

The Enterprise Engineering Series

Vinay Kulkarni · Sreedhar Reddy
Tony Clark · Henderik A. Proper

The AI-Enabled Enterprise

 Springer


The Enterprise Engineering Series

Founding Editors


Jan L. G. Dietz, Technical University of Delft, Delft, The Netherlands

José Tribolet, Instituto Superior Tecnico, Technical University of Lisbon,
Lisboa, Portugal

Editors-in-Chief

David Aveiro , Faculty of Exact Sciences and Engineering, University of Madeira,
Funchal, Portugal

Robert Pergl, Faculty of Information Technologies, Czech Technical University in Prague,
Praha 6, Czech Republic


Henderik A. Proper , TU Wien Informatics, Vienna, Austria

Editorial Board Members

Joseph Barjis, Institute of Engineering and Management, San Francisco, USA

Giancarlo Guizzardi , Free University of Bozen-Bolzano, Bolzano, Italy

Jan A. P. Hoogervorst, Antwerp Management School, Antwerp, Belgium

Hans B. F. Mulder , University of Antwerp, Antwerp, Belgium

Martin Op't Land, Antwerp Management School, Antwerp, Belgium

Marné de Vries, Industrial and Systems Engineering, University of Pretoria, Pretoria,
South Africa

Robert Winter, Institute for Information Management, University of St. Gallen, St. Gallen,
Switzerland


Enterprise Engineering is an emerging discipline for coping with the challenges (agility, adaptability, etc.) and the opportunities (new markets, new technologies, etc.) faced by contemporary enterprises, including commercial, nonprofit and governmental institutions. It is based on the paradigm that such enterprises are purposefully designed systems, and thus they can be redesigned in a systematic and controlled way. Such enterprise engineering projects typically involve architecture, design, and implementation aspects.


The Enterprise Engineering series thus explores a design-oriented approach that combines the information systems sciences and organization sciences into a new field characterized by rigorous theories and effective practices. Books in this series should critically engage the enterprise engineering paradigm, by providing sound evidence that either underpins it or that challenges its current version. To this end, two branches are distinguished: Foundations, containing theoretical elaborations and their practical applications, and Explorations, covering various approaches and experiences in the field of enterprise engineering. With this unique combination of theory and practice, the books in this series are aimed at both academic students and advanced professionals.

Vinay Kulkarni • Sreedhar Reddy • Tony Clark •
Henderik A. Proper


The AI-Enabled Enterprise

 Springer

Vinay Kulkarni 
Tata Consultancy Services Research
Pune, Maharashtra
India

Sreedhar Reddy 
Tata Consultancy Services Research
Pune, Maharashtra
India

Tony Clark 
School of Engineering & Applied Science
Aston University
Birmingham, UK

Henderik A. Proper 
TU Wien
Vienna, Austria

ISSN 1867-8920 ISSN 1867-8939 (electronic)
The Enterprise Engineering Series
ISBN 978-3-031-29052-7 ISBN 978-3-031-29053-4 (eBook)
<https://doi.org/10.1007/978-3-031-29053-4>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

Preface

A future enterprise will be a complex ecosystem (or system of systems) that operates in a dynamic uncertain environment. It will need to continue delivering the stated goals while dealing with unforeseen changes along multiple dimensions such as events opening up new opportunities or constraining the existing ones, competitor actions, regulatory regime, law of the land and technology advance/obsolescence. Customers increasingly demand highly personalized services and user experiences that can change depending on market trends. The goals that drive the operation of an enterprise can change over time. Businesses are increasingly regulated. Given the increased dynamics, existing regulations will keep changing frequently, and new regulations will get introduced at a faster rate. Responsive compliance management with minimal exposure to risk will therefore be a universal key requirement that will be felt increasingly acutely across business domains. Increasingly, enterprises are pledging to the sustainable development goals proposed by the UN. Quite a few domains are witnessing stiff competition from new entrants such as FinTech companies in banking. Enterprises need to significantly reduce the costs to continue to be viable in the face of this technology-centric agile competition. Moreover, as the Covid-19 pandemic has revealed, enterprises need to be prepared to quickly adapt in the face of black swan events.

These dynamics will play out equally significantly across the three planes of an enterprise: *intent* dealing with the purpose of the enterprise, leading to its goals and associated strategies; *process* dealing with the operationalization of the strategy in terms of business processes, roles and responsibilities; and *organizational* dealing with the organization of the socio-cyber-physical actors and infrastructures that enact the different processes. Changes may originate in one plane and ripple through to the other planes. Given the increased rate of change, the time window available for bringing the three planes back in sync will continue to shrink. This will put hitherto unseen demands on enterprises, namely, responsive decision-making, in the face of uncertainty and swift adaptation so as to support continuous transformation without compromising on certainty. This calls for integration of “running the business” and “changing the business” objectives, leading to continuous enterprise engineering.

Enterprises have always dealt with change. While they do leverage technology, such as enterprise modelling and decision-making techniques, there is heavy reliance on human expertise and past experience. Large size, system of systems nature and increasing dynamics seem to have stretched this practice to its limits. Incomplete information and inherent uncertainty coupled with the law of bounded rationality further exacerbate the problem. As business becomes more dynamic with rapidly evolving goals of stakeholders accompanied by unforeseen and uncertain environment factors, the supporting software systems too need to dynamically evolve in a uniquely non-invasive way so as to meet the requirements and goals of every stakeholder. With increasing pervasiveness of software, this need will be felt ever more acutely and across all business verticals. While considerable progress has been made by [self-]adaptive software community in meeting the adaptation needs pertaining to executing machinery, there is little work reported on the adaptation of business functionality and processes. However, as knowledge driving the adaptation stays static, the enterprise tends to continue slipping further over time.

The book argues that challenges faced by an enterprise defined above are rooted in uncertainty. At any given time, the knowledge that an enterprise has about its environment is termed its knowledge boundary. Given that this boundary is limited, can never be complete and may be inconsistent with reality, it is not always possible to know that the strategy or operation is correct. Furthermore, given the scale and complexity of an enterprise, it is not always possible to define the correct behaviour or even know the effect of combining the behaviours of complex subsystems. Uncertainty arises from constant changes in the environment, regulations, policies and resources. Uncertainty can be addressed through making an enterprise intelligent.

An intelligent enterprise uses knowledge-based learning in various ways to improve the performance or to adapt. Traditionally, such knowledge and learning has been based around human experts who are in control of various subsystems and use levers to improve performance and adapt to changes. However, relying on humans in this way is expensive, difficult to quality assure and introduces a knowledge management problem in terms of staff retention.

An AI-Enabled Enterprise uses technology to implement the knowledge-based learning that enables an enterprise to exhibit intelligence. The technology allows the intelligent enterprise to scale up by reducing the reliance on human experts. Learning is achieved by monitoring the behaviour of the enterprise and its environment in order to take fully or partially automated operating decisions. This learning is augmented by continuously monitoring external knowledge sources too.

The book discusses the key challenges that need to be overcome in achieving AI-Enabled Enterprises, namely, the role of Digital Twin(s) in evidence-backed design, enterprise cartography that's far beyond process mining, decision-making in the face of uncertainty, software architecture for continuous adaptation and democratized knowledge-guided software development together enabling coordinated and coherent design of a

continuously adapting enterprise. For each challenge, the book proposes a line of attack along with the associated enabling technology and illustrates the same through a near-real-world use case.

Pune, Maharashtra, India
Pune, Maharashtra, India
Birmingham, UK
Vienna, Austria

Vinay Kulkarni
Sreedhar Reddy
Tony Clark
Henderik A. Proper

Contents

1	The AI-Enabled Enterprise	1
	Vinay Kulkarni, Sreedhar Reddy, Tony Clark, and Henderik Proper	
2	Decision-Making in the Face of Uncertainty	13
	Vinay Kulkarni	
3	Regulatory Compliance at Optimal Cost with Minimum Exposure to Risk	35
	Vinay Kulkarni	
4	Continuously Adapting Software	57
	Tony Clark	
5	Democratized Hyper-automated Software Development	85
	Sreedhar Reddy	
6	Coordinated Continuous Digital Transformation	101
	Henderik Proper and Bas van Gils	
7	A Case Study: Wellness Ecosystem	121
	Vinay Kulkarni and Sreedhar Reddy	

About the Authors

Tony Clark is a Professor of Computer Science and Deputy Dean in the College of Engineering and Physical Sciences at Aston University. He has experience of working in both academia and industry on a range of software projects and consultancies. His current interests are using adaptation and model-based techniques to create Digital Twin(s). Further information can be found at <https://research.aston.ac.uk/en/persons/tony-clark>.

Bas van Gils is a driven and experienced consultant, trainer and researcher. In the last few years, he has helped professionals and organizations in realizing their digital aspirations: from strategy to realization. The core disciplines in his work are (1) digital transformation, (2) enterprise architecture and (3) data management. He has worked in different industries, both in Europe and the United States, as a teacher and consultant. His academic work (teaching, research) is grounded in the scientific community at Antwerp Management School. Further information can be found at <https://www.linkedin.com/in/basvg/>.

Vinay Kulkarni is Distinguished Chief Scientist at TCS Research where he heads Software Systems and Services Research. His research interests include enterprise Digital Twin(s), learning-native software systems, multi-agent systems, model-driven software engineering and enterprise modelling. At present, exploring feasibility of imparting learning-aided adaptation to enterprises at strategy, process and system level through use of modelling, simulation and analytics. The vision is to integrate modelling, AI and control theory to support the dynamic adaptation of complex systems of systems using Digital Twin(s). Further information can be found at <https://in.linkedin.com/in/vinayvkulkarni>.

Henderik A. Proper, Erik for friends, is a Full Professor in Enterprise and Process Engineering in the Business Informatics Group at the TU Wien. His general research interest concerns the foundations and applications of domain modelling in an enterprise context. He has experience of working in academia and industry. Further information can be found at <https://www.erikproper.eu/about.html>.

Sreedhar Reddy is a Distinguished Chief Scientist at TCS Research. His research interests include model-driven engineering, databases, knowledge engineering, natural language processing and Machine Learning. His recent work includes development of a computational platform to support decision-making in the manufacturing industry using modelling and simulation, knowledge and Machine Learning. His current interest is in intelligent software systems that bring together knowledge, learning and Digital Twin(s) to intelligently adapt to changes in their requirements, goals and operating environments. Further information can be found at <https://www.linkedin.com/in/sreedhar-reddy-96a83326/>.



The AI-Enabled Enterprise

1

Vinay Kulkarni , Sreedhar Reddy , Tony Clark ,
and Henderik Proper 

Motivation

A future enterprise will be a complex ecosystem (or system of systems) of socio-cyber-physical actors that operates in a dynamic uncertain environment. It will need to continue delivering its goals while dealing with unforeseen changes along multiple dimensions such as customer needs, competitor actions, regulatory regime, law of the land and technology advance/obsolescence. Also, goals themselves may change with opening up of new opportunities, constraining the existing ones, disruptions like pandemics, etc. These dynamics will play out equally significantly across the three planes of an enterprise:

- *Strategy* dealing with the purpose of the enterprise leading to its goals and associated strategies
- *Process* dealing with operationalization of the strategy in terms of business processes, roles and responsibilities
- *Systems* dealing with the automation of business processes to the extent possible involving socio-cyber-physical actors

Important considerations at the system plane are the assignment of work to actual socio-cyber-physical actors and the needed balance between the work that can be done by

V. Kulkarni (✉) · S. Reddy

Tata Consultancy Services Research, Pune, Maharashtra, India

e-mail: vinay.vkulkarni@tcs.com

T. Clark

Aston University, Birmingham, UK

H. Proper

TU Wien, Vienna, Austria

computer-based actors and what can best be done by human actors. Another consideration is the trade-off involving multiple factors, such as efficiency, predictability, health (robots can be sent to places where humans should not go), ethics, etc.

Changes may originate in any of the enterprise planes and ripple through to the other planes. Given the increased rate of change, the time window available for bringing the three planes back in sync will continue to shrink. This will put hitherto unseen demands on enterprises, namely, responsive decision-making in the face of uncertainty and swift adaptation so as to support continuous transformation without compromising on certainty. This need will be felt along all three planes of the enterprise, thus necessitating that corresponding software be adaptive “by design”. This calls for new software architecture that supports adaptation as a “first class” concern. These new needs coupled with increasing pervasiveness of software will lead to an exploding demand for software. Traditional software development processes rely on large teams of skilled workforce. This approach is increasingly found wanting due to the large cycle times involved and severe shortage of skilled workforce. Keeping pace with rapidly advancing implementation technologies further exacerbates this problem. As a result, there is a need for reducing the current heavy dependence on developers possibly by enabling business domain experts play a greater role in software development. Rapidly advancing AI technologies coupled with model-driven engineering can be exploited to address this need.

With enterprises getting increasingly regulated, new regulations need to be complied with, and these regulations themselves will keep changing due to the enhanced dynamics. As a result, there is a need to continually stay compliant at optimal cost with minimal exposure to risk.

Moreover, enterprises are no more islands; instead, they are fast evolving into an ecosystem with complex interdependencies, thus further accentuating the need for rapid adaptation along multiple dimensions. The adaptation needs to ensure that objectives of all stakeholders are achieved in a balanced manner.

Current State

Typically, enterprise devises its strategies assuming a largely static environment. These strategies are then implemented using a set of business processes which in turn are automated using software systems. Both business processes and software systems are designed to deliver fixed set of goals while operating in a largely static environment.

Even in static world assumption, some questions remain inadequately addressed, for instance, “Is the bar set too low as regards the goals?”, “Is the strategy optimal?”, “What’s the optimal process?”, “How to effect the right adaptation?” and so on.

