptual modelling tasks). This leads to the final research question: *"RQ3 - To what extent do conceptual models support stakeholders in the design of AI systems?"*

To test RQ3, a controlled experiment could be conducted in which one group of data scientists and developers is tasked with directly designing and implementing an ML pipeline, while another group is required to first create an OntoUML model before proceeding with the design and implementation of the ML pipeline. The results of these two groups would then be compared to evaluate the impact of conceptual modelling on the design and execution of AI systems. Such an experiment could also test whether the conceptual modelling task facilitates communication between data scientists and domain experts.

It is expected that the conceptual models produced for the potential experiments described above will be (i) constructed using a well-founded modelling language or ontology as a foundation (e.g. UFO, OntoUML), and (ii) thoroughly validated by domain experts. Finally, it is important to highlight that the RQs described above are formulated from our initial explorations. While some are based on subtle differences in the results (e.g. AUC curves), they remain valuable for further investigation, as they may reveal more significant and impactful differences in other ML applications, particularly those involving large KGs.

6 Threats to Validity

Reproducibility and generalisability are the primary limitations of our results. Reproducibility, a well-recognised challenge in ML [1], remains difficult in this context. While our findings demonstrate that CM-based KGs lead to more consistent GNN models, achieving identical results to those presented in Sect. 4 is challenging due, for example, to the inherent randomness in certain components

of the *rd-explainer* pipeline. To mitigate this issue, we ran the training and output generation ten times for each case and target disease, averaging the scores to reduce the impact of variability.

Although we synthesised our findings into additional RQs to guide future research, we have not yet tested our findings sufficiently to draw generalisable conclusions. This limitation arises from the fact that our study focused on a single type of ML method, used the same set of data sources within a specific domain, and was constrained by limited computational resources. Therefore, it is crucial to investigate the impact of conceptual models in different domains and with other types of ML methods, as well as with different types of data. For example, while our results may be relevant to graph data and AI methods designed for such data, the application of our method to other data types may not lead to meaningful differences in results.

Finally, an aspect not addressed in this paper but worth considering in future work is the biological basis for the observed behaviours in the experiments. For example, OI has a relatively homogeneous genetic profile, well-documented phenotypic characteristics, and a well-established body of medical knowledge– particularly in terms of phenotypic markers. These factors might simplify the identification of meaningful patterns within the data, potentially contributing to improved prediction accuracy and consistency.

7 Related Works

The use and impact of conceptual models and ontologies in AI has been a topic of discussion in the literature. Various studies examine how these artefacts can be applied in the field, such as to enhance outcomes and the design of AI systems. At this stage, we view related works as complementary efforts toward a shared ultimate goal: enhancing the understanding and application of ontologies in AI, and vice versa.

Confalonieri & Guizzardi [8] outlined the different roles that ontologies and ontology-based conceptual models can play for neuro-symbolic AI systems from three key perspectives: reference modelling, common sense reasoning, and knowledge refinement and complexity management. Similarly, Maaß & Storey [20] explored the benefits and synergies of integrating conceptual modelling with ML and proposed a framework that uses conceptual modelling to support the design and development of ML solutions.

Lukyanenko *et al.* [19] explored how conceptual modelling can address challenges in extracting insights from large datasets with machine learning. They showed that conceptual modelling aids various ML project phases: defining goals in the business understanding phase, modelling data and identifying quality issues in the data understanding phase, supporting attribute selection and transformation in data preparation, enhancing ML algorithm effectiveness with domain knowledge, improving result interpretability, and documenting process changes during deployment.

8 Final Remarks

This work presented an exploratory study to investigate the impact of restructuring knowledge graphs using conceptual models–a step of FAIRification–on the performance of a GNN algorithm. We tested the GNN model's behaviour when applied to both original and CM-based KGs across three different rare diseases targeted for drug repurposing. The initial results provided valuable insights and led to the formulation of additional refined research questions for future investigation, focusing on areas such as supporting the ML design process, improving predictive performance, and enhancing the consistency of predictions.

The most prominent results of this work relate to the consistency of the predictions. We evaluated the prediction lists generated across ten runs of the GNN method on both the original and CM-based KGs for each target disease. This evaluation involved pairwise comparisons measuring the overlap of predictions in each list. In all cases, the prediction lists generated from training the GNN model on the CM-based KGs were more consistent than those generated from the training on the original KGs.

For FAIR and FAIRification, the initial findings herein presented serve as a proof-of-concept of the benefits of applying the FAIR principles to ML applications. Our results demonstrate that FAIR data, structured using well-defined conceptual models, have the potential to enhance the consistency of ML outputs without negatively impacting performance. Additionally, other elements of FAIR further enhance ML design and deployment. For instance, FAIR metadata improve resource discovery (e.g. finding data for training) and simplifies reuse by clearly specifying the conditions under which the data can be reused.

ML and FAIR research are rapidly evolving fields. Our work contributes to this progress by exploring synergies between conceptual models, FAIR principles, and ML. As the next step, we aim to expand our research by applying our method to new domains, diverse data sources, and a broader range of ML approaches for more robust and generalisable results.

Acknowledgments. This work has received funding from the European Union's Horizon 2020 programme under grant agreements N°825575, N°101156595, N°101132943 and N° 101159589, and the EU4Health Programme under grant agreement N° 101129863.

References

1. Albertoni, R., Colantonio, S., Skrzypczyński, P., Stefanowski, J.: Reproducibility of machine learning: Terminology, recommendations and open issues. arXiv preprint arXiv:2302.12691 (2023)
2. Attia, Y.M., Ewida, H., Ahmed, M.S.: Successful stories of drug repurposing for cancer therapy in hepatocellular carcinoma. In: Drug Repurposing in Cancer Therapy, pp. 213–229. Elsevier (2020)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(2) (2012)

4. Bernabé, C., et al.: Go-plan: a goal-oriented method for FAIRification planning. Inf. Serv. Use 18758789251324008 (2024)
5. Bernabé, C., et al.: Restructuring knowledge graphs with conceptual models - supplementary material (2025). https://doi.org/10.6084/m9.figshare.28576469
6. Brnabic, A., Hess, L.M.: Systematic literature review of machine learning methods used in the analysis of real-world data for patient-provider decision making. BMC Med. Inform. Decis. Mak. **21**(1), 54 (2021)
7. Chen, X., Ji, Z.L., Chen, Y.Z.: TTD: therapeutic target database. Nucleic Acids Res. **30**(1), 412–415 (2002)
8. Confalonieri, R., Guizzardi, G.: On the multiple roles of ontologies in explanations for neuro-symbolic AI. Neurosymbolic Artif. Intell. **1**, 1–14 (2025)
9. Gao, Z., et al.: edge2vec: representation learning using edge semantics for biomedical knowledge discovery. BMC Bioinform. **20**, 1–15 (2019)
10. Guarino, N., Guizzardi, G., Mylopoulos, J.: On the philosophical foundations of conceptual models. In: Information Modelling and Knowledge Bases XXXI, vol. 321, pp. 1–15. IOS Press (2020)
11. Guizzardi, G., Botti Benevides, A., Fonseca, C.M., et al.: UFO: unified foundational ontology. Appl. Ontol. **17**(1), 167–210 (2022)
12. Guizzardi, G., Fonseca, C.M., Almeida, J., Sales, T.P., Benevides, A.B., Porello, D.: Types and taxonomic structures in conceptual modeling: a novel ontological theory and engineering support. Data Knowl. Eng. **134**, 101891 (2021)
13. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Adv. Neural Inf. Process. Syst. **30** (2017)
14. Hand, D.J., Christen, P., Kirielle, N.: F*: an interpretable transformation of the f-measure. Mach. Learn. **110**(3), 451–456 (2021)
15. Hanson, B., et al.: Garbage in, garbage out: mitigating risks and maximizing benefits of AI in research. Nature **623**(7985), 28–31 (2023)
16. Jacobsen, A., Kaliyaperumal, R., da Silva Santos, L.O.B., et al.: A generic workflow for the data FAIRification process. Data Intell. **2**(1-2), 56–65 (2020)
17. Jacobsen, A., et al.: FAIR principles: interpretations and implementation considerations. Data Intell. **2**(1–2), 10–29 (2020)
18. Janssens, A., Martens, F.K.: Reflection on modern methods: revisiting the area under the ROC curve. Int. J. Epidemiol. **49**(4), 1397–1403 (2020)
19. Lukyanenko, R., Castellanos, A., Parsons, J., Chiarini Tremblay, M., Storey, V.C.: Using conceptual modeling to support machine learning. In: Information Systems Engineering in Responsible Information Systems. CAiSE 2019. vol. 350, pp. 170–181. Springer (2019)
20. Maass, W., Storey, V.C.: Pairing conceptual modeling with machine learning. Data Knowl. Eng. **134**, 101909 (2021)
21. Mao, A., Mohri, M., Zhong, Y.: Cross-entropy loss functions: theoretical analysis and applications. In: International Conference on Machine Learning, pp. 23803–23828. PMLR (2023)
22. Ortigossa, E.S., Gonçalves, T., Nonato, L.G.: Explainable artificial intelligence (XAI)–from theory to methods and applications. IEEE Access (2024)
23. Perdomo-Quinteiro, P., Wolstencroft, K., Roos, M., Queralt-Rosinach, N.: Knowledge graphs and explainable AI for drug repurposing on rare diseases. bioRxiv (2024). https://doi.org/10.1101/2024.10.17.618804
24. Pushpakom, S., et al.: Drug repurposing: progress, challenges and recommendations. Nat. Rev. Drug Discov. **18**(1), 41–58 (2019)
25. Rauch, F., Glorieux, F.H.: Osteogenesis imperfecta. Lancet **363**(9418), 1377–1385 (2004)