With the world becoming more dynamic, these questions recur frequently and become even more difficult to address. For example, these questions need to be suitably addressed all over again with changes in competitor behaviour, customer needs, regulatory regime, etc.

Figure 1.1 depicts this current state.

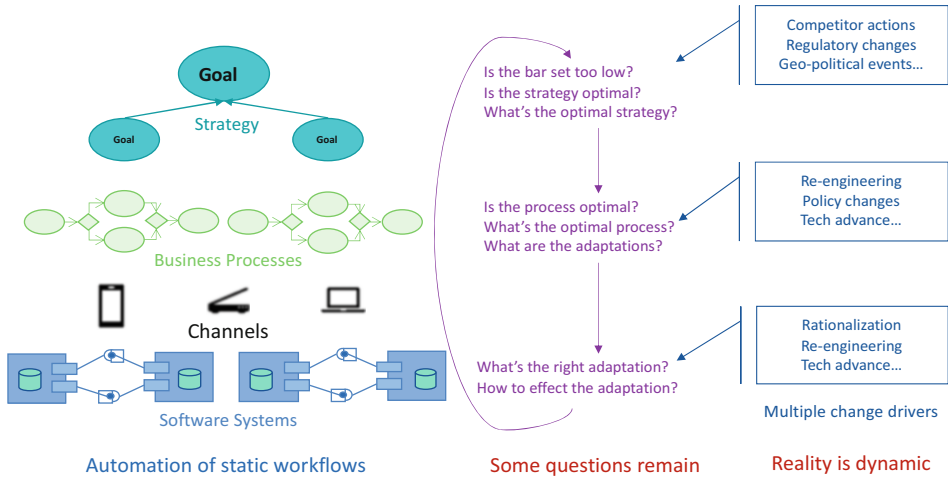


Fig. 1.1 Current state

Decision-Making in the Face of Uncertainty

Current practice relies on the following decision-making approaches:

Optimization Here, the idea is to formulate decision-making as an optimization problem solved using rigorous mathematical techniques to arrive at the desired behaviour. This approach requires possible behaviours of the enterprise and its operating environment to be known a priori and expressed in pure analytical terms. However, this is possible only when the behaviour is governed by laws of nature, physics, thermodynamics, chemistry, etc., for instance, controlling the boiler of a captive power plant [1], scheduling the crude arrival and mixing for a refinery [2]. Complexity and uncertainty arising from issues such as human behaviour, cyber-physical interactions and communications with a changing environment lead to an enterprise exhibiting emergent behaviour that is difficult or impossible to represent in terms of mathematical equations.

Machine Learning Build statistical models from past data of enterprise. The model is then used to answer questions relating to issues such as:

- The reasons for specific desirable or undesirable behaviour
- The results of applying a specific perturbation to the enterprise
- Is there a more effective way to achieve the stated goals?

In the light of recent advances in Machine Learning, this approach holds a lot of promise. The approach relies on two key conditions:

- Past data used to learn a model must be a representative set of all behaviours of the enterprise.
- The future behaviour of interest must be an extrapolation of the past as represented by the historical data.

Given these two conditions, Machine Learning can produce effective results [3, 4]. However, the intrinsic uncertainty and incompleteness of historical enterprise behavioural data mean the learnt model is likely to be incomplete and uncertain when faced with new situations. As a result, the decisions thus arrived can be sub-optimal at best and incorrect at worst.

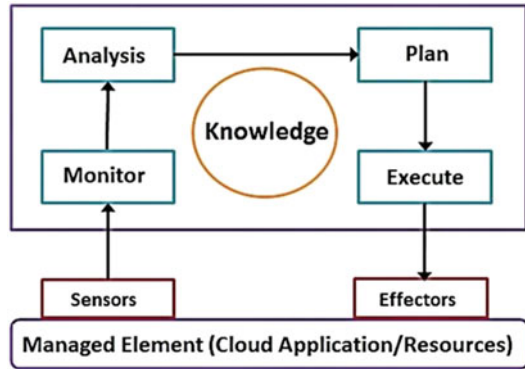
Human-Centric Traditionally, decision-making in large-scale enterprises has relied solely on human experts [5, 6]. While experts are able to come up with the right intervention in the localized context (i.e. a system or a subsystem they have expertise of), they find it difficult to justify what made them choose this intervention over other candidates [7]. Typically, experts are not able to state possible repercussions of an intervention on the rest of the system. Also, human experts are vulnerable to the law of bounded rationality [8].

Thus, it can be said that current practice falls short of effectively addressing the problem of decision-making in the face of uncertainty. A new approach seems called for. This is discussed in detail in Chap. 2.

Software Architecture for Continuous Adaptation

Just identifying the right decision (policy or strategy) is not enough. It also needs to be implemented effectively. Given the pervasiveness of software in enterprises, implementation of the decision will mean modifying the relevant set of software systems across the three planes. In order to achieve this, software must be designed to adapt to unforeseen changes along dimensions such as functionality, business processes, technology infrastructure, user experience, goals and operating environment. Moreover, it should do so while being aligned with the enterprise goals and conforming to internal policies and external regulations.

Considerable progress has been made by [self-]adaptive software community [8–10] to achieve software adaptation at run-time, e.g. MAPE-K architecture shown in Fig. 1.2. Several conceptual architectures are proposed [11, 12] to support adaptation with respect to computing resources. However, there is little work reported on adaptation of business functionality and processes that scales to effectively address real-life industry problems. Moreover, state of the art of [self-]adaptive software only addresses *known knowns* and *known unknowns* leaving out *unknown knowns* and *unknown unknowns*. Clearly, a new architecture to support continuous adaptation is needed which is discussed in detail in Chap. 4.

Fig. 1.2 MAPE-K architecture

Automated Compliance with Minimal Exposure to Risk

Businesses are increasingly regulated. Regulatory compliance is a board-level concern and one of the top-3 CEO-level concerns across business verticals.¹ Regulatory compliance at optimal cost with minimal exposure to risk is a critical need faced by modern enterprises across business domains. Failure to comply not only leads to heavy fines but also reputational risk. Enterprises need to be cognizant of compliance vs risk trade-off as exorbitantly high cost of compliance can make the enterprise unviable.² Given the increased dynamics, existing regulations will keep changing frequently, e.g. recent changes in Know Your Customer (KYC) regulation; and new regulations will get introduced at a faster rate, e.g. General Data Protection Regulation (GDPR). Responsive compliance management with minimal exposure to risk will therefore be a key universal requirement that will be felt increasingly acutely across business domains.

Effective compliance management requires legal, domain and IT expertise to come together in a coordinated manner facilitated by sophisticated tool support. With expertise already in short supply, document-centric compliance management tools such as GRC³ frameworks put heavy analysis and synthesis burden on human experts exacerbating the problem further. Lack of automation means these tools are vulnerable to cognitive limit and fatigue-induced errors of commission and omission. Given the increasing stress on regulated businesses, high dynamics and complexity, enterprises will want to be “compliant by design”. The current document-centric manual process of compliance management is turning out to be ineffective². Clearly, there is a need for new technology to help enterprises stay compliant with minimal exposure to risk in a responsive manner. Chapter 3 presents a solution to this need.

¹ <http://www.smbceo.com/2020/02/08/what-is-the-role-of-the-ceo-in-regulatory-compliance/>

² <https://www.bankingexchange.com/bsa-aml/item/8202-cost-of-compliance-expected-to-hit-181bn>

³ <https://www.ibm.com/learn/cloud/grc>

Democratized Knowledge-Guided Software Development

The increasing pervasiveness of large, complex enterprise-wide software systems and the need for dynamic adaptation will make new demands on software development life cycle (SDLC). Tools and technologies exist for the latter stages of the SDLC; however, the early stages (i.e. requirements elicitation, requirements engineering and design) continue to be document-centric and manual.

With software increasingly being used to drive growth (as opposed to bookkeeping) and to aid decision-making, the quantum of software development effort will significantly increase, and it is fair to conclude that there will not be a sufficiently trained workforce of software developers to meet this surge in demand. Moreover, development of this software will require reasonably deep domain knowledge which software developers are unlikely to possess.

Therefore, there is a clear need for new software development method and technology to co-opt domain experts into SDLC. It is safe to assume that a large part of the information required for SDLC will exist in semi-/unstructured form. Recent advances in Natural Language Processing (NLP) and Machine Learning (ML) provide a promising line of attack to extract requirements [13, 14] and domain knowledge from semi-/unstructured information sources into model and/or knowledge form [15, 16]. Moreover, this should take place continuously. Some of these challenges are being individually addressed by several communities. There is a need to integrate and build further upon them to come up with a new SDLC method and supporting toolset. Chapter 5 discusses in detail a line of attack to meet this need.

Continuously Adapting Software

A complex system such as an AI-Enabled Enterprise differs from standard systems in that it cannot be created with a single fixed behaviour. Such a system is complex and must either adapt to find an optimum behaviour or must modify its behaviour as the behaviour of its environment changes or the business goals are redefined. One way of achieving such adaptation is to embed a Digital Twin in the AI-Enabled Enterprise in such a way that the twin monitors the behaviour of the enterprise, compares the measurements against idealized behaviour and then issues controls in order to adapt the enterprise towards the required activity.

The creation of Digital Twin(s) for adaptive behaviour is problematic because of the scale and complexity of the behaviour that is to be monitored and controlled. Such behaviour (where it is known) can be viewed as a search space where the twin is responsible for exploration and navigation of the search space to ensure that an acceptable path through the system states is selected. For all non-trivial enterprises, such a state space is vast and cannot be completely navigated: approximation must be used.

Approximate solutions to finding a path through the search space exist. A typical example of such approximate solution technologies is *deep learning* implemented in the form of neural networks. While this approach produces results, the solution for encoding, fine-tuning and verification of the resulting deep learning network can be a problem due to the loss of information and the scale of the data that is processed.

This chapter provides an overview to the key features of using such a learning approach to achieving continuously adapting software and then shows how modelling and prototype simulation can be used to develop a complete but limited scale digital twin in order to understand the key features and gain confidence that appropriate adaptation will be achieved. This chapter introduces the TwinSim technology and demonstrates its features with respect to a range of case studies. Finally, the chapter provides an overview of a research roadmap for engineering Digital Twin(s) for adaptive software.

Coordinated Continuous Digital Transformation

The overall focus of this book is on the transformation of enterprises towards *AI-Enabled Enterprises*, involving a strong role for both AI and digital twin technologies. At the same time, for enterprises, the transformation towards AI-Enabled Enterprises is “just” a logical, albeit important, next phase in the continuous flow of digital transformations which enterprises are (and need to be) engaged in. Accordingly, in this chapter, we zoom in on both the challenges facing enterprises regarding digital transformations in general and the transition to AI-Enabled Enterprises in particular.

We will start by defining more precisely what we mean by digital transformation. We will see how these transformations have a profound impact on the structure of an enterprise. This also implies that it is important to ensure that such (enterprise) transformations are well-coordinated. What is even more challenging regarding the coordination of digital transformations is that the continual progress, and wide organizational impact, of digital technologies also implies that digital transformation should be seen as a continuous process.

Enterprise (architecture) models are traditionally regarded as an effective way to enable informed coordination and decision-making regarding enterprise (and digital) transformations. In line with this, this chapter will take a model-enabled perspective on the needed coordination. In doing so, we will argue for the need to identify (and manage) so-called enterprise design dialogues, where *enterprise models* are positioned as a key artefact in support of these *enterprise design dialogues*. Before concluding, we also review some of the challenges and opportunities towards future research.

The AI-Enabled Enterprise

The point solutions listed above are individually useful; however, they need to be integrated in order to address the scale of the challenges faced by modern enterprises. There is a clear need to develop a holistic approach, architecture, method and toolset so as to be able to use intelligence at all levels of an enterprise system that addresses its design, construction, deployment and adaptive maintenance.

An enterprise is:

A complex system of systems that can be viewed along three planes, namely, Intent, Process and Systems. An enterprise uses its systems to automate its processes in order to implement a strategy so as to achieve its stated goals. An enterprise must ensure that the strategy, processes and systems are valid, complete and consistent and must ensure that the various planes are commensurately updated in response to changes in a dynamic operating environment. An enterprise does not operate in isolation and must interact with its environment that includes *enterprise stakeholders*. An enterprise must maintain knowledge about its environment including the behaviour and motivation of its stakeholders which allows it to operate effectively.

The challenges faced by an enterprise in meeting the objectives stated above are rooted in *uncertainty*. At any given time, the knowledge that an enterprise has about its environment is termed its *knowledge boundary*. Given that this boundary is limited, can never be complete and may be inconsistent with reality, it is not always possible to know that the strategy or operation is correct. Furthermore, given the scale and complexity of an enterprise, it is not always possible to define the correct behaviour or even know the effect of combining the behaviours of complex subsystems. Uncertainty arises from constant changes in the environment, regulations, policies and resources.

Uncertainty can be addressed through making an enterprise intelligent:

An intelligent enterprise continuously learns in order to address the challenges arising from uncertainty. Learning may occur in order to ensure that the enterprise continuously improves the alignment of its behaviour with its strategic goals. Learning may be necessary in order to adapt the enterprise to changes that occur across any of its planes or in its environment. Learning will also be necessary in order to discover information at its knowledge boundary in order to interact more effectively with its environment.

An intelligent enterprise uses the learnt knowledge in various ways to improve the performance or to adapt. Traditionally, such knowledge and learning has been based around human experts who are in control of various subsystems and use levers to improve performance and adapt to changes. However, relying on humans in this way is expensive, is difficult to quality assure and introduces a knowledge management problem in terms of staff retention.