26. Roessler, H.I., Knoers, N.V., van Haelst, M.M., van Haaften, G.: Drug repurposing for rare diseases. Trends Pharmacol. Sci. **42**(4), 255–267 (2021)
27. Shefchek, K.A., et al.: The Monarch Initiative in 2019: an integrative data and analytic platform connecting phenotypes to genotypes across species. Nucleic Acids Res. **48**(D1), D704–D715 (2020)
28. Szigyarto, C.A.K., Spitali, P.: Biomarkers of Duchenne muscular dystrophy: current findings. Degener. Neurol. Neuromuscul. Dis. 1–13 (2018)
29. Ursu, O., et al.: DrugCentral: online drug compendium. Nucleic Acids Res. gkw993 (2016)
30. Walker, F.O.: Huntington's disease. Lancet **369**(9557), 218–228 (2007)
31. Wilkinson, M.D., et al.: The FAIR guiding principles for scientific data management and stewardship. Sci. Data **3**, 1–9 (2016)

WATCHDOG: an ontology-aWare risk AssessmenT approaCH via object-oriented DisruptiOn Graphs

Stefano M. Nicoletti[1(✉)], E. Moritz Hahn[1], Mattia Fumagalli[2], Giancarlo Guizzardi[1], and Mariëlle Stoelinga[1,3]

[1] University of Twente, Enschede, The Netherlands
{s.m.nicoletti,e.m.hahn,g.guizzardi,m.i.a.stoelinga}@utwente.nl
[2] Free University of Bozen-Bolzano, Bozen, Italy
mattia.fumagalli@unibz.it
[3] Radboud University, Nijmegen, The Netherlands

Abstract. When considering risky events or actions, we must not downplay the role of involved objects: a charged battery in our phone averts the risk of being stranded in the desert after a flat tyre, and a functional firewall mitigates the risk of a hacker intruding the network. The *Common Ontology of Value and Risk* (COVER) highlights how the role of objects and their relationships remains pivotal to performing transparent, complete and accountable risk assessment. In this paper, we operationalize some of the notions proposed by COVER – such as *parthood* between objects and *participation* of objects in events/actions – by presenting a new framework for risk assessment: WATCHDOG. WATCHDOG enriches the expressivity of vetted formal models for risk – i.e., *fault trees* and *attack trees* – by bridging the disciplines of *ontology* and *formal methods* into an ontology-aware formal framework composed by a more expressive modelling formalism, *Object-Oriented Disruption Graphs* (DOGs), logic (DOGLog) and an intermediate query language (DOGLang). With these, WATCHDOG allows risk assessors to pose questions about *disruption propagation*, *disruption likelihood* and *risk levels*, keeping the fundamental role of objects at risk always in sight.

Keywords: ontology · logic · risk · COVER · fault trees · attack trees

1 Introduction

Risk assessment is a key activity to identify, analyze and prioritize the risk in a system, and come up with (cost-)effective countermeasures [38]. This is true when considering *safety* (i.e., the absence of risk connected with unintentional malfunctions) and *security* (i.e., the absence of risk linked with intentional attacks)

This work was partially funded by the NWO grant NWA.1160.18.238 (PrimaVera), the EU's Horizon 2020 Marie Curie grant No 101008233, ERC Consolidator and Proof of Concept Grants 864075 (*CAESAR*) and 101187945 (*RUBICON*).

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 314–331, 2025.
https://doi.org/10.1007/978-3-031-94571-7_18

[35]. To perform transparent, complete and accountable risk assessment, it is fundamental to explicitly account for the role objects play in *Events* (including *Actions*) in which they participate, and for how their status affects safety and security interplay: a door being locked causes the impossible escape event in case of fire but simultaneously stops the action of a burglar entering your house [23,27,36]. Formalisms widely employed in industry and academia to conduct risk assessment – such as *fault trees* [31] and *attack trees* [34] – are not equipped to explicitly reason about objects. Without an explicit representation of objects at risk, it is impossible to evaluate their role in risk scenarios and to correctly evaluate the overall risk imposed on these objects: e.g., what is the risk level my car is subject to, considering that both its tyres can break (safety-related event) and its onboard computer can be hacked (security-related action)?

Our objective here is to provide an ontology-grounded formal approach for object-based risk representation and reasoning by combining and extending standard formalisms for safety (*fault trees*) and security (*attack trees*). To address this lack of expressivity, two promising fields must be taken into account: *ontologies for risk* and *model-based risk assessment*. On the one hand, risk ontologies – like the *Common Ontology of Value and Risk* (COVER) [33] – excel in providing a structured ground for reasoning about a specific domain of knowledge, transparently and explicitly laying out key concepts and relationships needed to reason about risk. While excellent for conceptualization and transparency, ontologies are however not designed to enable quantitative and applied risk evaluations. On the other hand, specific model-based technologies from the field of formal methods – like *fault trees* (FTs) and *attack trees* (ATs) – excel in providing applicable, tried and tested instruments for rigorous and quantitative risk assessment. These methods, however, sometimes rely on opaque conceptual assumptions and, in particular, do not offer the expressivity needed to explicitly reason about objects at risk. With WATCHDOG we propose a risk assessment framework that enriches and extends the expressivity of vetted model-based technologies – such as FTs and ATs – while grounding them in the conceptual clarity of COVER.

Fundamental elements highlighted by the COVER ontological framework [33] – such as the *participation* of a given object in a risk-related *action*/*event* or the *parthood* relationship between different objects – are not expressible in classical risk assessment formalisms, such as FTs and ATs. We address this gap and operationalize these concepts by presenting *object-oriented DisruptiOn Graphs* (DOGs), a new formalism that extends the strengths of classical FT- and AT-based risk analysis accounting for the role of objects at risk (OaRs) in *disruption propagation*, *likelihood* and *risk calculation*[1]. Morever, to perform transparent decision-making w.r.t. safety and security of systems, practitioners need the ability to analyse their models in a meaningful and thorough way. To cater for this need, we present DOGLog – a logic to formally query DOGs – and DOGLang, an intermediate domain-specific language to ease the querying pro-

[1] We adopt these notions from previous work. For a more in-depth discussion on *risk*, *disruption*, *propagation*, and their relation with *risk propagation*, see [12,35].

cess. With DOGLog and DOGLang practitioners can query DOGs to learn meaningful information about systems *disruptions* and *risk levels*. One could ask, for example: Given that one of my tyres breaks, is my entire field trip compromised? Is the probability of an attacker compromising the network larger/smaller than p? Given an object at risk (e.g., my laptop), what is the most risky security Action/safety Event in which it participates? What is the maximum risk level imposed on my laptop, given all the Actions/Events in which it participates? Finally, we showcase property specification in DOGLog and DOGLang on a *DOG* model for a variant of small but well-known and representative example from safety-security literature, modelling safety and security risks on a household given the status of a door lock [23,27,36].

2 Baseline Research

Fault Trees (FTs) and Attack Tress (ATs) constitute a sensible starting point as they are popular technologies that already encode key concepts highlighted in the *Common Ontology of Value and Risk* (COVER) – such as *Events* and *Actions* – and pose a solid ground for model-based risk assessment. Fault tree analysis (FTA) [31] is a widespread technique to support safety risk assessment, and the use of fault trees is required, e.g., by the Federal Aviation Administration, the Nuclear Regulatory Commission, in the ISO 26262 standard [17] for autonomous driving and for software development in aerospace systems. A *fault tree* (FT) (see Fig. 1, right) models *Events* that describe how component failures arise, and propagate disruption through the system, eventually leading to system-level failures. Leaves in a FT represent *basic events* (BEs), i.e. elements of the tree that do not need further refinement. Once these fail, the failure is propagated through the *intermediate events* (IEs) via *gates*, to eventually reach the *top level event* (TLE), which symbolizes system failure. When considering model-based risk assessment of systems security – *attack trees* (ATs) are widely employed. ATs (see Fig. 1, left) are hierarchical diagrams that represent malicious *Actions* that can lead to a system being compromised [24,34]. ATs are referred to by many system engineering frameworks, e.g. *UMLsec* [21] and *SysMLsec* [30], and are supported by industrial tools such as Isograph's *AttackTree* [19]. The TLE of an AT represents the attacker compromising the entire system, and the leaves represent *basic attack steps* (BASes): actions of the attacker that can no longer be refined. As for FTs, intermediate nodes in ATs are labelled with gates.

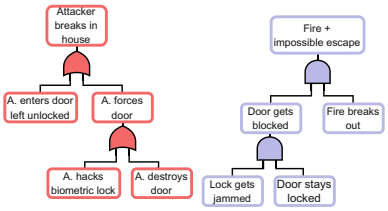


Fig. 1. An attack tree (left) and a fault tree (right) for the locked door example.

The *Common Ontology of Value and Risk* (COVER) [33] is based on *UFO* [16] – a foundational ontology. It embeds a domain-independent conceptualization of risk, has been subject to validation and proper comparison to the literature of risk in risk analysis and management at large (e.g. [18]), and it is built

upon widespread definitions of risk. This ontology has already shown its utility in constructing formalisms for risk quantification and propagation [12], and embeds several key assumptions about the nature of risk, which align with those in the literature in risk assessment. Firstly, risk is **experiential**. This means that the notions of "event" and "object" are deeply entangled and, when assessing the risk an object is exposed to, one aggregates risks ascribed to events that can impact the object. For instance, consider the risks your laptop is exposed to. To assess them, you will need to consider: 1. which of your goals depend on your laptop (e.g. work deliverables); 2. what can happen to your laptop such that it would hinder its capability to achieve your goals (e.g. its screen breaking); 3. which other events could cause these (e.g. you dropping it on the floor). Then the risk your laptop (object) is exposed to is the aggregation of the risk of it falling (event) and breaking (event), and so on. The second assumption is that risk is **contextual**. Thus, the magnitude of the risk an object is exposed to may vary even if all its intrinsic properties (e.g., vulnerabilities or states) stay the same. To exemplify, let us pick one risk event involving your house door, namely that of robbers breaking into it. Naturally, the properties of the door – such as having a strengthened blocking mechanism – influence the magnitude of this risk. Still, the tools used by the robbers can significantly increase how risky the breaking event is. Lastly, another assumption that we derive from COVER is that risk is grounded on **uncertainty** about events and their outcomes. This is a very standard position – see [18] and [3] – which implies that likelihood is positively correlated with how risky an event is. For instance, the risk of encountering a bear is higher while walking in a forest than in an urban park, simply because it is more likely in the former case.