This leads to a proposal that uses Artificial Intelligence to automate parts of the learning and decision-making processes:

An *AI-Enabled Enterprise* uses technology to implement learning and knowledge-based reasoning that enables an enterprise to exhibit intelligence. The technology allows the intelligent enterprise to scale up by reducing the reliance on human experts. Learning is achieved by monitoring the behaviour of the enterprise and its environment in order to take fully or partially automated operating decisions. This learning is augmented by continuously monitoring external knowledge sources too.

Artificial intelligence techniques may be used in a number of ways with respect to an AI-Enabled Enterprise:

Knowledge Acquisition

An AI-Enabled Enterprise will rely on multiple data sources that must be processed and consolidated in order to create a repository of knowledge that can be used to predict the required behaviour, observe the consequences of actions, form a basis for intelligent decision support and control and adapt dynamic behaviour.

Decision Support An AI-Enabled Enterprise may rely on strategies that have been devised using knowledge-based decision support techniques. Such techniques may be used prior to system construction in order to decide on the correct operating policies or may be used dynamically to assist human experts in decision-making when a system needs to adapt or to address unforeseen situations. Digital Twin(s) may be used to perform in silico decision support activities that are then transferred to the enterprise.

Policy Development An AI-Enabled enterprise will follow a knowledge-based operating policy that allows it to reason dynamically about complex situations. For example, the policy may have been constructed using knowledge acquisition in order to construct a collection of rules to follow or may have been constructed using Machine Learning techniques applied to historic enterprise data. Digital Twin(s) may be used to create a policy by experimenting with various execution strategies or by creating synthetic execution histories from which policies can be learned.

Control An AI-Enabled Enterprise will employ knowledge-based control. This may take the form of a digital twin that uses domain knowledge to compare the current enterprise state with a “perfect state” and to reason about actions that are required in order to bring the enterprise back on track and in line with its goals. Control may also be achieved through the use of a knowledge-based policy that has been created statically or using dynamic Machine Learning techniques. Due to the complexity of the behaviour of a large-scale enterprise, it is likely that knowledge and control is decentralized.

Adaptation An AI-enabled enterprise will employ knowledge-based adaptation to ensure that it achieves its goals despite the operating policy for the enterprise being uncertain. Due to uncertainty, such an enterprise may be designed to have sub-optimal performance

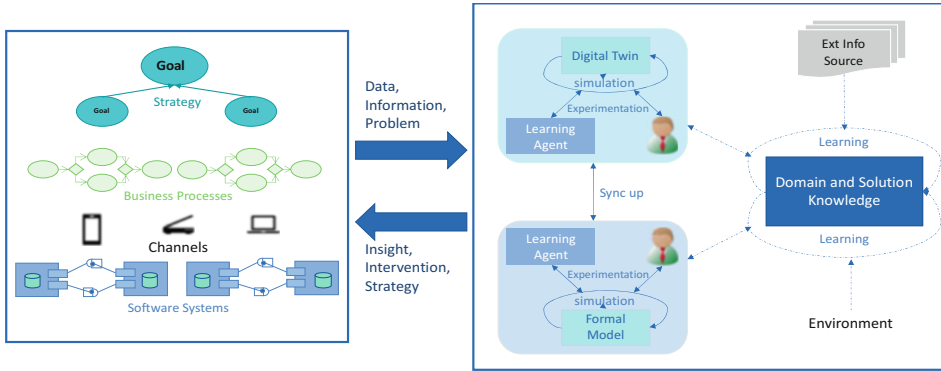


Fig. 1.3 An AI-Enabled Enterprise

initially and to use knowledge-based techniques in order to learn adaptation strategies that will increase the alignment with goals, policies, etc. Given the nature of large complex enterprises, it is likely that unforeseen situations will arise and an AI-enabled enterprise will use learning and domain knowledge to adapt to these unknowns in order to maintain goals. Furthermore, it is likely that goals may need to change and an AI-enabled enterprise will be able to gracefully modify its behaviour to achieve re-alignment.

AI-enabled enterprises are not limited in terms of the technologies that are used to achieve them. However, given the current state of the art in AI, the following is a list of techniques that might be employed across a range of AI-enabled enterprises: knowledge acquisition, rule-based systems, multi-agent systems, model-driven development, model checking, probabilistic reasoning, Machine Learning, deep neural networks, Bayesian networks, data analytics, pattern recognition, Digital Twin(s), control theory, decision support systems, adaptive systems, distributed AI, explainable AI, etc.

Figure 1.3 shows an example of an AI-enabled enterprise that uses a digital twin connected to a learning agent to compare the idealized behaviour of a formal model with the real-world behaviour of an enterprise. The formal model encodes the desired goals and operating policies of the enterprise and allows the real-world behaviour to be compared with an idealized behaviour. The comparison provides the learning agent with a dynamically evolving historical trace that is the basis of an intervention strategy for the enterprise.

Illustrative Example

We illustrate the AI-enabled enterprise through a prototypical consumer-producer ecosystem. Dynamic nature of the ecosystem demands an adaptive response from stakeholders to the changes in their environment such that goals of all stakeholders continue to be fulfilled over time even when the goals themselves change.

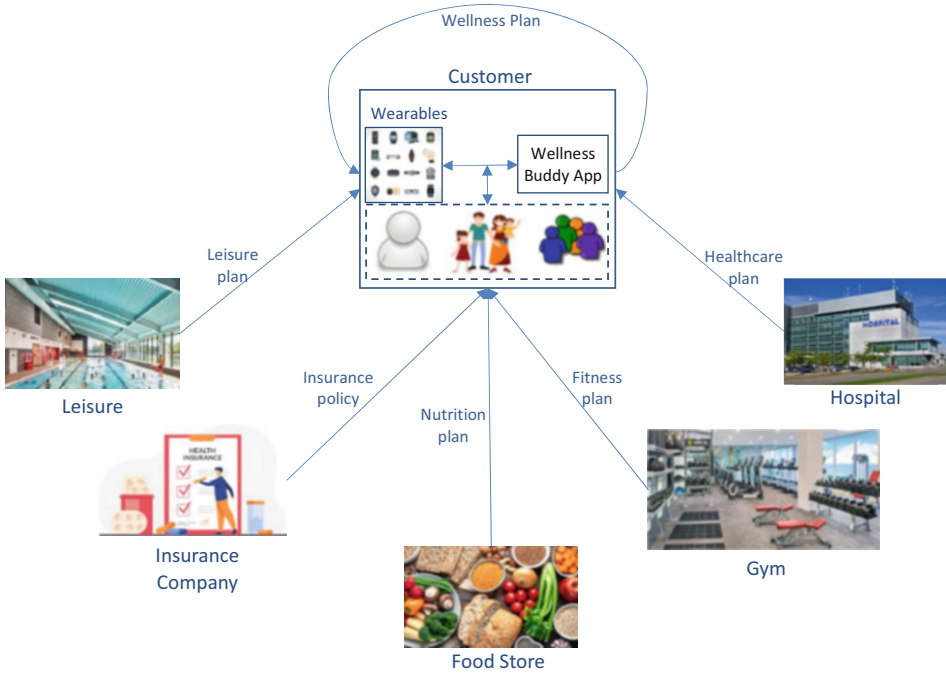


Fig. 1.4 Wellness use case

We present a use case revolving around the wellness needs of individuals and families serviced by a set of providers such as fitness centres, food stores, hospitals, insurance companies, resorts, etc. as shown in Fig. 1.4. Typically, the service providers have only partial information about their customers and competitors. They begin by classifying their clientele into a broad set of buckets and designing services keeping in mind prototypical customer for each bucket. Services thus designed end up delivering sub-optimal value to specific individuals in a bucket and also result in sub-optimal return on investment for the provider.

Today, wellness seeker needs to select the right set of services on the offer and stitch them together to construct a plan that meets his wellness needs. This need can be better served through a value integrator who has access to the relevant information about all stakeholders and can provide services to each stakeholder that help meet their needs.

In today’s fast-paced world where customer needs as well as competitor landscape keep changing rapidly, the services need to suitably keep pace with these changes. This calls for quick identification of correct response to a change and quick and effective implementation of the response. This need too can be better served through a value integrator that continuously monitors the changes to recommend suitable adaptations to the stakeholders.

Chapter 7 discusses this use case in detail.

References

1. Biswas, J., Kumar, R., Mynam, M., Nistala, S., Panda, A., Pandya, R., Rathore, R., & Runkana, V. (2017). *Method and system for data based optimization of performance indicators in process and manufacturing industries*. Patent No. US10636007B2.
2. Wagle, S., & Paranjape, A. A. (2020). Use of simulation-aided reinforcement learning for optimal scheduling of operations in industrial plants. *WSC*, 572–583.
3. Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). *Deep face recognition*.
4. Tüfekci, P. (2014). Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60, 126–140.
5. March, J. G. (1994). *Primer on decision making: How decisions happen*. Simon and Schuster.
6. Hutton, R. J. B., & Klein, G. (1999). Expert decision making. *Systems Engineering: The Journal of The International Council on Systems Engineering*, 2(1), 32–45.
7. Simon, H. A. (1990). Bounded rationality. In *Utility and probability* (pp. 15–18). Palgrave Macmillan.
8. Salehie, M., & Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(2), 1–42.
9. Lemos, D., Rogério, H. G., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., et al. (2013). Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II* (pp. 1–32). Springer.
10. Calinescu, R., Ghezzi, C., Kwiatkowska, M., & Mirandola, R. (2012). Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM*, 55(9), 69–77.
11. Oreizy, P., Gorlick, M. M., Taylor, R. N., Heimhigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D. S., & Wolf, A. L. (1999). An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems and Their Applications*, 14(3), 54–62.
12. Iglesia, D. G., La, D., & Weyns, D. (2015). MAPE-K formal templates to rigorously design behaviors for self-adaptive systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(3), 1–31.
13. Mu, Y., Wang, Y., & Guo, J. (2009). Extracting software functional requirements from free text documents. In *2009 International Conference on Information and Multimedia Technology* (pp. 194–198). IEEE.
14. Casamayor, A., Godoy, D., & Campo, M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology*, 52(4), 436–445.
15. Saxena, K., Patil, A., Sunkle, S., & Kulkarni, V. Mining heterogeneous data for formulation design. In *1st Multi-Source Data Mining (MSDM 2020)*. Workshop of 20th IEEE International Conference on Data Mining (ICDM 2020), 17–20 Nov 2020, Sorrento, Italy. Online Event.
16. Wadhwa, N., Sarath, S., Shah, S., Reddy, S., Mitra, P., Jain, D., & Rai, B. (2021). *Device fabrication knowledge extraction from materials science literature*. AAAI-IAAI.



Vinay Kulkarni 

Introduction

We live in a hyperconnected world characterized by complex system of systems that comprise a large number of interconnected socio-cyber-physical agents. These systems often need to change/adapt/transform in order to stay relevant while operating in a dynamic and uncertain environment where unforeseen changes can occur along multiple dimensions, such as regulations regime, technology advance/obsolescence, mergers and acquisitions, law of the land and black swan events like Covid-19 pandemic. These changes may open new opportunities or pose serious threats. Introducing the right intervention in the right system at the right time in response to a change is therefore critical in order to benefit from an opportunity or to mitigate a threat. The ever-shortening window of opportunity provides little room for later course correction.

A large number of these complex systems are techno-socio-economic systems where human actors, software systems and cyber-physical systems interact with each other in a complex manner. Typically, information about such large complex systems exists in a fragmented and distributed manner, and that too is rarely kept up to date. It is humanly impossible to get a holistic understanding of the complex system from these information fragments. Instead, the complex system is comprehended in terms of subsystems that are well-understood in isolation and interactions between these subsystems. While structure of these subsystems can be discerned in detail, understanding of the behaviour remains somewhat fuzzy as it does not conform to the laws of physics/chemistry/thermodynamics.

V. Kulkarni (✉)

Tata Consultancy Services Research, Pune, Maharashtra, India

e-mail: vinay.vkulkarni@tcs.com

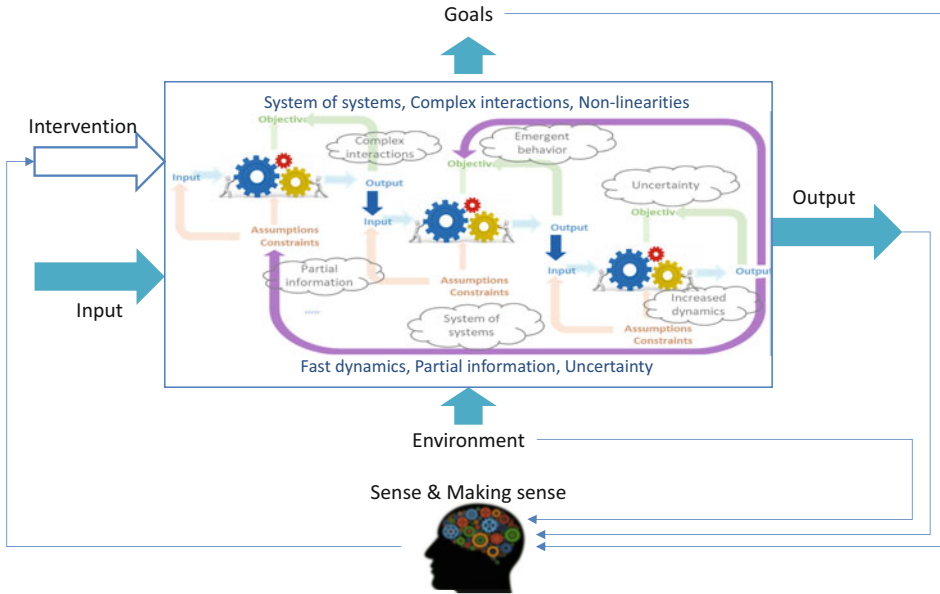


Fig. 2.1 Decision-making in complex systems

Deciding an effective intervention for a complex system (of systems) in response to a change in its environment (or goal) and introducing it within short opportune time is a difficult task. This requires a deep understanding of aspects such as structural decomposition of complex system into subsystems, relationships between these subsystems and emergent behaviour of systems and subsystems. The scale of complex system, its socio-technical characteristics and the dynamic and uncertain nature of its operating environment make decision-making a challenging endeavour.