In COVER, a Risk Experience is a multifaceted hypothetical occurrence that can be broken down into Risk Events, further categorized into Threat Events and Loss Events. Threat Events are hypothetical occurrences capable of precipitating Loss Events, which, in turn, are incidents that undermine the objectives of a Risk Subject, the Agent whose viewpoint is under scrutiny in the risk evaluation process. A Loss Event might involve Objects at Risk and Risk Enablers. Dispositions of objects at risk and risk enablers that can be manifested as threat and loss events are Vulnerabilities. As discussed in [33] COVER accounts also for a numerical evaluation of Risk linked to a Risk Assessment. We emphasise here that the quantification of *risk* is applicable solely to *anticipated scenarios* that have the potential (though not certainty) to materialize. The ontology tackles this concern by acknowledging the feasibility of anticipated events, as discussed in [13].

In order to ground WATCHDOG in COVER, we make a number of design assumptions, which are used in the construction of both the new proposed model (DOGs) and logic (DOGLog). We further discuss how they relate to – and are grounded in – the COVER ontology, their implications and limitations. Throughout the paper, we highlight them with the **assumption** tag whenever they play an active role. We distinguish between two types of assumptions: *operational* and *structural*. Operational assumptions introduce constraints that are stricter

than necessary due to the novel nature of this work, requiring a cautious and incremental approach. Structural assumptions, on the other hand, directly derive from the existing COVER ontology as-is. We elaborate on these in the sequel:

- **Assumption 1** (*structural*): the attribution of impact on parent elements in the *DOG* is independent of the attribution on children elements. COVER grounds this assumption by clearly distinguishing the notions of *probability*, *impact*, and *risk*. On DOGs we propagate disruption (i.e., attacks/failures propagation in the system) and – quantitatively – their probability values. In the case of disruption likelihood, the values assigned to each event depend on those assigned to the other events to which they are related. Differently, impact values are assigned independently to each single event. For example, the probability value assigned to a "breaking laptop" event naturally depends on the probability value of a possible related event such as "stumbling". This is not the case for what concerns the loss value (or associated impact) of the two events (in itself, stumbling may not be a problem, while laptop breaking represents something serious). Note that by distinguishing between probability and impact values, we enable a clearer assessment of an event's impact, independent of the likelihood of its occurrence;
- **Assumption 2** (*structural*)**:** OaRs that can participate in parent elements of a *DOG* are a collection of all OaRs that can participate in their children elements, plus additional OaRs added by the risk assessor. This assumption is in line with the representation of events in COVER. In this context, events modelled as children of other events in the graph can be naturally taken as parts of the parent event, and in a simplified view, this allows inferring that objects that participate in parts (or sub-events) of an event, also participate in the event itself;
- **assumption 3** (*operational*)**:** for each *OaR* that participates in an element (event/action) of the *DOG*, we assume that its parts participate in it as well, but the opposite does *not* hold. For instance, if *Door* participates in *Door stays locked* also its part – namely, *Lock* – participates in it, but if *Lock* participates in *Lock breaking* this does not imply that *Door* participates in that event. This is, again, aligned with the theory about events, objects and their parts encoded by UFO and inherited by COVER[2];
- **assumption 4** (*operational*)**:** in computing risk values, we assume the attacker already knows which failure occurred in the system before acting. Looking at the conceptualization provided in COVER, this aligns with the composition of the "risk experience" concept, which not only accounts for the role of "passive" elements involved in the assessment of risk (e.g., "object at risk") but also for the role of the "active" elements involved (see, for instance, the concept of "threat object" and related *threat capability*). This, then, supports the two-step representation of the assessment process we propose, where, firstly, vulnerabilities in a system are identified and, secondly, based on these, threats can be activated. This solution allows for simulations that act as operationalizations of the concept of "risk experience";

[2] For a more detailed focus on this assumption we refer the reader to [14,15].

- **assumption 5** (*operational*)**:** the attacker can adapt its strategy to the event under consideration. E.g., when computing max total risk (Sect. 4) we assume the attacker can maximise the risk level for each individual event in the graph by choosing the best actions at each iteration. Operating in this way gives practitioners the safest possible risk metric, as they are provided with a worst-case upper bound when computing total risk. Also here, the assumption is inspired by how the "risk experience" is represented in COVER, where the "threat capability" of a "threat object" participating in a "threat event" is always directly related to "loss events" and related risks.

3 Object-Oriented Disruption Graphs

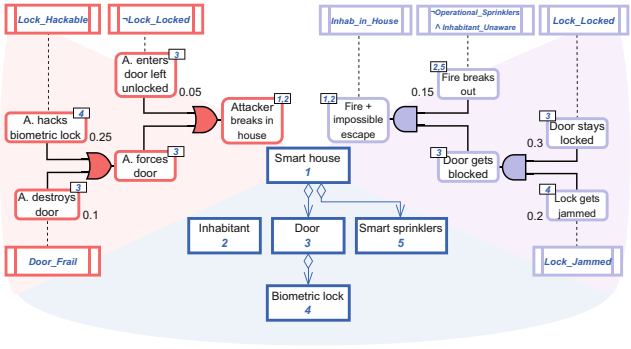


Fig. 2. *DOG* for the smart house locked door example. (Color figure online)

Object-oriented DisruptiOn Graphs (DOGs) extend classical FT- and AT-based risk analysis by integrating key concepts from COVER, i.e., adding needed constructs to reason about objects at risk during risk assessment. Figure 2 represents an object-oriented disruption graph that captures risks on a smart house. This scenario extends the well-known locked door example, typical of safety-security literature [23,27,36], with IS engineering specificities. On the left – in **red**– *Actions* of an attacker are represented in an AT. On the right – in **violet**– *Events* that can cause failures are represented in a FT. Root nodes in both the FT and AT – the *top level events* (TLEs) – can be mapped to COVER's *Loss Events*. Each of the events/actions in the FT and AT is labelled with *objects at risk* (OaRs) that can *participate* in it – a white rectangle on the corner, containing numbers for objects that participate in the labelled event/action. These numbers refer to the *Object Graph* – at the bottom, in **blue**– where arrows in UML-like notation represent the *parthood* relationship between objects. Specific combinations of properties of OaRs that are needed for events/actions to happen – what we call *conditions* – are typeset in blue and linked to events/actions via a

dashed line. For example, the condition ¬*Lock_Locked* is needed for the action *Attacker enters door left unlocked* to happen. Finally, example probability values are typed next to each leaf node in the FT/AT components. With this model, one can ask ontology-aware questions that do not overlook the role of objects at risk, e.g., 1. Given that an *Attacker destroys the door* and that the *Fire does not break out*, are any of the two loss events happening? 2. Is the probability of both successfully forcing the door and fire breaking out lower than 0.05? 3. What is the most risky event in which *Inhabitant* participates, assuming that *Lock is Locked*? 4. What is the minimal risk level associated with the OaR *Door*, given all the events/actions in which it participates?
In Sect. 5 we will formalize these queries via our logic (DOGLog) and query language (DOGLang).

Definition 1 (*Object-Oriented Disruption Graph*). *An* Object-Oriented Disruption Graph *(DOG) G is a tuple (A, F, O, B) where A is an attack tree, F is a fault tree, O is an object graph and B is a disruption knowledge base.*

As mentioned in [35], ATs and FTs can be syntactically unified under the *disruption tree* (*DT*) model:

Definition 2 (*Disruption Tree*). *A* disruption tree *(DT) T is a tuple (N, E, t) where (N, E) is a rooted directed acyclic graph, and $t : N \rightarrow \{\texttt{OR}, \texttt{AND}, \texttt{LEAF}\}$ is a function s.t. for $v \in N$, it holds that $t(v)$=*`LEAF` *iff v is a leaf. Moreover, ch: $N \rightarrow 2^N$ gives the set of* children *of a node and T has a unique root, i.e., R_T.*

We also define the set of intermediate events $\texttt{IE} = N \setminus \texttt{LEAF}$. Moreover, if $u \in ch(v)$ then u is called a *child* of v, and v is a *parent* of u. Furthermore, we employ only *AND*- and *OR*-gates in the AT/FT components of the model. The behaviour of a *DT* T can be expressed through its *structure function* [31] - f_T: if we assume the convention that a *LEAF* has value 1 if disrupted and 0 if operational, the structure function indicates the status of the root node – or *top level event* (*TLE*) – given the status of all the LEAVEs of T. Thus, for each set of LEAVEs we can identify its characteristic vector $\vec{b}$: we refer to this vector as a *scenario*. We denote by $\mathscr{S}_T = 2^{LEAF_T}$ the universe of scenarios of T. When further distinction is needed between ATs and FTs constructs, we use (respectively) the subscripts $\__A$ and $\__F$: e.g., we refer to a scenario on an AT (resp. FT) as an *attack scenario* (resp. *fault scenario*), represented by $\vec{b}_A$ (resp. $\vec{b}_F$). As shown before, in FT- and AT-related literature nodes canonically represent respectively *events* and *attack steps*: one might easily map *attack steps* and *events* to the terminology chosen in the COVER ontology, for which nodes N_A of an AT represent *Actions* and nodes N_F of a FT represent *Events*. From this point on, we will use the general term *elements* to refer indistinctly to nodes in FTs and ATs. To enrich ATs and FTs, we introduce *Objects at Risk* (OaRs) that explicitly capture impacted objects in (safety and security) risk experiences.