For decision-making, the system can be viewed as a transfer function from *Input* value space to *Output* value space as shown in Fig. 2.1. Designing a suitable transfer function is a challenging task in itself. The transfer function needs to be continually modified in response to changes in the environment and/or goals which is further exacerbated when the available information about the system is incomplete and distributed in fragments and the environment is uncertain. This in a nutshell is the decision-making problem.

Current Practice

Three broad approaches are used for decision-making in the face of uncertainty. They are (1) treating the decision-making problem as an optimization problem, (2) model-based approaches and (3) expertise-driven human-centric approaches.

Decision-Making as an Optimization Problem

This approach works best when system behaviour is known a priori, the desired goals are defined precisely and system behaviour is amenable to precise analytical specification that's capable of rigorous analysis using automated means. System behaviour is specified in terms of a system of equations such as ordinary or partial differential equations. The goals constitute a computable objective function. Technique such as Linear Programming [1] can be used to solve the system of equations with the objective function as maximization or minimization criterion. This approach can produce globally optimal solution modulo problem formulation.

This approach to decision-making works well for physical systems such as power plants, boilers, assembly lines, etc. However, it is found inadequate for techno-socio-economic systems where the system behaviour is not known (rather, knowable) a priori and cannot be specified in purely analytical terms. This approach begins to lose effectiveness when system behaviour is probabilistic and subject to high dynamics. The same is true when system goals change over time.

Model-Based Decision-Making

This is arguably the most widely explored approach to decision-making. In essence, this is a quantitative approach [2] that involves precise interpretation of system data, structure and behaviours. The quantitative approach is further classified into three categories: (1) inferential approach, (2) experimental approach and (3) modelling and simulation approach [3].

The inferential approach [4] analyses the existing system data (i.e. trace or historical data) to infer the characteristics of a system or an enterprise. This approach is effective when the operating environment is static.

The experimental approach analyses the system by manipulating the system variables and observing their effects on system output in a controlled environment. This approach is often infeasible or not an economical option for large business critical enterprises.

The modelling and simulation approach imitates the system using a purposive model, explores a range of scenarios by simulating the possible interventions to be incorporated into the model and interprets the simulation results to check whether the desired goals are met. This approach exists in several avatars based on how the model is constructed (i.e. top-down or bottom-up) [5], kind of the model (i.e. data-centric or domain-centric) and nature of the modelling language (i.e. rigorous or enterprise).

Top-down approach models the system as a whole and adopts the reductionist view to decompose it into parts that can further be decomposed ad infinitum till reaching the granularity where the part is either atomic or understood completely. The key concerns with top-down approach are that it can't support emergent behaviour and needs to have complete information about the whole system a priori. As a result, top-down approach typically fails for large systems.

Bottom-up approach starts from atomic or well-understood parts and arrives at a holistic view of a system through composition. While bottom-up approach is capable of observing emergent behaviour, it is found wanting in representing complex structure and handling uncertainty.

Data-centric approach uses statistical models (e.g. ARIMA [6]), Artificial Intelligence (AI) and Machine Learning (ML) techniques to analyse the historical data to identify the appropriate interventions, i.e. Lever. This approach is extremely effective for systems for which comprehensive and relevant data is available. However, this exclusively data-reliant approach works only when the available data represents all possible behaviours of the system and the future is linear extrapolation of the past.

Domain model-centric approach, in contrast, represents the transfer function and environment of the system using a variety of analysable models that can broadly be classified into two types: mathematical models and enterprise models (EM). Mathematical models, such as linear programming [1] and integer programming [7], can specify the complex system of systems in terms of lumped-up mathematical formulae, thus enabling the formulation of decision-making problem as a multi-variate optimization problem. However, this approach is vulnerable to data inadequacy, i.e. survival bias [8]. Over the years, Simulink and MATLAB are extensively used for modelling and analysing systems where relevant data is inadequate, domain understanding is less ambiguous and objective function is well formed, i.e. devoid of conflicting goals. These techniques can be used to derive optimal solution in a local context (i.e. for a system within a system of systems). However, they are found inadequate in predicting system-wide ramifications of introducing intervention in a locality and hence incapable of guaranteeing global robustness of the solution.

Enterprise Models (EMs) fall into two broad categories, top-down coarse-grained and bottom-up fine-grained, with the former outnumbering the latter [9]. Enterprise models are spread across a wide spectrum. For example, some focus on capturing the enterprise in a manner that's amenable for human-centric analysis, e.g. Zachman Framework [10]; some provide sophisticated graphical representation of enterprise that's also somewhat amenable to automated analysis, e.g. ArchiMate [11]; and some support machine interpretable and/or simulatable specifications that help analyse a range of system aspects, for instance, BPMN [12] for modelling the process aspect, i* [13] for modelling enterprise goals and system dynamic [14] model for modelling enterprise behaviour in an aggregated form. The multi-modelling and co-simulation environments, such as DEVS [15] and MEMO [16], demonstrate further advancement that supports the analysis of multiple aspects. The bottom-up fine-grained EMs start from the parts or micro-behaviours and derive macro-behaviour of a system through composition, e.g. Erlang [17] and actor [18].

In summary, coarse-grained models are good for macro-level analysis but fail to capture the notion of system of systems characterized by conflicting goals, individualistic behaviour of fine-grained units and emergent behaviour. While fine-grained models are useful in supporting a constructionist view of modelling complex system of systems, they often fail to scale so as to be able to analyse large business and social systems. The state-of-

the-art enterprise modelling techniques are capable of addressing situations that are characterized by limited uncertainty and limited number of conflicting goals; however, they are found wanting on uncertainty and trade-off dimensions [19].

Human-Centric Decision-Making

The qualitative approach [20] is concerned with the subjective assessment of the complex system through a range of management techniques such as interviews, discussions and field studies. The state of the practice of organizational decision-making [21] is predominantly qualitative where decision-makers intuitively analyse various performance indicators, i.e. Measures, compare them with the desired Goals and reflect on their experience to arrive at the required change/interventions. Business systems try to mitigate the risk of such intuition-driven decisions by enabling controlled experimentation in a sandbox environment, i.e. A/B testing [22]. However, any experimentation involving real system is a time-, effort- and cost-intensive endeavour requiring iterations over multiple unsuccessful attempts before reaching a “good-enough solution”. Enhanced dynamics, shortening window of opportunity and increasing uncertainty are making this intuition-based ideate-build-experiment approach risky and ineffective.

Thus, it can be said that current practice of decision-making in the face of uncertainty seems inadequate.

Solution

We envisage a line of attack that borrows proven ideas from modelling and simulation, control theory and artificial intelligence and builds upon further to be able to use them in an integrated manner as shown in Fig. 2.2. At the heart of this line of attack is the concept of enterprise digital twin – a virtual high-fidelity machine-processable representation of enterprise. It is amenable to solution space exploration through what-if/what-if analysis. Thus, it can be used as an “in silico” experimentation aid where experts subject the digital twin to a variety of perturbations. As the digital twin is a high-fidelity representation of the system, its response to a perturbation is in the ballpark of actual system response. Experts can interpret this response in the light of their knowledge and experience to arrive at a candidate set of suitable interventions. The set can be validated for correctness and efficacy using the digital twin itself by running appropriate simulations, thus leading to the identification of the most suitable intervention. Thus, digital twin considerably reduces the need for real-life experimentation with the system and leads to significant savings in time, cost and effort.

Though highly useful as an “in silico” experimentation aid, the digital twin does not reduce intellectual burden on human experts. To this end, we use an AI technique known as Reinforcement Learning [23]. Basically, we use digital twin as an “experience generator”

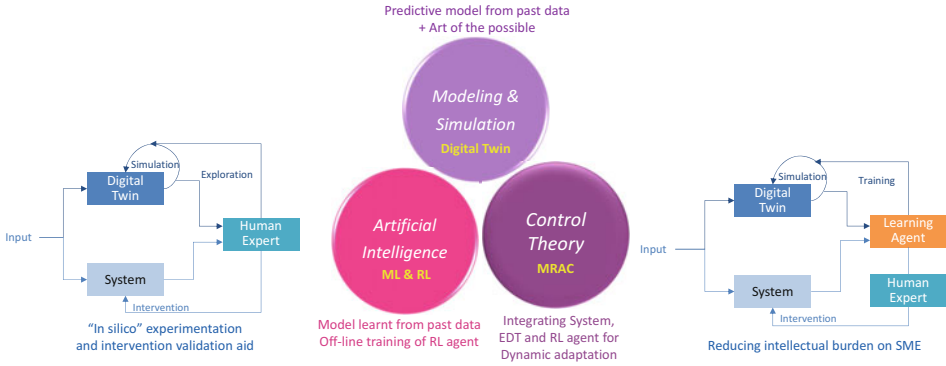


Fig. 2.2 Digital twin-centric simulation-based approach to decision-making

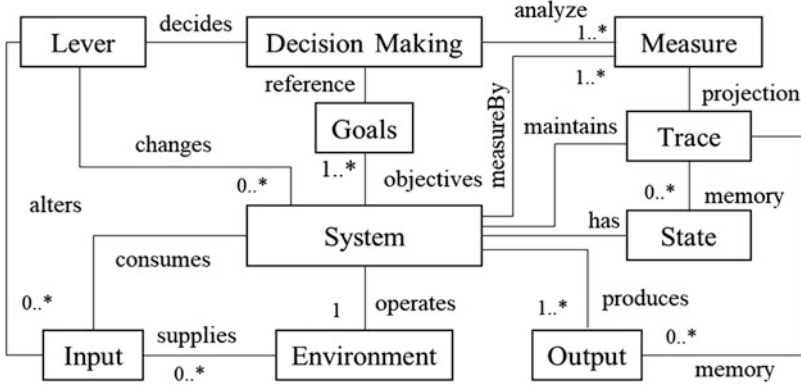


Fig. 2.3 Decision-making meta-model

from which the Reinforcement Learning agent (RL agent) learns what action to perform when to achieve the overall objective. We bring the system, digital twin and RL agent together in an adaptation architecture based on the Model Reference Adaptive Control (MRAC) paradigm [24]. We have ideas to extend this architecture to support dynamic adaptation where even the goals can change over time.

Decision-Making Meta-Model

Figure 2.3 depicts a meta-model for decision-making. A system can be viewed as a transfer function from *Input* value space to *Output* value space. A *Measure* constitutes possibly derived projection of interest on output. System exists to meet the stated *Goals* while operating in a dynamic and uncertain *Environment* that may constrain inputs and/or system

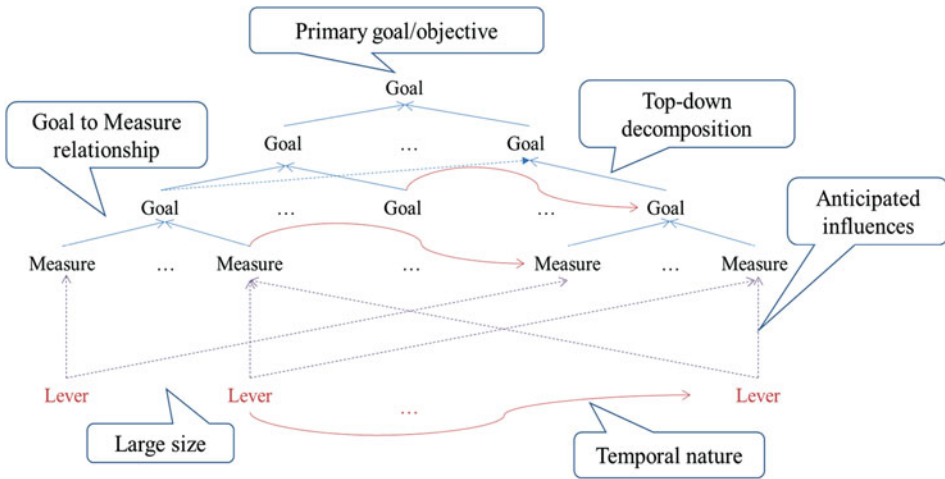


Fig. 2.4 System goals, measures and levers

behaviour. System goal is an objective function over output value space and system execution *Trace* when goal has temporal characteristics. Moreover, the goals can have a complex decomposition structure with inter-dependent and cross-cutting goals as shown in Fig. 2.4. The former imposes an order on meeting the goals, and the latter usually leads to trade-offs. The *Levers* represent a set of candidate *interventions* that can be introduced to nudge the system. Applying a lever brings about a discernible change in system output or a derivation thereof of interest, i.e. a *Measure*. Goal is an objective function on measures and thus computable. A goal, after evaluation, can be viewed as a measure. Thus, an iterative process of *analyse measures* \rightarrow *identify lever* \rightarrow *apply lever* \rightarrow *check satisfiability of goal* helps traverse the goal decomposition structure (of Fig. 2.4) in a bottom-up manner. This could be a possible solution to the decision-making problem.