Definition 3 (*Object Graph*). *An* object graph *(OG) O is a rooted directed acyclic graph (N_O, E_O, OP, cOP) where: 1. nodes in N_O represent* Objects at Risk *(OaRs); 2. directed edges in $E_O \subseteq N_O \times N_O$ represent the* parthood *relation between OaRs; 3.* properties *on OaRs are atomic propositions $op \in$ OP; and 4. $cOP\colon N_O \to 2^{OP}$ returns a set of atomic propositions of a node $v \in N_O$.*

Moreover, $ch\colon\ N_O \to 2^{N_O}$ gives the set of *parts* of a node and O has a unique root, denoted R_O. As previously hinted, OaRs and the object graph (OG) are represented by connected blue rectangles (see Fig. 2, page 6). Similarly to *DT*s' evaluation, we need a way to evaluate properties of OaRs, thus:

Definition 4 (*Evaluating Properties of OaRs*). *We let a* configuration $\vec{b}_O$ *be the Boolean vector assigning values to properties of OaRs in OP and we let $f_O\colon \mathbb{B}^n \times OP \to \mathbb{B}$ be a valuation such that $f_O(\vec{b}_O, op) = 1$ iff the Boolean value of a property $op \in OP$ equals 1 given $\vec{b}_O$.*

Furthermore, we let $\mathscr{C}$ be the set of all possible configurations. Finally, we introduce a *Disruption Knowledge Base* (DKB) that establishes a formal relation between elements in ATs and FTs and OaRs that can participate in them. We also define attribution of an impact value to ATs and FTs elements and their *preconditions* and *conditions* in the DKB. *Preconditions* are relevant properties of OaRs that can participate in a given event/action. These properties must be arranged in a specific way for events/actions to happen: *conditions* express this arrangement.

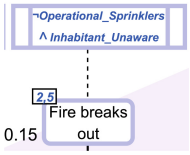


Fig. 3. An excerpt of Fig. 2.

Example 1. Consider the excerpt of the *DOG* of Fig. 2 in Fig. 3: conditions for the event *Fire breaks out* – in blue, connected with a dashed line – are encoded in the Boolean formula $\neg Operational\_Sprinklers \wedge Inhabitant\_Unaware$: in fact, the OaRs *Smart house* and *Inhabitant* participate in the event *Fire breaks out* and *Operational_Sprinklers* and *Inhabitant_Unaware* are the relevant properties of these participating objects at risk. These are exactly the *preconditions* for event *Fire breaks out* and they must be set resp. to *false* and *true* in conjunction for *Fire breaks out* to happen.

Definition 5 (*Disruption Knowledge Base*). *A* disruption knowledge base *(DKB) B is a tuple (D, Im, Pa, Pr) where:*

1. *$D = N_A \cup N_F \cup N_O$ is an* entity domain *where N_A, N_F and N_O are pairwise disjoint*
2. *$Im\colon N_A \cup N_F \to \mathbb{R}_{\geq 0}$ is a function that returns an* impact factor *for each* element *$v \in N_A \cup N_F$*
3. *$Pa\colon N_A \cup N_F \to 2^{N_O}$ is a function that for each* element *$v \in N_A \cup N_F$ returns a set of OaRs that can participate in v*
4. *For each element $v \in N_A \cup N_F$, $Pr\colon N_A \cup N_F \to 2^{OP}$ is a function that returns the set of its* preconditions *$Pr(v) = cOP(o_1) \cup cOP(o_2) \ldots \cup cOP(o_n)$ with $o_i \in Pa(v)$*

5. *For each element* $v \in N_A \cup N_F$, conditions *on* v *are represented by a Boolean formula* $Cond(v)$ *over its* preconditions $Pr(v)$

Do note that *conditions* naturally map to *descriptions of situations* in COVER. **Assumption 1:** Moreover, we assume that attribution of impact on parent elements (see item 2) is independent of the attribution on children elements. **Assumption 2:** We also assume that OaRs that can participate in parent elements (events/actions) – being decorated with AND-gates or OR-gates in the DT– are a collection of all OaRs that can participate in their children elements, plus additional OaRs added by the risk assessor (exactly what the Pa function returns). As seen with properties of OaRs, we need a way to evaluate whether conditions on a given DT element (it being an action or an event) are satisfied:

Definition 6 (*Evaluating Conditions of Elements*). *We let* $f_{Cond}\colon \mathbb{B}^n \times Form \rightarrow \mathbb{B}$ *be the evaluation function for conditions – with* $Form$ *being the set of Boolean formulae – such that given a configuration* $\vec{b_O}$ *and conditions* $Cond(v)$ *for an element* v, $f_{Cond}(\vec{b_O}, Cond(v)) = 1$ *iff the Boolean assignment in* $\vec{b_O}$ *satisfies* $Cond(v)$.

To summarize: for each element v we construct a Boolean formula $Cond(v)$ over the set of preconditions $Pr(v)$, which are exactly the relevant *properties* of OaRs that *participate* in v which are needed for v to happen. To do so, we collect the n OaRs that can participate in v with $Pa(v)$ and – for each of these objects $o_i \in \{o_1, \ldots, o_n\}$ – we collect its *properties* via $cOP(o_i)$. **Assumption 3:** For each *OaR* that participates in element (event/action) v, we assume that its parts participate in v as well, but the opposite does *not* hold. The union of all these collected sets $cOP(o_i)$ is exactly the set $Pr(v)$ of *preconditions* on v. The Boolean formula $Cond(v)$ over preconditions in $Pr(v)$ for v represents its *conditions*. It is important to note that, in a practical setting, a user would iteratively be asked whether 1. all participating objects in v and 2. all preconditions of a given participating *OaR* are relevant for conditions on v. The ability to isolate and compute separately both preconditions and conditions will be functional in this setting. E.g., the property *Lock_ Locked* is relevant for preconditions of an element $v =$ *Attacker enters door left unlocked*, but *Lock_ Hackable* is not: thus, it is not included as an atom in $Cond(v)$. As hinted, understanding computations on DOGs requires further attention on the interplay between Actions (resp. Events) in ATs (resp. FTs) and their conditions. Since $Cond(v)$ is a Boolean formula over preconditions for $v \in N_A$ (resp. $v \in N_F$), for v to be attacked (to fail) the entire Boolean formula composed by $v \wedge Cond(v)$ must evaluate to *true*. To account for this, let us define an extended structure function for DTs where, for v to be disrupted it would be necessary to have $f_T^e(\vec{b}, \vec{b_O}, v)$ return 1, i.e., both the node in question should be disrupted and its conditions must be satisfied.

Definition 7 (*Extended Structure Function*). *The extended structure function of a disruption tree* T *is a function* $f_T^e\colon \mathbb{B}^n \times \mathbb{B}^n \times N \rightarrow \mathbb{B}$ *that takes as input a scenario* $\vec{b}$, *a configuration* $\vec{b_O}$ *and an arbitrary element* $v \in N$. *We define it as follows:*

$$f^{\mathcal{O}}_{T}(\vec{b}, \vec{b}_O, v) = \begin{cases} b_i \wedge f_{Cond}(\vec{b}_O, Cond(v)) & \text{if } v = v_i \in \texttt{LEAF} \\ \bigvee\limits_{v' \in ch(v)} f^{\mathcal{O}}_{T}(\vec{b}, \vec{b}_O, v') \wedge f_{Cond}(\vec{b}_O, Cond(v)) & \text{if } v \in \texttt{IE and } t(v) = \texttt{OR} \\ \bigwedge\limits_{v' \in ch(v)} f^{\mathcal{O}}_{T}(\vec{b}, \vec{b}_O, v') \wedge f_{Cond}(\vec{b}_O, Cond(v)) & \text{if } v \in \texttt{IE and } t(v) = \texttt{AND} \end{cases}$$

4 DOGLog: A Logic to Reason About DOGs

We construct our logic on three syntactic layers, represented with ϕ, ψ and ξ. Layer 1 formulae reason about disruption propagation: the atomic propositions a in DOGLog can represent any element in an AT or FT, and any property of OaRs, i.e., $a \in N_A \cup N_F \cup OP$. Formulae can be combined through usual Boolean connectives. Furthermore, we can set evidence to construct what-if scenarios: $\phi[a \mapsto bool]$ sets the element a in ϕ to either 0 or 1, representing an event/action taking place or an object property being *true* or *false*. Finally, DOGLog allows reasoning about *minimal risk scenarios* (MRSs): minimal assignments on leaves of the FT and AT, such that a formula is satisfied (e.g., such that an event/action takes place). Note that MRSs are always evaluated by fixing a specific configuration, i.e. a specific status of object properties. Layer 2 formulae reason about disruption propagation probabilities. We can check whether the disruption probability of a given ϕ formula (e.g., a given event/action) is bounded by a selected threshold – with our comparison operator $\bowtie \in \{<, \leq, =, \geq, >\}$ – and we can also set probabilistic evidence, i.e., formulate scenarios where an event/action $e \in N_A \cup N_F$ is assigned a specific disruption probability q. Combining layer 2 formulae with Boolean operators is also allowed. Lastly, layer 3 reasons about safety- and security-related risk levels. One can ask what are the most risky actions/events in which an OaR participates – with $* \in \{A, F\}$ symbolizing resp. AT and FT nodes, i.e., actions and events. Moreover, one can ask what is the max/min total risk level to which an OaR is subject, aggregating risk from both safety- and security-related events/actions in which it participates, and what is the optimal configuration of object properties to minimise risk on an OaR. Finally, one can also set evidence on object properties, i.e., forcefully set them to *true* or *false* to create insightful what-if scenarios. Note that when setting evidence we usually assign values to $a \in$ LEAVEs $\cup OP$ in layer 1, and to $e \in$ LEAVEs in layer 2. We can however assign values to IEs of ATs and FTs if 1. $a/e \in N_A \cup N_F$ is a module [10], i.e., all paths between descendants of a/e and the rest of the AT or FT pass through a/e 2. and none of the descendants of a/e are present in the formula. If so, we prune that (sub)AT or (sub)FT (and relative conditions) and treat occurring IEs as LEAVEs. We can formally define the syntax for DOGLog as follows:

Layer 1: $\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \phi[a \mapsto bool] \mid \text{MRS}(\phi)$
Layer 2: $\psi ::= \text{P}(\phi) \bowtie p \mid \neg\psi \mid \psi \wedge \psi \mid \psi[e \mapsto q]$
Layer 3: $\xi ::= \mathsf{MostRisky}_{*}(o) \mid \underset{\text{max}}{\mathsf{TotalRisk}}(o) \mid \underset{\text{min}}{\mathsf{TotalRisk}}(o) \mid \mathsf{OptimalConf}(o) \mid \xi[op \mapsto bool]$

5 Object-Oriented Risk Queries: DOGLog and DOGLang

To ease the usability of our logic, we present DOGLang, a Domain Specific Language (DSL) for DOGLog. Defining languages and tools to specify properties and requirements is common: in [9] the authors capture high-level requirements for a steam boiler system in a human-readable form with SADL. Further controlled natural languages for knowledge representation include Processable English (PENG) [37], Controlled English to Logic Translation (CELT) [28], Computer Processable Language (CPL) [7] and FRETish [8]. DOGLang is constructed by adhering to the same design philosophy of *LangPFL* – a domain specific language for FTs that was developed in the literature [25]. As for *LangPFL*, DOGLang is

Table 1. Risk queries for G (Fig. 2) in natural language, DOGLog and DOGLang.