This process is typically carried out with the real system albeit in a sandbox environment. As a result, it's a time-, cost- and effort-intensive endeavour relying extensively on human experts for the design of experiments as well as analysis and synthesis of results of the experiments. Moreover, time constant of this iterative process is rather large, thus putting a question mark on its efficacy when the system is operating in a dynamic environment. Supporting a different manifestation of the iterative process is clearly a critical need. This is where Digital Twin(s) can help.

Digital Twin

Digital Twin (DT) is a purposive virtual high-fidelity representation of a complex system of systems that is amenable to rigorous quantitative analysis through what-if and if-what

scenario playing. Thus, Digital Twin(s) can be helpful in supporting the “in silico” realization of the iterative process, i.e. *analyse measures* → *identify lever* → *apply lever* → *check satisfiability of goal*.

A purposive DT for supporting quantitative decision-making models the complex system as a set of intentional autonomous interacting agents. Each agent operates to achieve its own goals (i.e. intentional) in a pro-active manner (i.e. autonomous) wherein it may use other agents (i.e. interacting). These interactions could be collaborative or competitive or co-opetitive. Typically, majority of the interactions are collaborative (or sometimes co-opetitive) for a complex system driven by a single goal. For instance, while Marketing and Products units collaborate to deliver better sales, they also compete for resources. Ecosystems such as markets exhibit competitive interactions. Here, the desirable fixed point of decision-making process is pareto-optimality [25] if not Nash equilibrium [26].

An agent observes the environment, makes sense of the observations and performs actions to achieve its objectives. Knowledge and reasoning play a role in making sense of observations. If this knowledge is uncertain, then the inferences/conclusions drawn from the observations have a stochastic nature. It is these inferences (i.e. sense made of the observations) that determine the action to be performed. The action could be changing the local state of agent or sending a message to other agents. These actions can be stochastic to model uncertainty. Agents can exist at different levels of granularity, i.e. an agent can be a composition of a set of next-level agents as shown in Fig. 2.5.

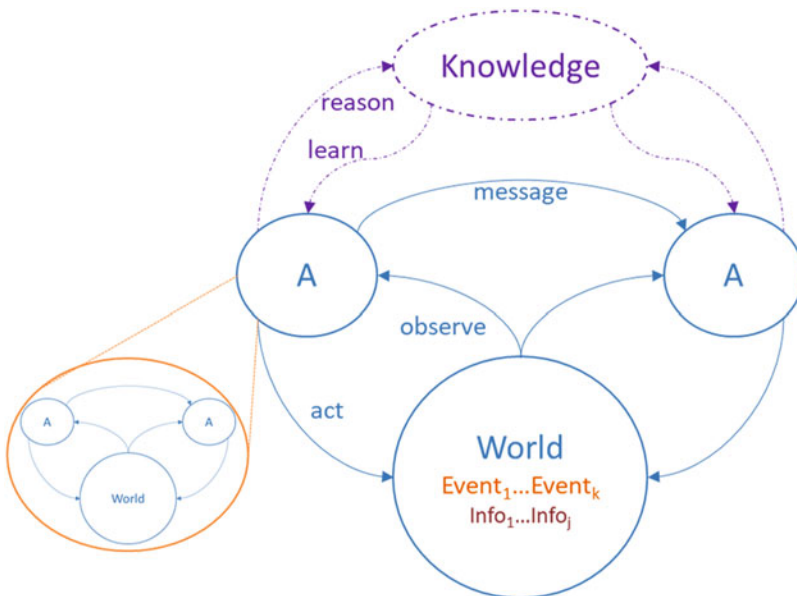


Fig. 2.5 Composable smart agent

An agent can adapt its behaviour in response to the changes in its environment. Essentially, an agent has a set of situation-specific behaviours, and it can switch from one behaviour to another depending on the situation it finds itself in. An agent adapts its behaviour not only to achieve local goals but also to help achieve system goals. Augmented with knowledge acquisition machinery, an agent can also learn new behaviours in a human-in-control manner.

Such Digital Twin(s) can serve as an “in silico” experimentation aid to support data-driven evidence-backed decision-making in the face of uncertainty.

“In Silico” Experimentation Aid for Decision-Making

Digital Twin(s) support a knowledge-guided tool-assisted approach to decision-making for complex systems as shown in Fig. 2.6. The key tenets of the approach are:

Holding a mirror: Domain experts recreate the current state by subjecting the purposive digital twin to the same input (as that of real system) in an identical environment setting.

The simulated output helps explain why things are the way they are. In case the current state does not meet the desired goals, suitable intervention is necessary.

Analyse measures: Domain experts interpret the simulation results to ascertain if the goals are met. This activity may involve computing measures from the raw simulation results.

In essence, domain experts identify how far the current state is from the desired state.

Identify levers: Domain experts use the distance measure and their expertise to arrive at a candidate set of agent-centric interventions such as changing the behaviour of an agent

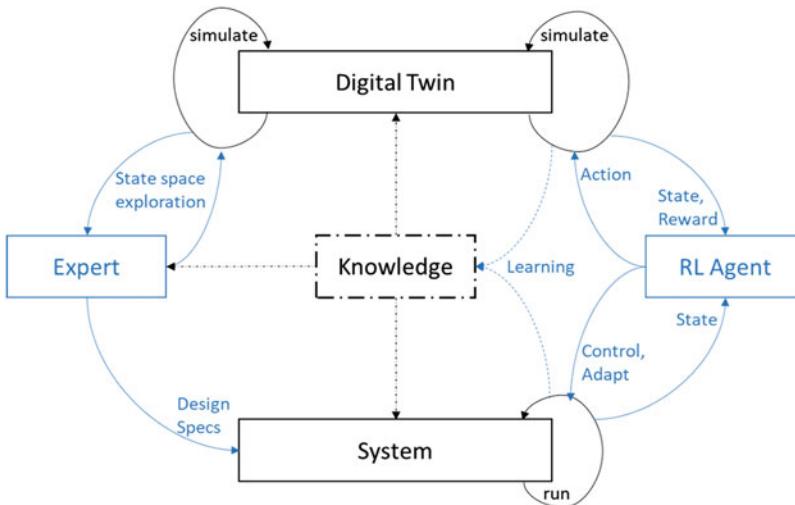


Fig. 2.6 Knowledge-guided decision-making aid

or changing the interaction between two agents. Any likely change in the environment is also explored.

Apply lever: The modified digital twin and environment are then subjected to appropriate what-if scenario thus in a new simulation run. The resultant output is examined for satisfaction of goals. If goals are not met, the process repeats till the candidate set of interventions is exhausted. If still unmet, the goals need to be appropriately attenuated, and the whole process is repeated.

Though highly useful as an “in silico” experimentation aid, the digital twin does not reduce the intellectual burden on human experts. To this end, we use Reinforcement Learning (RL) techniques to train a controller through interactions with the digital twin that acts as an “experience generator”. Also, domain knowledge can help guide solution space exploration by RL agent, thus leading to faster and better learning. The controller may nudge the system directly or through a human agent.

We now discuss technology support necessary to implement the proposed digital twin-centric approach.

Technology Infrastructure

Specification Language

We have developed an actor-based language called ESL [27] by extending the “actor” concept from [28] to specify the digital twin of a complex system (of systems) as shown in Fig. 2.7. We use the term “agent” synonymously with “actor”. ESL helps represent the

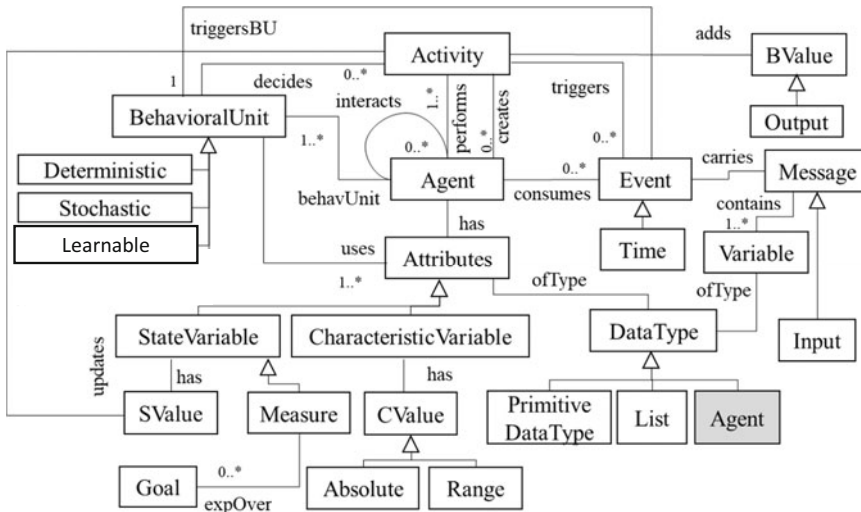


Fig. 2.7 Agent-based realization of digital twin

complex system as a set of intentional (i.e. there is a well-defined goal to be achieved) autonomous (i.e. capability to achieve the goal in a pro-active manner) composable (i.e. an agent can be realized in terms of a set of fine-grained interacting agents) agents. *Agent* listens to *Events* of interest and responds either by performing an *Activity* or sending a *Message* to some other agent. Agent behaviour can be *Deterministic* (i.e. there is only one Activity to perform in response to an event occurrence), *Stochastic* (i.e. one from the set of candidate activities is performed in response to an event occurrence – usually guided by a probability distribution) and *Learnable* (i.e. a model learnt from past data). Agent has a set of *Attributes* representing its state and special characteristics. *StateVariable* denotes the former and *CharacteristicVariable* the latter. *Values* of StateVariables are singleton, whereas those of CharacteristicVariables could be ranges. *Measure* can be a StateVariable or a computation involving StateVariables. *Goal* is an objective function over Measures. Agents might be hierarchically composed of agents. Environment is also modelled as an agent.

Thus, one can represent a complex system virtually as a digital twin which can be suitably initialized and where desired what-if simulations can be performed. In case the goal is unmet, domain experts identify the appropriate interventions based on expertise and experience. The *Levers* to introduce these interventions are change of CValues, change of Activity and change of probability distribution over Activities.

DT Construction

Construction of purposive digital twin for decision-making follows a top-down as well as bottom-up approach. Using a top-down approach, the complex system is successively decomposed till the behaviour of leaf-level subsystem is reasonably well-understood. These leaf-level subsystems are implemented as atomic actors having well-defined goals they aim to achieve. The next higher-level subsystems are implemented as composite actors. Actor composition relationship leads to dependency relationship between their respective goals. The bottom-up pass also reveals goal interference if any. This relationship can lead to trade-offs. Domain uncertainties are captured in the form of stochastic behaviour of actors.

The required domain information to be captured in a purposive DT comes from a wide spectrum of semi-/un-/structured sources including databases, execution logs, standard operational procedure notes, policy documents and understanding of domain experts. As a result, an army of experts from wide-ranging fields of expertise is required to manually create the digital twin specification – clearly a time-, cost-, effort- and intellect-intensive endeavour. The purposive meta-model serves as a lens to mine/author appropriate model fragments, corresponding to a view of the purposive meta-model, from these information sources. The meta-model also serves as an aid to integrate these model fragments, thus ensuring correctness and internal consistency. Knowledge of the problem domain and system helps construct the purposive digital twin from the integrated model. Figure 2.8 presents a pictorial overview of the framework we have developed for the accelerated creation of digital twin models from information available in semi-/un-/structured form

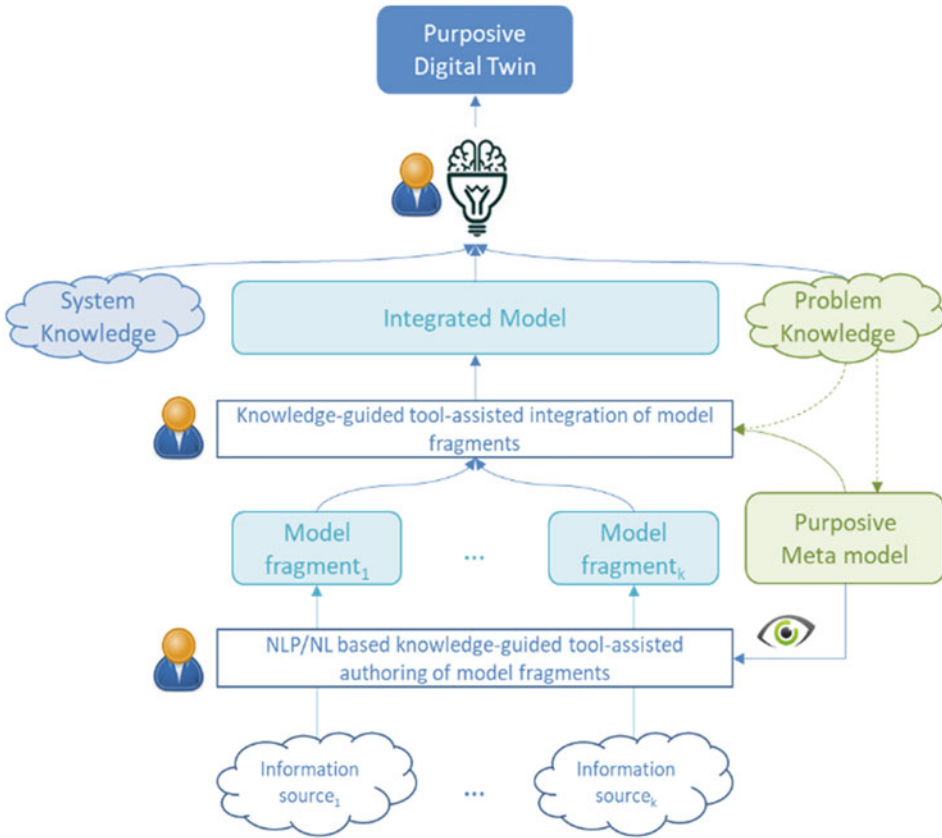


Fig. 2.8 Knowledge-guided tool-assisted construction of Digital Twin(s)

[29]. It comprises automation aids based on (1) Natural Language Processing (NLP) and Machine Learning (ML) techniques for gathering the desired information from a given information source, (2) meta-model-driven techniques for the integration and reconciliation of the model fragments and (3) model validation-based techniques for identifying the missing model fragments.