Natural Language	Property in DOGLog	DOGLang
Given that an *Attacker destroys the door* and that the *Fire does not break out*, are any of the two TLEs happening?	$TLE1 \vee TLE2$ $[ADD \mapsto 1, FBO \mapsto 0]$	**assume:** set ADD = 1 set FBO = 0 **check:** TLE1 or TLE2
What are all the *MRS*s such that both *Loss Events* happen, given that *Lock is Locked* and *Door is Frail*?	$[\![TLE1 \wedge TLE2$ $[LiL \mapsto 1, DiF \mapsto 1]]\!]_{\mathscr{M}}$	**assume:** set LiL = 1 set DiF = 1 **computeall:** MRS[TLE1 and TLE2]
Is the probability of both successfully forcing the door and fire breaking out lower than 0.05?	$\mathsf{Prob}(AFD \wedge FBO) < 0.05$	**assume:** **check:** Prob[AFD and FBO] < 0.05
What is the most risky event in which *Inhabitant* participates, assuming that *Lock is Locked*?	$\mathsf{MostRisky}_{F}(Inhab.)[LiL \mapsto 1]$	**assume:** set LiL = 1 **computeall:** MostRiskyF[Inhab.]
What is the max risk level associated with the OaR *Door*, given all the events/actions in which it participates?	$\underset{\text{max}}{\mathsf{TotalRisk}}(Door)$	**assume:** **compute:** MaxTotalRisk[Door]
What is the minimum risk level on *Door*, assuming that OaR *Look* exhibits property *Lock Pick-able*?	$\underset{\text{min}}{\mathsf{TotalRisk}}(Door)[LP \mapsto 1]$	**assume:** set LP = 1 **compute:** MinTotalRisk[Door]
What are the properties that all OaRs must exhibit, in order to minimise the risk level associated with the object *House*, assuming we fix that *Door is Frail*?	$\mathsf{OptimalConf}(House)[DiF \mapsto 1]$	**assume:** set DiF = 1 **computeall:** OptimalConf[House]

inspired by the aforementioned languages for their ease of use and close proximity to natural language. DOGLang expresses only a fragment of DOGLog. Notably, nesting of formulae is disallowed: we retain most of the expressiveness of DOGLog while making property specification easier. In DOGLang, *DOG* elements are referred to with their short label and each operator in DOGLog has a counterpart in the DSL: Boolean operators, not, and, or, impl ...; setting the value of *DOG* elements to Boolean or probabilistic values, set, set_prob; minimal risk scenarios MRSs , MRS[...] ; operators to check disruption probability thresholds, Prob[...] $\bowtie$ ... (note that $\bowtie \in \{<, \leq, =, \geq, >\}$); and to reason about risk levels aggregated on a given object and about risky actions/events in which this object participates, MostRiskyA[...], MostRiskyF[...], MaxTotalRisk[...], MinTotalRisk[...], OptimalConf[...]. One can specify properties in DOGLang by utilizing operators inside structured templates. Assumptions on the status of *DOG* elements can be specified under the assume keyword. These assumptions will be automatically integrated with the translated formula accordingly, e.g., set or set_prob will be translated with the according operators to set evidence, while other assumptions will be the antecedent of an implication. A second keyword separates specified formulae from the assumptions and dictates the desired result: compute and computeall compute and return desired values, i.e., probability values, and lists of events/actions/configurations and MRSs respectively, while **check** establishes if a specified property holds.

In Table 1 we exemplify some queries on the *DOG* for the smart house locked door example in Fig. 2. These queries exemplify the expressive power enabled by DOGs, DOGLog and DOGLang and are chosen to reflect the different Layers of our logic. It is important to note that syntactically the *DOG* model does not represent answers to these queries right away, so one cannot read them from Fig. 2 directly. Answers are computed as per semantics of both the model and the logic: e.g., the *A. forces door* node is a composite element, and the computation of this complex probability value follows probability composition on the structure of the model in Fig. 2.

6 Enabling Risk Computations: DOGLog Semantics

To enable object-oriented risk computations and to ground the meaning of formulae into the enriched model presented in Sect. 3, we define formal semantics for DOGLog. For the first layer of the logic, formulae are evaluated on the following model $\mathscr{M} = \langle \vec{b_R}, \vec{b_O}, G \rangle$ where a *risk scenario* $\vec{b_R} = (b_1, \ldots, b_k)$ is defined as $\vec{b_R} = \vec{b_A} \cup \vec{b_F}$, $\vec{b_O}$ is a configuration and G is a *DOG*. Formally:

$$
\begin{aligned}
&\mathscr{M} \models a && \text{iff} \begin{cases} \text{with } a \in N_A & f_A^{\mathscr{O}}(\vec{b}_A, \vec{b}_O, a) = 1 \\ \text{with } a \in N_F & f_F^{\mathscr{O}}(\vec{b}_F, \vec{b}_O, a) = 1 \\ \text{with } a \in OP & f_O(\vec{b}_O, a) = 1 \end{cases} \\
&\mathscr{M} \models \neg\phi && \text{iff } \mathscr{M} \not\models \phi \\
&\mathscr{M} \models \phi \wedge \phi' && \text{iff } \mathscr{M} \models \phi \text{ and } \mathscr{M} \models \phi' \\
&\mathscr{M} \models \phi[a_i \mapsto \mathit{bool}] && \text{iff} \begin{cases} \text{with } a_i \in N_A \cup N_F & \mathscr{M}' \models \phi \text{ with } \vec{b}_R^{\,\prime} = (b'_1, \ldots, b'_k) \in \mathscr{M}', \\ & b'_i = \mathit{bool} \in \mathbb{B} \text{ and } b'_j = b_j \text{ for } j \neq i \\ \text{with } a_i \in OP & \mathscr{M}' \models \phi \text{ with } \vec{b}_O^{\,\prime} = (b'_1, \ldots, b'_m) \in \mathscr{M}', \\ & b'_i = \mathit{bool} \in \mathbb{B} \text{ and } b'_j = b_j \text{ for } j \neq i \end{cases} \\
&\mathscr{M} \models \text{MRS}(\phi) && \text{iff } \vec{b}_R \in [\![\phi]\!]_{\mathscr{M}}
\end{aligned}
$$

With $[\![\phi]\!]_{\mathscr{M}}$ we denote the *minimal satisfaction set* of risk scenarios for ϕ, i.e., the set of minimal risk scenarios $\vec{b}_R$ that satisfy ϕ given G. We define $[\![\phi]\!]_{\mathscr{M}}$ as follows: $[\![\phi]\!]_{\mathscr{M}} = \{\vec{b}_R \mid \langle \vec{b}_R, \vec{b}_O, G\rangle \models \phi \wedge \nexists \vec{b}_R^{\,\prime}.\vec{b}_R^{\,\prime} \subseteq \vec{b}_R \wedge \langle \vec{b}_R^{\,\prime}, \vec{b}_O, G\rangle \models \phi\}$. Note that the set of all minimal risk scenarios for a given ϕ – i.e., $[\![\phi]\!]_{\mathscr{M}}$ – is always computed by fixing a specific configuration $\vec{b}_O$ first.

Layer two formulae require the introduction of probabilities. First, we need to decorate the leaves of the AT and the FT in G with probability values. To do so, we let an *attribution on* G be a map $\alpha\colon \text{LEAVEs} \to [0, 1]$. With a slight abuse of notation, we simply write α_G for the probability attribution on both the leaves of the AT A and the FT F in G. We then let $\rho(\phi)$ define the probability of a given layer one formula ϕ. Intuitively: given ϕ and a configuration $\vec{b}_O$, we consider every possible fault scenario $\vec{b}_F \in \mathscr{S}_F$ on F and how that would impact truth values of FT nodes in ϕ. For each of these fault scenarios, we compute the maximal probability of successfully attacking AT nodes in ϕ under the given configuration $\vec{b}_O$. **Assumption 4:** Note that – with this setup – we assume the attacker already knows which FT nodes in ϕ failed. Consequently, we let:

$$\rho(\phi, \vec{b}_O)_{A,F} = \sum_{\vec{b}_F \in \mathscr{S}_F} \mathsf{Prob}(\vec{b}_F) \times \mathbb{P}_A(\mathsf{Set}(\phi, \vec{b}_F, \vec{b}_O))$$

where the probability associated to each fault scenario $\vec{b}_F \in \mathscr{S}_F$ – with v as a BE– is calculated via $\mathsf{Prob}(\vec{b}_F) = \prod_{i=1}^{k} b_i \times \alpha(v_i) + (1 - b_i) \times (1 - \alpha(v_i))$ and where the maximal probability of successfully attacking ϕ is given by multiplying attributions on BASes in every minimal attack scenario for ϕ – $\vec{b}_A \in [\![\phi]\!]_A$ – to then take the maximum between the resulting values of these attacks. Formally:

$$\mathbb{P}_A(\phi) = \max_{\vec{b}_A \in [\![\phi]\!]_A} \prod_{v \in \vec{b}_A} \alpha(v)$$