DT Validation

The utility and efficacy of a constructed digital are largely dependent on how rich it is in terms of analyses supported and how closely does it represent the real system. We provide two established ways of validating the digital twin, namely, conceptual validity and operational validity [30]. In conceptual validity, the domain experts certify how comprehensive the constructed actor topology is from domain perspective. We ensure operational validity through simulation wherein constructed digital twin is subjected to known past events leading to a simulation trace which is then examined to ascertain if the resultant behaviours are identical to the ones from the past. Our simulation engine generates rich

execution traces containing detailed information necessary for analysis. We have developed a pattern language to specify the desired behaviour and a pattern-matching engine to look for the specified patterns in the simulation trace [31]. This generic solution to ascertain correctness can be further augmented by manual validation of the input and output and control variables of the simulation.

Illustrative Real-World Applications

The proposed approach and supporting tools are validated for utility and efficacy on a set of real-world problems spanning a wide spectrum of business verticals. A representative sample is presented below.

Case Study from Telecom

Telecom space can be viewed along two broad dimensions, namely, physical system of telecom network infrastructure and business system that offers products to customers through various processes. We focused on the business system where the key objective is to assist Sales, Product and Customer Care heads to fulfil their respective goals using digital twin technology.

The principal goal of Head of Sales is to acquire as many right customers as possible for the existing product portfolio and customer care processes. The principal objective of Head of Products is to create and maintain a product portfolio that will best meet the communication needs of the existing (and prospective) customer base and customer care processes. The principal objective of Head of Processes is to help acquire new customers and retain existing customers at a minimum cost for the existing customer base and product portfolio.

Current practice is for the three heads to operate in silos to independently arrive at strategies aimed at achieving their stated goals. These strategies are evaluated in a sandbox environment using A/B testing. This is a time-, cost- and effort-intensive approach. Moreover, while the strategy could be the most suitable for achieving the local objective (i.e. Sales, Product or Process), there is no way to check if its introduction may have adverse ripple effect on the other two aspects. As telecom is arguably the most dynamic of business verticals, telecom service providers end up operating in a reactive (or catch-up) mode for most of the time with little a priori assurance of meeting the desired goals. As a result, this domain is characterized by high customer churn, long tail of inactive products, low customer satisfaction and poor Customer Lifetime Value (CLV).

We addressed several problems for a large CSP having tens of millions of customers, hundreds of active products and hundreds of processes. We constructed a purposive digital twin [32] that encapsulates the knowledge of Products, Processes and Individual customers (e.g. age, gender, station of life, early adaptor/laggard, etc.) leading to a good estimate of product and services expectations. A simulation-based iterative process helped us arrive at the right configuration of Products (features, launch strategy, price point, etc.), Processes

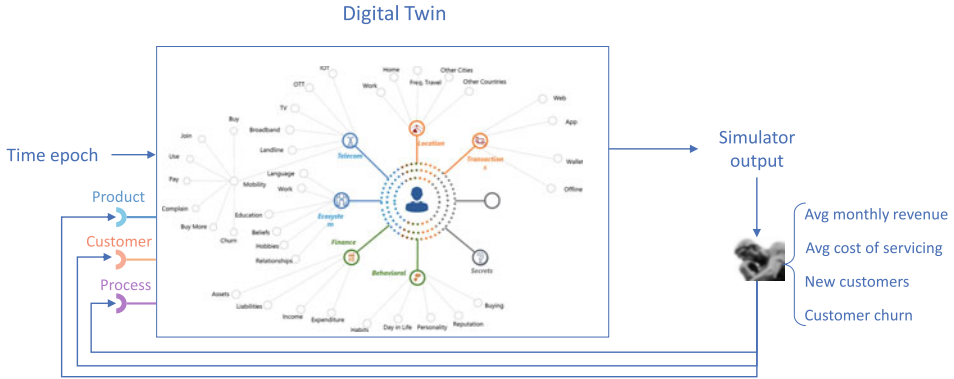


Fig. 2.9 Fine-grained digital twin for a large telecom service provider

(SLAs, technology investment, price point, etc.) and Customers (who will opt for which product at what price point, what's the best way to sell/upsell/cross-sell, how best to retain at what cost, etc.) as shown in Fig. 2.9. The measures we focused on were average monthly revenue, average cost of servicing customers, number of new customers added and number of customers left.

Specifically, we used this DT to solve these problems:

CLV optimization: The problem in a nutshell is as follows: Given the set of products and engagement services, what's the best Customer Lifecycle Value (CLV) possible for the given customer base? What changes need to be introduced in products and/or engagement services in order to further improve CLV? With capital expenditure and operating expenditure as non-negotiable constraints, we focused on a few products (e.g. international plan, family plan, etc.) to identify improvements in customer engagement and care to significantly improve CLV.

Better product launch: The telecom provider had come up with a launch strategy based on segment-wise experimentation promising 5.7% Take rate and 84% Retention rate. We used the fine-grained DT to come up with a product launch strategy promising 2.1% Take rate and 17% retention rate. In reality, Take rate was 2.5% and Retention rate was 16%. Furthermore, DT was used to fine-tune the product launch strategy, thus delivering 2× improvement.

Response to onset of Covid-19 pandemic: Onset of Covid-19 pandemic in February 2020 led to significant surge in demand for internet connectivity as the world shifted to work from home mode. Telecom service providers were keen on meeting this demand through *Unlimited Plan* (i.e. fixed pay regardless of use) products as opposed to *Metered Plan* (i.e. pay per use) products as the former have significantly less post-sell cost. There were a range of Unlimited Plan products each catering to a customer segment. Impact of Covid-19 pandemic on the communication needs of these segments was a big unknown. As a result, defining the product in terms of features and price point was a big challenge. Moreover, the goal was to help all Unlimited Plans do well but not at the cost of other

Unlimited Plan products. The DT was used to arrive at product configurations for the key Unlimited Plan products.

Maximizing Throughput of Sorting Terminals

Sorting of packages is one of the most critical activities in package delivery industry. Sorting terminal is a cyber-physical system that aims to maximize its throughput. Figure 2.10 depicts the schematic of a typical sorting terminal comprising the conveyor belt (called *Sorter*) that carries the packages to be sorted, *Infeed* to introduce the packages from carriers to the conveyor belt, *Scanner* for identifying the destination delivery zone from the address written on the package, *Chutes* to collect the packages and *Robotic arm* (in front of every chute) to push the package into the chute. Each chute is assigned a team that collects the packages to be taken to designated loading stations. The sorting terminal can be configured along several parameters, such as the number and types of operational chutes, sorter speed, placement of scanner, assignment of destination delivery zones to a chute, assignment of collecting team to a chute and possible schedule of collecting-team-to-chute assignment. Some of these parameters are relatively more static than others, e.g. placement of scanner. For maximal throughput of the sorting terminal:

1. The package should spend as less time on the sorter as possible.
2. The package should get collected in the right chute.
3. Chutes should get emptied as quickly as possible.
4. No chute should ever be without a collecting team.
5. No package should remain unsorted beyond the prescribed rotations on the sorter, thus leading to manual handling of the package which is time- and cost-expensive.

Current practice is to predict the workload for a shift from past workloads using statistical prediction algorithm. Knowledge of workload is then used to define a suitable

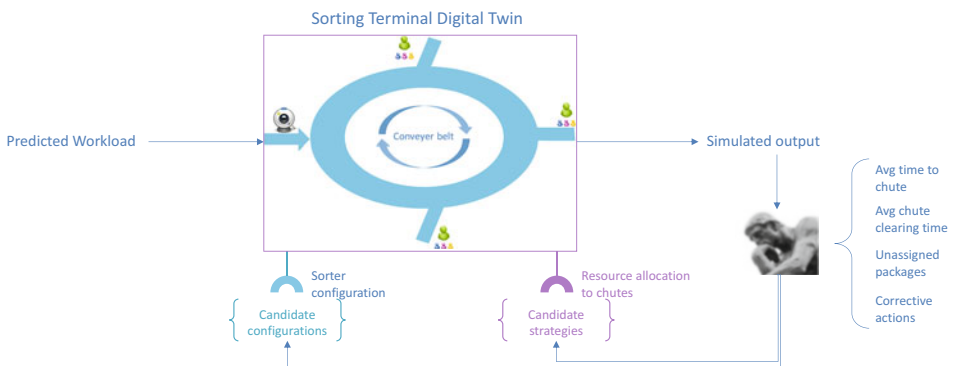


Fig. 2.10 Digital twin of a sorting terminal for maximizing the throughput

sorting terminal configuration that includes a sorting plan (package type to chute assignment), number of active chutes and collecting team size allocated to each chute. This step is part estimate part art. Significant uncertainty in shift workloads, high variance in the temporal order of the workload and varying skills of collecting team are the principal sources for deviation from business-as-usual operation during the shift. These outliers are addressed by the shift manager in an ad hoc manner. As a result, there is little a priori assurance forthcoming as regards throughput of sorting terminal for a shift.

To overcome this problem, we constructed a digital twin of the sorting terminal [33]. We experimented with a wide range of what-if simulations to arrive at the right configuration and to be prepared for possible outlier conditions. The parameters we focused on were average time to chute, average chute clearing time, number of unassigned packages and number of corrective actions introduced. We also ran the digital twin in parallel with the actual sorting terminal in a shift to serve as an early warning system. Moreover, plausible solutions to mitigate the outlier condition were worked out “in silico” – the proverbial “forewarned is forearmed” situation.

Optimizing Shop Stock Replenishment for a Retail Chain

Optimum stock replenishment for shops is a critical need faced by the retail industry. The goal of replenishment is to regulate the availability of the entire product range in each store, subject to the spatio-temporal constraints imposed by (1) available stocks in the warehouses, (2) labour capacity for picking and packaging products in the warehouses, (3) the volume and weight carrying capacity of the trucks, (4) the transportation times between warehouses and stores, (5) the product receiving capacity of each store and (6) available shelf space for each product in each store. Probabilistic behaviours of the individual elements lead to the uncertainties that emerge in the replenishment process, for example, unavailability and varying productivity of the labours, unavailability and unaccounted delays of the trucks and customer footfall leading to eventual purchase at stores.

Current practice is to construct a predictive model based on the past data pertaining to shop stock replenishment. Given the scale of the problem, i.e. number of warehouses, number of stores, number of trucks, number of product types and number of products, purely analytical specification gets too large and hence vulnerable to the errors of omission and commission. As a result, an aggregated lumped-up model is used to train a Reinforcement Learning (RL) agent to learn a policy, i.e. a sequence of actions to be performed by the various stakeholders for shop stock replenishment. The coarseness of model leads to RL agent learning a sub-optimal policy.

We constructed a fine-grained model of the supply chain [34] where each warehouse, shop, truck, container, shelf, product, customer, etc. is modelled as an actor having well-defined (though probabilistic) behaviour. We can model the individual characteristics such as buying propensity of a customer, breakdown vulnerability of a truck, packing errors of a packer, etc. at the finest level of detail. This fine-grained model of the supply chain is used

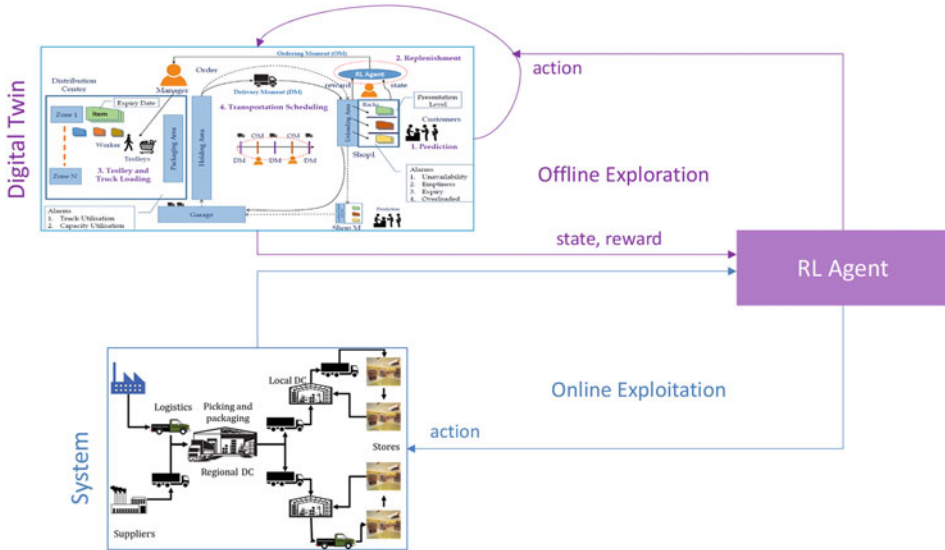


Fig. 2.11 Digital twin for optimal shop stock replenishment

to train RL algorithm to learn a policy for shop stock replenishment. Our approach led to fine-grained learning with faster convergence (refer to Fig. 2.11).

Prediction and Control of Covid-19 Pandemic in a City

During a pandemic like Covid-19, one of the key priorities of the public health administration is to understand the dynamics of the transmission of virus [35] and use that knowledge to design effective control measures to keep its impact on public health within manageable and tolerable limits. While the dynamics of Covid-19 virus transmission and spread in a heterogeneous population is not fully understood, it is known, though, that the spread of infection is related to people's movement, the nature of the area where people congregate and number and frequency of proximal contacts. It is also known the demographic factors and comorbidity play a role in the lethality. Therefore, public health authorities have primarily focused on restricting people movement to varying degrees by imposing lockdowns. In addition to saving lives, lockdowns have been the primary instruments for managing the load on local healthcare systems.

Unfortunately, lockdowns have a serious downside in the form of impact on economy [36]. Until the pandemic is brought under control through large-scale availability of medication or vaccines, one is resigned to pay this price. Having accepted that, one would like to strike a balance between load on healthcare system and revival of the economy. As the dynamics of the spread of Covid-19 depend heavily on individual

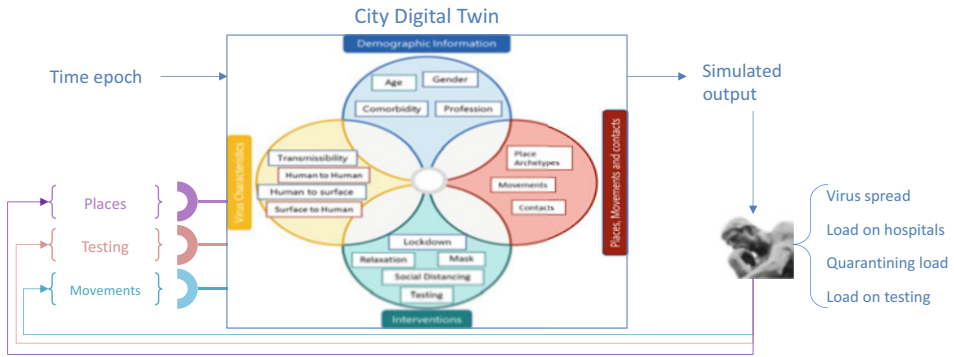


Fig. 2.12 City Digital Twin for the prediction and control of Covid-19 pandemic

localities, devising effective means to help administrators take local decisions is a critical need.

We believe that devising a universal model to predict evolution of Covid-19 at varying levels of granularity such as city localities, cities, counties and countries is a difficult proposition. Instead, a purpose-specific, locality-based, fine-grained model addressing a set of relevant aspects of interest can play a crucial role in decision-making for controlling the pandemic. Therefore, we developed a novel agent-based digital twin of a city [37] to support simulation-based approach to predict and control Covid-19 epidemic as shown in Fig. 2.12. The defining characteristic of the city digital twin is a set of suitable agent types necessary to capture heterogeneity in terms of people, places, transport infrastructure, healthcare infrastructure, etc. As a result, we are able to construct a fine-grained model of the city that is amenable to what-if and if-what scenario playing. We populated the city digital twin using data from the city administration, together with suitable augmentation. The fine-grained nature of digital twin enabled us to address the critical concerns such as the rate and the extent of the spread of the epidemic, demographic and comorbidity characteristics of the infected people, load on the healthcare infrastructure in terms of specific needs such as number of admissions requiring critical care (supplementary oxygen, ventilator support, intensive care, etc.), load on institutional quarantine centres and so on. We set up appropriate what-if scenarios to identify the most effective intervention from the candidate set to control epidemic as well as bring back normalcy. We vetted the simulation results against epidemic-related data released by an Indian city.

Helping Organizations Transition from Work from Home to Work from Office Mode

Organizations are struggling to ensure business continuity without compromising on delivery excellence in the face of Covid-19 pandemic-related uncertainties. The uncertainty

exists along multiple dimensions such as virus mutations, infectivity and severity of new mutants, efficacy of vaccines against new mutants, waning of vaccine-induced immunity over time and lockdown/opening-up policies effected by city authorities. Moreover, this uncertainty plays out in a non-uniform manner across nations, states and cities and even within the cities, thus leading to highly heterogeneous evolution of pandemic. While work from home (WFH) strategy has served well to meet ever-increasing business demands without compromising on individual health safety, there has been an undeniable reduction in social capital. With Covid-19 pandemic showing definite waning trends and employees beginning to miss the office environment, several organizations are considering the possibility of safe transition from WFH to work from office (WFO) or a hybrid mode of operation. An effective strategy needs to score equally well on possibly interfering dimensions such as risk of infection, project delivery and employee (and their dependents) wellness. As large organizations will typically have a large number of offices spread across a geography, the problem of arriving at office-specific strategies becomes non-trivial. Moreover, the strategies need to adapt over time to changes that cannot be deduced upfront. This calls for an approach that's amenable to quick and easy adaptation.

We developed a digital twin-centric approach (refer to Fig. 2.13) that (1) leverages pure data-centric statistical model, coarse-grained system dynamic model and fine-grained agent-based model, (2) helps human experts arrive at pragmatic strategies to effect WFH to WFO transition keeping the key stakeholders satisfied and (3) easily adapts the strategies over time. We have validated the approach with a large organization, and the results are encouraging.

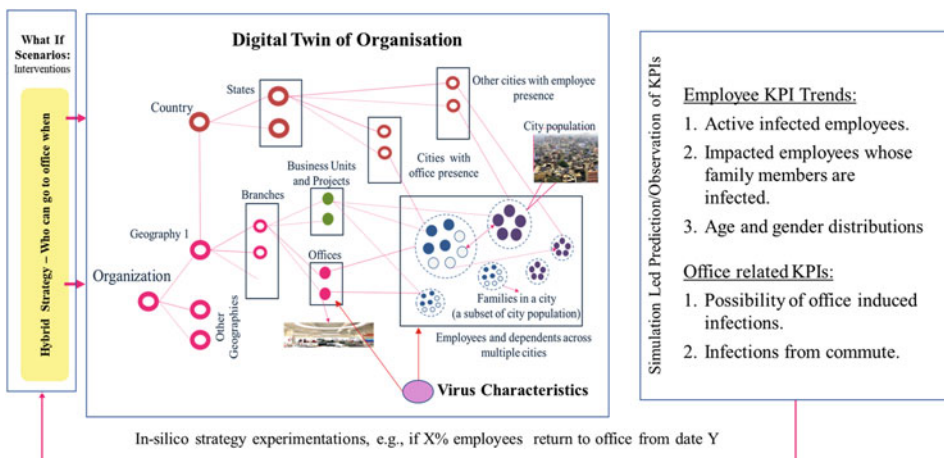


Fig. 2.13 Organization digital twin for WFH to WFO transition

Summary and Future Work

We live in a world comprising a complex system of systems that changes along multiple dimensions in a manner that cannot be deduced upfront. Therefore, decision-making in the face of uncertainty is a critical need. We argued that the state of the art and current practice do not address this need to the desired level of satisfaction and sophistication.

We proposed a simulation-based approach that innovatively integrates and builds further upon proven ideas from modelling and simulation, artificial intelligence and control theory. At the heart of the approach is the concept of digital twin – a purposive virtual high-fidelity simulatable representation of the reality that’s amenable to what-if and if-what scenario playing. We described the core technology infrastructure necessary to implement the proposed approach in a “human-in-the-loop” manner.

We discussed the utility and efficacy of the approach on real-world industry-scale use cases spanning across business (telecom case study and retail case study), cyber-physical (sorting terminal case study) and societal (Covid-19 case studies) domains. Almost everywhere, the proposed approach has fared better than current practice.

We believe the city digital twin can be repurposed to address emerging socio-techno-economic challenges such as sustainable enterprise, smart city and wellness and healthcare. While we seem to be on the right track, there is a lot that needs to be done as regards multi-paradigm Digital Twin(s), multi-objective Reinforcement Learning, agents that are composable “first class”, adaptive Digital Twin(s), method support for construction and use of Digital Twin(s) and leveraging reasoning and knowledge. We also see the possibility of taking the idea of digital twin to the software systems to support dynamic adaptation in the face of uncertainty. The combined ability of arriving at the right decisions and effecting them in an efficacious way will help realize an adaptive enterprise where adaptations are justification-backed.

References

1. Ignizio, J. P., & Cavalier, T. M. (1994). *Linear programming*. Prentice-Hall.
2. Currall, S. C., & Towler, A. J. (2003). *Research methods in management and organizational research: Toward integration of qualitative and quantitative techniques*. Sage Publications.
3. Kothari, C. R. (2004). *Research methodology: Methods and techniques*. New Age International.
4. Michalski, R. S. (1993). Inferential theory of learning as a conceptual basis for multi strategy learning.
5. Thomas, M., & McGarry, F. (1994). Top-down vs. bottom-up process improvement. *IEEE Software*, 11(4), 12–13.
6. Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.
7. Schrijver, A. (1998). *Theory of linear and integer programming*. Wiley.
8. Mangel, M., & Samaniego, F. J. (1984). Abraham Wald’s work on aircraft survivability. *Journal of the American Statistical Association*, 79(386), 259–267.

9. Sandkuhl, K., et al. (2016). Enterprise modelling for the masses—from elitist discipline to common practice. In *IFIP Working Conference on The Practice of Enterprise Modeling* (pp. 225–240). Springer.
10. Zachman, J. A. (2003). *The Zachman framework for enterprise architecture. Primer for Enterprise Engineering and Manufacturing*. Zachman International.
11. Iacob, M. E., Jonkers, H., Lankhorst, M., Proper, E., & Quartel, D. A. C. (2012). *ArchiMate 2.0 specification*.
12. White, S. A. (2004). Introduction to BPMN. *Ibm Cooperation*, 2(0), 0.
13. Horkoff, J., & Yu, E. (2010). Visualizations to support interactive goal model analysis. In *2010 Fifth International Workshop on Requirements Engineering Visualization* (pp. 1–10). IEEE.
14. Meadows, D. H. (2008). *Thinking in systems: A primer*. Chelsea Green.
15. Camus, B., Bourjot, C., & Chevrier, V. (2015). *Combining devs with multi-agent concepts to design and simulate multi-models of complex systems*.
16. Frank, U. (2011). *The MEMO meta modelling language (MML) and language architecture* (No. 43). ICB-research report.
17. Armstrong, J. (2013). *Programming Erlang: Software for a concurrent world*. Pragmatic Bookshelf.
18. Hewitt, C. (2010). *Actor model of computation: Scalable robust information systems*. arXiv preprint arXiv:1008.1459.
19. Vernadat, F. (2020). Enterprise modelling: Research review and outlook. *Computers in Industry*, 122, 103265.
20. Mcmillan, C. J. (1980). Qualitative models of organisational decision-making. *Journal of General Management*, 5(4), 22–39.
21. Daft, R. L. (2015). *Organization theory and design*. Cengage Learning.
22. Kohavi, R., & Longbotham, R. (2017). Online controlled experiments and A/B testing. *Encyclopedia of Machine Learning and Data Mining*, 7(8), 922–929.
23. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
24. Butler, H. (1992). *Model reference adaptive control: From theory to practice*. Prentice-Hall.
25. Stiglitz, J. E. (1981). Pareto optimality and competition. *The Journal of Finance*, 36(2), 235–251.
26. Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1), 48–49.
27. Clark, T., Kulkarni, V., Barat, S., & Barn, B. (2017, June). ESL: An actor-based platform for developing emergent behaviour organisation simulations. In *International conference on practical applications of agents and multi-agent systems* (pp. 311–315). Springer.
28. Agha, G. A., Mason, I. A., Smith, S. F., & Talcott, C. L. (1997). A foundation for actor computation. *Journal of Functional Programming*, 7(1), 1–72.
29. Sunkle, S., Saxena, K., Patil, A., & Kulkarni, V. (2022). AI-driven streamlined modeling: Experiences and lessons learned from multiple domains. *Software and Systems Modeling*, 21(3), 1–23.
30. Sargent, R. G. (2004, December). Validation and verification of simulation models. In *Proceedings of the 2004 Winter Simulation Conference* (Vol. 1). IEEE.
31. Clark, T., Barn, B., Kulkarni, V., & Barat, S. (2017). *Querying histories of organisation simulations*. ISD 2017.
32. Barat, S., Kulkarni, V., Kumar, P., Bhattacharya, K., Natarajan, S., & Viswanathan, S. (2020, July). Towards effective design and adaptation of CSP using modelling and simulation based digital twin approach. In *Proceedings of the 2020 summer simulation conference* (pp. 1–12).
33. Ghosh, S., Pal, A., Kumar, P., Ojha, A., Paranjape, A., Barat, S., & Khadilkar, H. (2021). A simulation driven optimization algorithm for scheduling sorting center operations. In *In 2021 Winter Simulation Conference (WSC)* (pp. 1–12). IEEE.

34. Barat, S., Khadilkar, H., Meisheri, H., Kulkarni, V., Baniwal, V., Kumar, P., & Gajrani, M. (2019, May). Actor based simulation for closed loop control of supply chain using reinforcement learning. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 1802–1804).
35. World Health Organization, et al. (2020). *Modes of transmission of virus causing Covid-19: Implications for IPC precaution recommendations*. Scientific brief, 27 March 2020 (Technical Report). World Health Organization.
36. Fernandes, N. (2020). *Economic effects of coronavirus outbreak (Covid-19) on the world economy*. Available at SSRN 3557504.
37. Barat, S., Parchure, R., Darak, S., Kulkarni, V., Paranjape, A., Gajrani, M., & Yadav, A. (2021). An agent-based digital twin for exploring localized non-pharmaceutical interventions to control COVID-19 pandemic. *Transactions of the Indian National Academy of Engineering*, 6(2), 323–353.