This last step is coherent with a more general framework for multiple metric computations on ATs previously defined in [24,26]. Finally, we account for how every possible fault scenario would impact truth values of atomic propositions

of FT nodes in ϕ by recursively defining $\mathsf{Set}(\phi, \vec{b}_F, \vec{b}_O)$, with $\vec{b}_F \in \mathscr{S}_F$:

$$\mathsf{Set}(a, \vec{b}_F, \vec{b}_O) = \begin{cases} \text{with } a \in N_A & a \\ \text{with } a \in N_F & \begin{cases} 1 & \text{iff } f_F^{\circledcirc}(a, \vec{b}_F, \vec{b}_O) = 1 \\ 0 & \text{otherwise} \end{cases} \\ \text{with } a \in OP & \begin{cases} 1 & \text{iff } f_O(a, \vec{b}_O) = 1 \\ 0 & \text{otherwise} \end{cases} \end{cases}$$

$$\mathsf{Set}(\neg\phi, \vec{b}_F, \vec{b}_O) = \neg\mathsf{Set}(\phi, \vec{b}_F, \vec{b}_O)$$

$$\mathsf{Set}(\phi \wedge \phi', \vec{b}_F, \vec{b}_O) = \mathsf{Set}(\phi, \vec{b}_F, \vec{b}_O) \wedge \mathsf{Set}(\phi', \vec{b}_F, \vec{b}_O)$$

$$\mathsf{Set}(\phi[a_i \mapsto bool], \vec{b}_F, \vec{b}_O) = \mathsf{Set}(\phi, \vec{b}_F, \vec{b}_O)[a_i \mapsto bool]$$

$$\mathsf{Set}(\mathrm{MRS}(\phi), \vec{b}_F, \vec{b}_O) = \mathrm{MRS}(\mathsf{Set}(\phi, \vec{b}_F, \vec{b}_O))$$

where 1 and 0 represent the *true* and *false* derived layer 1 formulae. Note that – also due to Set – some occurrences can lead to the application of the $\mathbb{P}_A$ function to either *true* or *false*, i.e., when $\phi = 1$ or $\phi = 0$. In these cases, we fix that $\mathbb{P}_A(1) = 1$ and $\mathbb{P}_A(0) = 0$. With $\bowtie \in \{<, \leq, =, \geq, >\}$ and an updated model $\mathsf{M} = \langle \vec{b}_O, G, \alpha_G \rangle$, semantics for layer two formulae can be defined as follows:

$$\mathsf{M} \models \mathrm{P}(\phi) \bowtie p \text{ iff } \rho(\phi, \vec{b}_O)_{A,F} \bowtie p; \qquad \mathsf{M} \models \neg\psi \text{ iff } \mathsf{M} \not\models \psi;$$

$$\mathsf{M} \models \psi \wedge \psi' \quad \text{iff } \mathsf{M} \models \psi \text{ and } \mathsf{M} \models \psi';$$

$$\mathsf{M} \models \psi[e_i \mapsto q] \text{ iff } \mathsf{M}(\alpha_G[\text{with}\,\alpha(v_i) \mapsto q]) \models \psi, \text{ with } v_i \in N_A \cup N_F$$

Note that the model M for layer 2 formulae does not contain a risk scenario $\vec{b}_R$: this is because in computing probabilities we already account for both 1. possible fault scenarios $\vec{b}_F$ and 2. possible attack scenarios $\vec{b}_A$. Layer 3 formulae require further attention on OaRs and the participation relation. To compute the risk level associated with an OaR o, given a certain configuration – e.g., the risk level of *Door*, given that *Lock_Locked* is set to *false* – we first identify the *set of elements in which o participates, and for which a satisfying risk scenario plus configuration exist*. We can consider events/actions as layer one atomic formulae a s.t. $a \in N_A \cup N_F$, and – with $* \in \{A, F\}$– formally define this set as:

$$(\!|o|\!)_* = \left\{ a \in N_* \mid o \in Pa(a) \wedge \exists \vec{b}_*, \vec{b}_O . f_*^{\circledcirc}(\vec{b}_*, \vec{b}_O, a) = 1 \right\}$$

Note that one might want to parameterize some elements of the given configuration $\vec{b}_O$ to, e.g., compute optimal assignments to minimize risk on a given OaR. To accommodate for this need, we let $\mathscr{C}_{[op \mapsto bool]}$ be the set of configurations that could still be compatible with a partial Boolean assignment $[op \mapsto bool]$. E.g.:

Example 2 Assume we want to consider only the configurations compatible with setting the evidence that *Lock_Locked* is *true*, i.e., $[Lock\_Locked \mapsto 1]$ and that we only have two other object properties to consider, the values of which are still not assigned. The resulting partial configuration can be represented as $\vec{b}_O = (1, \cdot, \cdot)$, where $\cdot$ represents the unassigned values of remaining object properties.

The set $\mathscr{C}_{[Lock\_Locked \mapsto 1]}$ would then contain all possible configurations whose assignments are still compatible with $\vec{b}_O = (1, \cdot, \cdot)$: e.g., $\vec{b}'_O = (1, 1, 0)$ would be in $\mathscr{C}_{[Lock\_Locked \mapsto 1]}$, while $\vec{b}''_O = (0, 1, 0)$ would be excluded from the set since the value of the first object property is set to zero (against $[Lock\_Locked \mapsto 1]$).

We let $\mathsf{objRiskVal} = \sum_{a \in (\!|o|\!)_A \cup (\!|o|\!)_F} (\rho(a, \vec{b}_O)_{A,F} \times Im(a))$ represent the cumulative risk value on a specific OaR o, given events and actions in which it participates. Intuitively, we sum the risk values from each event/action in which o participates, resulting from the probability of each event/action times its impact factor. Given a set of configurations $\mathscr{C}$, we let $\mathsf{Val}_{\mathscr{C}}$ define semantics for layer 3 formulae:

$$\mathsf{Val}_{\mathscr{C}}(\mathsf{MostRisky}_A(o)) = \operatorname*{argmax}_{a \in (\!|o|\!)_A} \max_{\vec{b}_O \in \mathscr{C}} (\rho(a, \vec{b}_O)_{A,F} \times Im(a));$$

$$\mathsf{Val}_{\mathscr{C}}(\mathsf{MostRisky}_F(o)) = \operatorname*{argmax}_{a \in (\!|o|\!)_F} \max_{\vec{b}_O \in \mathscr{C}} (\rho(a, \vec{b}_O)_{A,F} \times Im(a));$$

$$\mathsf{Val}_{\mathscr{C}}\left(\underset{\max}{\mathsf{TotalRisk}}(o)\right) = \max_{\vec{b}_O \in \mathscr{C}} \mathsf{objRiskVal}; \qquad \mathsf{Val}_{\mathscr{C}}\left(\underset{\min}{\mathsf{TotalRisk}}(o)\right) = \min_{\vec{b}_O \in \mathscr{C}} \mathsf{objRiskVal};$$

$$\mathsf{Val}_{\mathscr{C}}(\mathsf{OptimalConf}(o)) = \operatorname*{argmin}_{\vec{b}_O \in \mathscr{C}} \mathsf{objRiskVal}; \qquad \mathsf{Val}_{\mathscr{C}}(\xi[op \mapsto bool]) = \mathsf{Val}_{\mathscr{C}_{[op \mapsto bool]}}(\xi)$$

Assumption 5: Note that with semantics as given, the attacker can adapt its strategy to the node under consideration. E.g., when computing max total risk we assume the attacker can maximise the risk level for each individual node in the graph by choosing the best BASes at each iteration. We then sum risk levels derived from each of these single-node worst-case scenarios.

7 Related Work

This paper directly relates to approaches that seek to combine ATs and FTs and increase their expressive capabilities. In this sense, numerous works attempt combinations of FTs and ATs into joint safety-security models: these are collected in a recent survey on model-based formalisms for safety-security risk assessment [27]. Of the 14 selected formalisms in [27], 7 combine or extend FTs and ATs: Attack-Fault Trees [2], Component Fault Trees [22], Extended Fault Trees – also known as Fault Trees with Attacks [11], Boolean driven Markov processes (BDMPs) [5], Attack Tree Bow Ties [1], Failure-Attack-CounTermeasure Graphs [32], and State/Event Fault Trees (SEFTs) [29]. Of these formalisms, only BDMPs and SEFTs explicitly integrate properties of objects by joining FTs and Petri nets, expressing that certain disruptions can only happen in certain states. However, both BDMPs and SEFTs do not explicitly address how 1. the *parthood* relation between objects and 2. the *participation* relation between objects and events/actions can influence the propagation of disruptions and the computations of risk levels. Furthermore, they do not allow for aggregation of risk levels on a given object, nor do they allow to compute an optimal configuration of states to minimize risk. Lastly, [20] presents Object-Oriented Fault Trees (OFTs), where each FT node is described by an object with instance variables

containing information such as the node's parents, children and type. Despite the name, OFTs do not account for objects participating in different events represented by FT nodes, nor can they account for risk aggregation on objects, given safety- and security-related events/actions.

8 Discussion

To validate our approach, we intend to perform requirement elicitation from users: we then plan to translate these elicited requirements into concrete queries, modeled after those in Sect. 5 that serve for now a simple exemplification purpose. The effectiveness of our approach will be evaluated on the basis of the types of queries we are able to capture. Additionally, we aim to conduct user studies via the development of a mock-up implementation to further evaluate the approach hereby presented. Ideally, we envision users to be risk analysts, as our approach is a natural extension of models that they might already be familiar with. Finally, as far as scalability is concerned, we expect to formulate novel symbolic model-checking algorithms that encode presented computational semantics in Binary Decision Diagrams (BDDs) [6]: BDDs are promising as they have proven to be computationally successful and efficient already in the classical setting of FT and AT analysis [4].

It is important to notice that in the case of *operational* assumptions – stricter than the absolute necessary – we take care to never be incoherent with COVER: i.e., we could lift or weaken these assumptions in future work, while remaining still grounded in the COVER ontology. E.g., one could weaken **assumption 3** to account for the principle of mereological expansion or **assumption 4** – fundamental for risk computation – when considering a different ordering between failures and attacks, or again **assumption 5** to model different types of attackers. These diverse possibilities increase flexibility in modelling disruptive situations while remaining grounded in COVER.

9 Conclusion and Future Work

We presented WATCHDOG, an ontology-aware framework for object-oriented risk assessment that exploits both the strengths of model-based formal methods and ontologies: by combining ontologies with probabilistic risk quantification models, we enriched the expressive power of FTs and ATs and presented a more expressive ontology-aware model (DOGs), logic (DOGLog) and a query language (DOGLang). We chose COVER for its domain-independent nature and its foundation in a comprehensive analysis of existing work on risk ontologies. However, our approach remains flexible and does not preclude the integration of knowledge from other ontologies. Our research opens up interesting directions for future work. First, one could allow a more nuanced notion of propagation or of parthood, considering the parthood relationship of OaRs and mereology. Furthermore, one could enrich DOGs by introducing new concepts from (the

COVER) ontology, e.g., the notion of *goal*, or by introducing multiple *risk assessors* viewpoints via multiple ATs and FTs components. Finally, one could consider the effect of weakening assumptions, as discussed in Sect. 8.

References

1. Abdo, H., Kaouk, M., Flaus, J.M., Masse, F.: A new approach that considers cyber security within industrial risk analysis using a cyber bow-tie analysis (2017)
2. Arnold, F., Guck, D., Kumar, R., Stoelinga, M.: Sequential and parallel attack tree modelling. In: Proc. SAFECOMP. pp. 291–299 (2015)
3. Aven, T., Renn, O., Rosa, E.A.: On the ontological status of the concept of risk. Saf. Sci. **49**(8), 1074–1079 (2011)
4. Basgöze, D., Volk, M., Katoen, J., Khan, S., Stoelinga, M.: BDDs Strike Back - Efficient Analysis of Static and Dynamic Fault Trees. In: (NFM). vol. 13260, pp. 713–732 (2022)
5. Bouissou, M., Bon, J.L.: A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes. RESS (2003)
6. Brace, K.S., Rudell, R.L., Bryant, R.E.: Efficient implementation of a BDD package. In: 27th ACM/IEEE Design Automation Conference, pp. 40–45 (1990)
7. Clark, P., Harrison, P., Jenkins, T., Thompson, J.A., Wojcik, R.H., et al.: Acquiring and using world knowledge using a restricted subset of English. In: Flairs Conference, pp. 506–511 (2005)
8. Conrad, E., Titolo, L., Giannakopoulou, D., Pressburger, T., Dutle, A.: A compositional proof framework for FRETish requirements. In: CCP, pp. 68–81 (2022)
9. Crapo, A., Moitra, A., McMillan, C., Russell, D.: Requirements capture and analysis in ASSERT (TM). In: RE, pp. 283–291. IEEE (2017)
10. Dutuit, Y., Rauzy, A.: A linear-time algorithm to find modules of fault trees. IEEE Trans. Reliab. **45**(3), 422–425 (1996)
11. Fovino, I.N., Masera, M., De Cian, A.: Integrating cyber attacks within fault trees. Reliab. Eng. Syst. Saf. **94**(9), 1394–1402 (2009)
12. Fumagalli, M., et al.: On the semantics of risk propagation. In: RCIS (2023)
13. Guarino, N.: On the semantics of ongoing and future occurrence identifiers. In: Mayr, H.C., Guizzardi, G., Ma, H., Pastor, O. (eds.) ER 2017. LNCS, vol. 10650, pp. 477–490. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69904-2_36
14. Guarino, N., Baratella, R., Guizzardi, G.: Events, their names, and their synchronic structure. Appl. Ontol. **17**(2), 249–283 (2022)
15. Guizzardi, G., Wagner, G., de Almeida Falbo, R., Guizzardi, R.S., Almeida, J.P.A.: Towards ontological foundations for the conceptual modeling of events. In: ER, pp. 327–341. Springer (2013)
16. Guizzardi, G., et al.: UFO: unified foundational ontology. Appl. Ont. **17**(1) (2022)
17. International Standardization Organization: ISO/DIS 26262: Road vehicles, functional safety (2018). https://www.iso.org/standard/68383.html
18. ISO: Risk Management - Vocabulary, ISO Guide 73:2009 (2009)
19. Isograph: AttackTree (2023). www.isograph.com/software/attacktree/
20. Iverson, D.L., Patterson-Hine, F.: A diagnosis system using object-oriented fault tree models. In: Proceedings of the Artificial Intelligence for Space Applications, pp. 341–9 (1990)

21. Jürjens, J.: UMLsec: extending UML for secure systems development. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 412–425. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45800-X_32
22. Kaiser, B., Liggesmeyer, P., Mäckel, O.: A new component concept for fault trees. In: SCS, pp. 37–46. Citeseer (2003)
23. Kriaa, S., Bouissou, M., Colin, F., Halgand, Y., Pietre-Cambacedes, L.: Safety and security interactions modeling using the BDMP formalism: case study of a pipeline. In: SAFECOMP, pp. 326–341. Springer (2014)
24. Lopuhaä-Zwakenberg, M., Budde, C.E., Stoelinga, M.: Efficient and generic algorithms for quantitative attack tree analysis. IEEE TDSC (2022)
25. Nicoletti, S.M., Lopuhaä-Zwakenberg, M., Hahn, E.M., Stoelinga, M.: PFL: a probabilistic logic for fault trees. In: FM 2023, pp. 199–221 (2023)
26. Nicoletti, S.M., Lopuhaä-Zwakenberg, M., Hahn, E.M., Stoelinga, M.: ATM: a logic for quantitative security properties on attack trees. In: SEFM (2023)
27. Nicoletti, S.M., Peppelman, M., Kolb, C., Stoelinga, M.: Model-based joint analysis of safety and security: survey and identification of gaps. Comp. Sci. Rev. **50** (2023)
28. Pease, A., Murray, W.: An English to logic translator for ontology-based knowledge representation languages, In: NLP-KE. pp. 777–783. IEEE (2003)
29. Roth, M., Liggesmeyer, P.: Modeling and analysis of safety-critical cyber physical systems using state/event fault trees. In: SAFECOMP (2013)
30. Roudier, Y., Apvrille, L.: SysML-Sec: a model driven approach for designing safe and secure systems. In: MODELSWARD, pp. 655–664. IEEE (2015)
31. Ruijters, E., Stoelinga, M.: Fault Tree Analysis: a survey of the state-of-the-art in modeling, analysis and tools. Comp. Sci. Rev. **15–16**, 29–62 (2015)
32. Sabaliauskaite, G., Mathur, A.P.: Aligning cyber-physical system safety and security. In: Complex Systems Design & Management Asia, pp. 41–53. Springer (2015)
33. Sales, T.P., et al.: The common ontology of value and risk. In: ER (2018)
34. Schneier, B.: Attack trees. Dr. Dobb's journal **24**(12), 21–29 (1999)
35. Stoelinga, M., Kolb, C., Nicoletti, S.M., Budde, C.E., Hahn, E.M.: The marriage between safety and cybersecurity: still practicing. In: SPIN, pp. 3–21 (2021)
36. Sun, M., Mohan, S., Sha, L., Gunter, C.: Addressing safety and security contradictions in cyber-physical systems. In: CPSSW. Citeseer (2009)
37. White, C., Schwitter, R.: An update on PENG light. In: ALTA, pp. 80–88 (2009)
38. Zio, E.: The future of risk assessment. Reliab. Eng. Syst. Saf. **177**, 176–190 (2018)

Tutorials

Empowering Development and Use of DSMLs with the FMML^{X} and the XModeler$^{ML©}$

Ulrich Frank(✉)

Universität Duisburg-Essen, Essen, Germany
ulrich.frank@uni-due.de
https://umo.ris.uni-due.de/en/

Abstract. Tutorial participants will learn how multi-level language architectures significantly facilitate the design and implementation of DSMLs and how they enable new application system architectures that not only promote reusability and flexibility, but also foster user empowerment.

Keywords: language design · language architecture · multi-level modeling

1 Goals

The tutorial aims at providing an inspiring introduction to the development and implementation of DSMLs and applications with a multi-level language architecture. Based on a demonstration of how traditional language architectures like MOF compromise the development of models, DSMLs and of self-referential applications in general, the participants will learn how essential concepts of multi-level language architectures help to overcome these obstacles - and why they are especially suited for certain application areas including digital twins. Examples and case studies serve the illustration of these advantages. They are demonstrated with the freely available tool XModeler$^{ML©}$, which the participants may use during the tutorial.

2 Scope and Audience

This tutorial is appropriate for researchers and practitioners who want to know what multi-level language architectures are all about. They will learn essential features of these architectures and how they facilitate an efficient approach to developing more expressive, executable DSMLs, models in general, and powerful application systems that are integrated with models and languages at runtime. Participants should have knowledge of conventional information modelling, as exemplified by UML as well as of object-oriented programming.

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15701, pp. 335–336, 2025.
https://doi.org/10.1007/978-3-031-94569-4

3 Structure

The tutorial consists of four parts. The first part focuses on basic language concepts. They are motivated by examples that illustrate the substantial weaknesses of traditional language architectures such as MOF. Further examples serve to demonstrate the use of multi-level concepts within the language engineering, modelling and execution environment XModeler$^{ML©}$.

The second part consists of two case studies. The first case study demonstrates how a UML object model can be substantially improved by a multi-level reconstruction with respect to reuse, integration, adaptability and integrity. To that end, a short introduction to the inbuilt constraint language XOCL, which is similar to OCL, is given. The second case study shows the development and implementation of a DSML with the FMML^{X}and the XModeler$^{ML©}$. It includes the specification and implementation of a domain-specific concrete syntax. Both cases serve to make clear that the joint representation of model and program makes the effort required for approaches to synchronize both representations superfluous. In addition, it will be shown that corresponding application architectures empower users by allowing to navigate and eventually modify underlying conceptual models during runtime.

The third part aims at those participants who are interested in specific aspects of the underlying golden braid architecture. They are provided with a brief overview of core concepts and references to more detailed documentation. In the fourth part, we will discuss strategies of how to smoothly migrate from MOF like architectures to multi-level architectures. While multi-level language architectures constitute a new paradigm, they do not necessarily require to give up on existing models and architectures. The tutorial will be taught in an interactive mode. Questions as well as critical comments are highly appreciated.

The XModeler$^{ML©}$as well as videos, sample models and references are available at the project's webpages (www.le4mm.org). That allows participants to use the tool during the tutorial. Those who want to experience the flavour of multi-level modelling without leaving known territory may start their journey with UML-MX$^{©}$, a UML model editor that is integrated with the XModeler$^{ML©}$. It features the common representation of classes and objects and allows the execution of objects.

References

1. Clark, T., Sammut, P., Willans, J.: Applied metamodelling: a foundation for language driven development. https://arxiv.org/abs/1505.00149 (2008)
2. Clark, T. Frank, U.: Multi-level constraints. In: Proceedings of MODELS 2018 Workshops (2018)
3. Frank, U.: Multi-level modeling: cornerstones of a rationale. Softw. Syst. Model. **21**(1), 451–480 (2022). https://doi.org/10.1007/s10270-021-00955-1
4. Frank, U.: Prolegomena of a multi-level modeling method illustrated with the FMML^{X}. In: MODELS 2021. Proceedings of the 24th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (2021)

Addressing Vulnerabilities in Information Systems Engineering

Avi Shaked[1(✉)] and Nan Messe[2]

[1] Department of Computer Science, University of Oxford, Oxford, UK
avishakedse@gmail.com
[2] IRIT, Toulouse University, CNRS, INP, UT2, Toulouse, France
nan.messe@irit.fr

Abstract. This tutorial equips information systems researchers and practitioners with the knowledge and tools to manage vulnerabilities systematically and enhance the security and resilience of their systems. We introduce key cybersecurity knowledge bases and concepts, present a structured approach to vulnerability management based on cross-disciplinary interfaces, and explore the use of Artificial Intelligence in this domain.

Keywords: Vulnerability management · Cybersecurity · Security design

1 Tutorial Overview

Vulnerability management is a critical aspect of cybersecurity and information security. While cybersecurity is often perceived as a specialized field, existing cybersecurity bodies of knowledge provide a wealth of accessible content. Non-specialists can draw on this material to design information systems for security, provided they adopt a disciplined, integrative approach grounded in an informed understanding of cybersecurity concepts and frameworks. In this tutorial, we offer an overview of key security concepts and knowledge bases, highlighting their role in shaping a sustainable view of systems' security postures. We then present a practical, systematic approach to managing the vulnerabilities of information systems. Finally, we conclude by exploring the potential role of Artificial Intelligence in vulnerability management.

1.1 Cybersecurity Bodies of Knowledge

In contrast to the prevalent "specialist" perspective on cybersecurity, we empower the information systems community to address security concerns and engage in meaningful discussions with security professionals. We do this by introducing participants to prominent cybersecurity knowledge bases, explaining the core security concepts they embody, the relationships among these concepts, and other key security-related terms.

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15701, pp. 337–338, 2025.
https://doi.org/10.1007/978-3-031-94569-4

We survey the following knowledge bases and frameworks: Common Attack Pattern Enumerations and Classifications (CAPEC), Common Weakness Enumeration (CWE), Common Vulnerabilities and Exposures (CVE), and NIST Special Publication 800-53 (Security and Privacy Controls for Information Systems and Organizations). The surveyed concepts include: vulnerability, weakness, threat, attack pattern, software bill of materials (SBOM), asset, and security control.

1.2 Vulnerability Management

We describe the *vulnerability management* problem and its underlying dynamics. We establish a by-design mitigation approach as a marker of maturity, drawing on publications from security agencies worldwide. We also discuss various mechanisms for vulnerability detection, including code reviews and SBOM scanners.

We then introduce a novel, systematic approach to vulnerability management, which relies on a well-defined information-exchange interface between systems engineers and security leaders [5]. This enables integration of data from various knowledge bases and supports automated reasoning [6]. We demonstrate this approach using the open-source TRADES Tool [4].

Additionally, we explore complementary methods for vulnerability specification, identification, analysis, and reasoning – e.g., [1–3] – as well as the use of logic programming languages. We conclude by examining the potential benefits and limitations of applying neuro-symbolic Artificial Intelligence to address real-world challenges in vulnerability management.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alfasi, D., Shapira, T., Bremler-Barr, A.: VulnScopper: unveiling hidden links between unseen security entities. In: Proceedings of the 3rd GNNet Workshop on Graph Neural Networking Workshop, pp. 33–40 (2024)
2. Rouland, Q., Hamid, B., Jaskolka, J.: A model-driven formal methods approach to software architectural security vulnerabilities specification and verification. J. Syst. Softw. **219**, 112219 (2025)
3. Safronov, V., Bostan, I., Allott, N., Martin, A.: How memory-safe is IoT? Assessing the impact of memory-protection solutions for securing wireless gateways. In: 1st International Workshop on Internet of Things for Safety-Critical Cyber Physical Systems (2024)
4. Shaked, A.: A model-based methodology to support systems security design and assessment. J. Ind. Inf. Integr. **33**, 100465 (2023)
5. Shaked, A., Messe, N.: BridgeSec: facilitating effective communication between security engineering and systems engineering. J. inf. Secur. Appl. **89**, 103954 (2025)
6. Shaked, A., Messe, N., Melham, T.: Modelling tool extension for vulnerability management. In: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, pp. 56–60 (2024)

Semantic Interoperability Masterclass from Foundational Principles to Interactive Knowledge Cartography

Nicolas Figay[1(✉)] and Parisa Ghodous[2]

[1] Airbus Defence and Space SAS, Elancourt, France
Nicolas.figay@airbus.com
[2] LIRIS Laboratory, UMR 5205, University Lyon 1, Lyon, France
parisa.ghodous@univ-lyonl.fr

Abstract. In the contemporary digital landscape, achieving semantic interoperability is a key challenge for enterprise collaboration, knowledge sharing, and system integration. This masterclass explores a holistic and innovative approach, combining enterprise architecture (ArchiMate), domain ontologies (OWL), product and process models, and AI-enhanced service architectures into a unified, interactive and composite hyper-modeling framework. By leveraging semantic cartography, participants will learn how to explore knowledge, information, data, and services in a flexible and interconnected manner, supporting decision-making in networked enterprises.

Keywords: Semantic Interoperability · Polyglot hypermodel · Interactive Knowledge Cartography

1 Description

This session provides both foundational principles and practical applications, covering the nature of meaning, distinctions between data, information, and knowledge, and the role of ontologies in facilitating shared under-standing across diverse systems. A particular focus will be given to polyglot hypermodels, enabling visualization and interaction across heterogeneous representations. Participants will gain hands-on experience with OWL modeling, ArchiMate-based enterprise structures, and semantic graph-based exploration using interactive tools. AI's role in automating interoperability and enhancing reasoning capabilities will also be discussed.

By the end of the masterclass, participants will be equipped with the necessary knowledge and tools to apply se-mantic interoperability principles to real-world enterprise challenges, fostering enhanced communication, integration, and digital continuity within complex ecosystems.

The tutorial will rely on the usage of ArchiCG, a research demonstrator for interactive semantic cartography.

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 339–340, 2025.
https://doi.org/10.1007/978-3-031-94571-7_21

1.1 Goal

To equip participants with a comprehensive understanding of semantic interoperability, enabling them to model domain ontologies using OWL, structure enterprise architecture with ArchiMate, and create interactive semantic cartographies for enhanced system integration and decision-making.

1.2 Scope

Intended Audience: Information systems engineers, enterprise architects, knowledge management professionals, and researchers interested in semantic technologies and AI-enhanced interoperability.
Prerequisites: Basic understanding of semantic concepts and enterprise modelling is beneficial but not mandatory.
Topic Relevance and Novelty: Enterprises increasingly adopt AI, digital twins, and knowledge graphs, requiring a new meta-approach to interoperability. This masterclass introduces hypermodels as interactive and composite knowledge structures, blending ontologies, enterprise models, and service-oriented architectures into a seamless, navigable framework.

2 Structure of the Content

1. Introduction to the Notion of Meaning
2. Understanding Knowledge, Information, and Data
3. Communication Challenges in Natural Language and Data
4. Introduction to Ontologies and Polyglot Hypermodels
5. Foundations in Logic and Set Theory
6. Semantic Modeling with OWL and Interactive Graph Exploration
7. Building Semantic Interoperability through Composite Hypermodels
8. AI-Augmented Interoperability and Future Perspectives
9. Conclusion

References

1. Figay, N., et al.: Dynamic manufacturing network - from flat semantic graphs to composite models. Int. J. Prod. Res. **57**(20), 6569–6578 (2019)
2. Figay, N., Ghodous, P.: Extended hypermodel for Interoperability within the virtual enterprise. In: SITIS, pp. 393–400 (2009)
3. ArchiCG. https://nfigay.github.io/ArchiCG/. ArchiCG 1.16 (2025)

Author Index

J. Krogstie et al. (Eds.): CAiSE 2025, LNCS 15702, pp. 341–343, 2025.
https://doi.org/10.1007/978-3-031-94571-7