Regulatory Compliance at Optimal Cost with Minimum Exposure to Risk

3

Vinay Kulkarni 

Introduction

Businesses are getting increasingly regulated. Regulatory compliance is a board-level concern and one of the top-3 CEO-level concerns across business verticals.¹ Failure to comply not only leads to heavy fines and reputational risk but may also make top management personally liable. Enterprises need to be cognizant of compliance vs risk trade-off as exorbitantly high cost of compliance can make the enterprise unviable.² Thus, regulatory compliance at optimal cost with minimal exposure to risk is a critical need faced by modern enterprises across business domains. The same problem manifests as regards internal policies with auditors playing the role of regulating body.

As modern enterprises operate in a dynamic environment, e.g. a bank typically receives about 200 regulatory alerts every day,³ the compliance management process needs to be suitably responsive to ensure the cost of compliance is proportional to the change being introduced. The change can also originate within the enterprise due to business events such as merger and acquisition, organizational restructuring, system integration, business process reengineering, technology obsolescence/advance, etc. Modern enterprises typically operate in several geographies which may have different regulations and/or geography-specific variants of a regulation. The compliance management process needs to be

¹ <http://www.smbceo.com/2020/02/08/what-is-the-role-of-the-ceo-in-regulatory-compliance/>

² <https://www.bankingexchange.com/bsa-aml/item/8202-cost-of-compliance-expected-to-hit-181bn>

³ <https://thefinanser.com/2017/01/bank-regulations-change-every-12-minutes.html/>

V. Kulkarni (✉)

Tata Consultancy Services Research, Pune, Maharashtra, India

e-mail: vinay.vkulkarni@tcs.com

cognizant of these variations to address the geography-specific needs without duplication of effort.

Effective compliance management requires legal, domain and IT systems expertise – all of which are in scarce supply. Ideally, the process should be able to call upon each of this expertise only when needed and in a manner not to impede the other two, thus leading to smooth coordinated compliance management. With enterprises rapidly evolving into a dynamic system of systems or ecosystem, the compliance problem gets further complicated.

Regulatory Compliance

Enterprise is driven by its stated goals. It defines strategies aimed at achieving these goals in the best possible manner. Operating processes and IT systems come up to implement these strategies. To ensure a level playing field, it's critical to ascertain that objectives of all key stakeholders are satisfied and no specific enterprise enjoys an unfair advantage over its competitors. To this effect, the regulating authority (or regulator) issues commensurate regulations that all enterprises must comply with. Regulator seeks a report from its regulatees at a pre-defined frequency asking for specific information that's capable of identifying whether any regulation is violated as the enterprise goes about executing its operating processes and IT systems. Figure 3.1 depicts the conceptual model of regulatory compliance.

The uncluttered conceptual view of regulatory compliance depicted in Fig. 3.1 gets complex and tangled in implementation. For example, regulations are rarely orthogonal; instead, they overlap, and this overlap is one of the principal contributors to complexity as explained below.

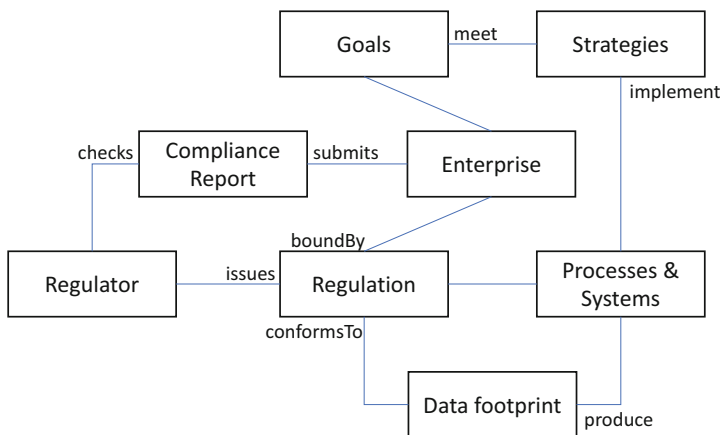


Fig. 3.1 Regulatory compliance – conceptual view

to. Therefore, checking for compliance is analogous to querying the enterprise information footprint where the query corresponds to an obligation.

As enterprise operates in a dynamic environment, the five documents need to be kept in sync for every alert received from the regulators. Also, the process and functional specifications documents may undergo a change in the light of business events such as technology change/obsolescence, process reengineering, mergers and acquisitions, etc. Therefore, regulatory compliance problem can be seen to comprise five sub-problems: *hygiene* (policy, control and process documents are consistent with each other and with regulation document), *checking* (the relevant subset of enterprise information footprint conforms to the obligations specified by a regulation), *change management* (identify the change in regulation and perform *hygiene* and *checking* tasks in a change-driven manner), *risk* (compute the risk of non-compliance and devise means to reduce it to the extent possible) and *make compliant* (identify the changes required to eliminate non-compliance and effect the changes appropriately).

Current Practice

The key challenges that need to be overcome for effective regulatory compliance management are identifying obligations from the regulation document; fetching relevant data to be examined from enterprise information footprint; checking whether the obligations hold; tracing non-compliant data to the relevant obligation, i.e. the text statement in regulation document; addressing non-compliance by introducing appropriate changes in process and software systems; and risk assessment and management. Effective compliance management requires legal, domain and IT experts to work together in a coordinated manner. For example, legal and domain experts need to work together to produce contextualized regulation document, and correct interpretation of the regulation into policies and controls in the enterprise context and their implementation in enterprise processes and systems requires teams of legal, domain and IT systems experts working together. With expertise already in short supply, this coordination calls for sophisticated tool support. However, current practice of compliance management is essentially document-centric and therefore largely manual. Generic and document-centric compliance management tools such as GRC⁴ frameworks put heavy analysis and synthesis burden on human experts exacerbating the supply problem further. Lack of sophisticated automation means use of GRC framework is vulnerable to cognitive limit of its users and fatigue-induced errors of commission and omission.

Several regulation-specific offerings exist that produce compliance reports out of the box. However, the onus of providing the right data of right veracity is solely on the user. As a result, these compliance solutions are vulnerable to *garbage in garbage out*. Furthermore,

⁴<https://www.ibm.com/learn/cloud/grc>

these point solutions designed strictly for standalone use present a formidable system integration challenge for using them in a cohesive integrated manner. This is a serious shortcoming especially given the large portfolio of regulations an enterprise needs to comply with. With regulators issuing frequent amendments and new regulations, this lack of responsiveness of the present compliance management technology is turning out to be untenable.

There is hardly any sophisticated support available for managing compliance-related risks. This critical need is today addressed relying solely on human experts.

Regulatory compliance problem has attracted research community as well. In contrast to manual document-oriented industry practices, research approaches have focused on rigor and automation using formal techniques to check compliance [1, 2]. These have seen poor adoption in industry due to the difficulty of manually coming up with formal specifications of regulation rules from the regulation document in NL text [3, 4]. Also, research approaches do not address the entire compliance management process.

Thus, it can be said that current practice falls short of effectively addressing the problem of complying at optimal cost with minimal exposure to risk. A new approach seems called for.

Tenets of a Desirable Line of Attack

Here is the intuition leading to key tenets.

Can a regulation be viewed as a logic program? The obligations can now be specified as Horn clauses [5] in terms of variables that correspond to the appropriate data elements from enterprise data footprint. One can choose from a variety of logics such as deontic [6] and defeasible [7] for specifying regulations as a logic program which can be analysed using sophisticated machinery available. Thus, it seems possible to automate the compliance checking aspect. However, the abstraction gap between regulation described in natural language (NL) text and an equivalent logic program needs to be bridged. This gap is typically too large to be bridged in a single leap. Instead, successive refinements need to be adhered to. Moreover, this refinement process needs to be amenable to a change-driven implementation as regulations as well as enterprise IT systems tend to change over time and the rate of change is continually increasing.

Can the refinement process be automation aided so as to reduce the heavy analysis and synthesis burden on human experts? A proven technique to achieve this objective is to leverage purposive structure underlying the problem, i.e. a purposive model. Figure 3.3 shows a subset of a purposive model for regulatory compliance management. Regulation is viewed as a set of obligations that the enterprise must provide evidence of having complied with. Obligation is best viewed as a rule comprising a condition to be met and an action to be performed when the condition is met. Condition can be a complex logical expression over variables which must come from enterprise data footprint. Antecedent of a rule refers to several kinds of actions – changing the value of a variable being one of the simplest. The

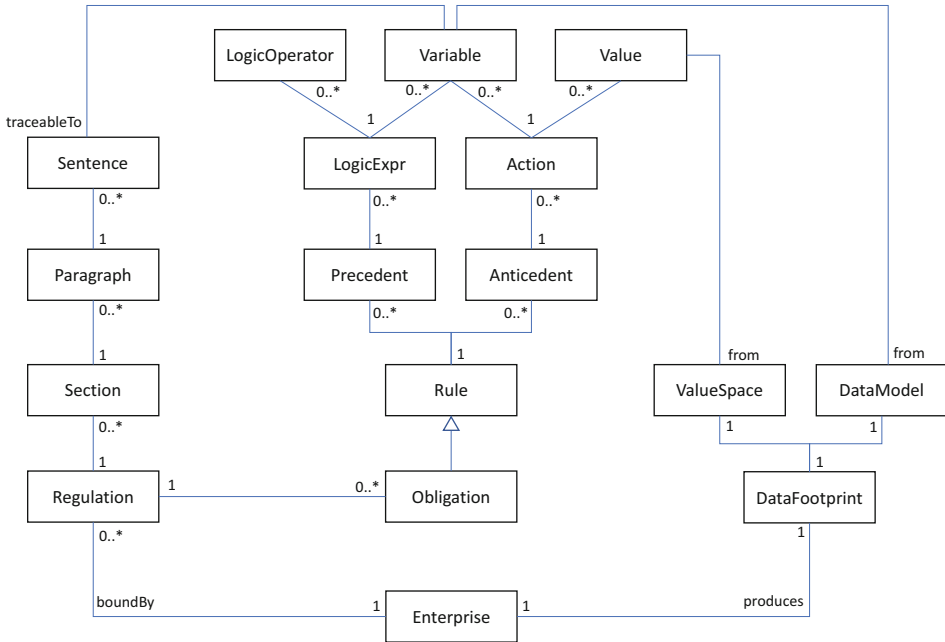


Fig. 3.3 Regulatory compliance – a subset meta-model

structure of Fig. 3.3 can serve as a lens to look at the regulation document in NL text. Existing NLP techniques can be used to populate this model from NL text document in a human-in-the-loop automated manner. Moreover, the resultant shift from document-centric to model-based operation not only brings in significant automation but also reduces intellectual overload on human experts as the formal model is amenable to rigorous analysis (and transformation) in an automated manner.

Can the refinement process be change-driven so as to ensure the cost of compliance is proportional to the change in regulation and/or enterprise? Shift to a model-based compliance management process can bring in this benefit too. Information about regulations, policies, controls, processes, obligations, variables, etc. is no longer spread across multiple NL text documents but at one place as a formal model. As a result, various kinds of relationships between the content of these documents can now be captured in the form of associations between these models. With such interconnected models, it is now possible to precisely compute the impact of a change in one model onto other models and that too automatically.

Can the refinement process enable coordinated involvement of legal, business and IT experts so as to ensure optimal use of scarce resources? Refinement of the generic text in regulation document issued by regulatory authorities involves legal and business experts with the latter bringing specifics of regulatee enterprise to the table. While it is possible to provide automation help, it is probably better to perform this task at NL text document

level. This is largely because legal experts are typically not very comfortable with sophisticated computing technologies. Subsequent steps of the refinement process (refer to Fig. 3.2) are characterized by the principle of separation of concerns and hence enable business and IT experts to work independently and yet in a coordinated manner. Shift from document-centric to model-based compliance management is the key. Moreover, if IT systems and business processes are architected for easy configuration using say feature modelling and variability modelling techniques, then model-based compliance management process can also deliver value towards making these business processes and IT systems compliant in an automated change-driven manner.

Can the compliance management process benefit from what-if simulations so as to be aware of likely risks in the future and be adequately prepared in advance for mitigating them? Moreover, can there be a simulation machinery that can serve as an early warning system by predicting outlier situations? Shift to model-based regulatory compliance seems to be the first step to achieve these objectives. However, the models need to be richer (i.e. capable of more sophisticated analysis) and more detailed (i.e. capturing information that helps in risk management).

AI-Aided Model-Based Automated Regulatory Compliance

We take a holistic approach to implement the tenets discussed above by changing current document-centric focus of compliance practice to a model-driven approach. Models capture information explicitly in a structured form, thus making relationships explicit. Moreover, models (1) are easy to navigate, (2) enable precise computation of change impact, (3) are machine-processable and (4) are amenable to automated analysis and transformation. We provide assistance for authoring these models from NL text documents using Natural Language Processing (NLP)- and Machine Learning (ML)-based extraction techniques and a near-natural language interface. We use the concept model generator (CMG) which enables human-in-the-loop modelling of the domain under consideration [8]. The construction of domain model uses *text ranking* (alternatively, clustering) and *open information extraction* to obtain key concepts and relations from which a base model is constructed automatically [9]. The text ranking implementation used by CMG is based on Google PageRank algorithm adapted to text. The domain expert refines the base model using a graphical user interface that provides facilities to explore the key concepts and their mentions and relations within the underlying text (of regulations, policies and controls) [10]. Legal and domain experts together author models of the regulation, policies, controls and business processes from their respective source documents to create a purposive digital twin of the enterprise as depicted in Fig. 3.4.

We provide technology support for domain experts to validate these models for consistency and correctness, thus enabling the early detection of errors. Logic programming and

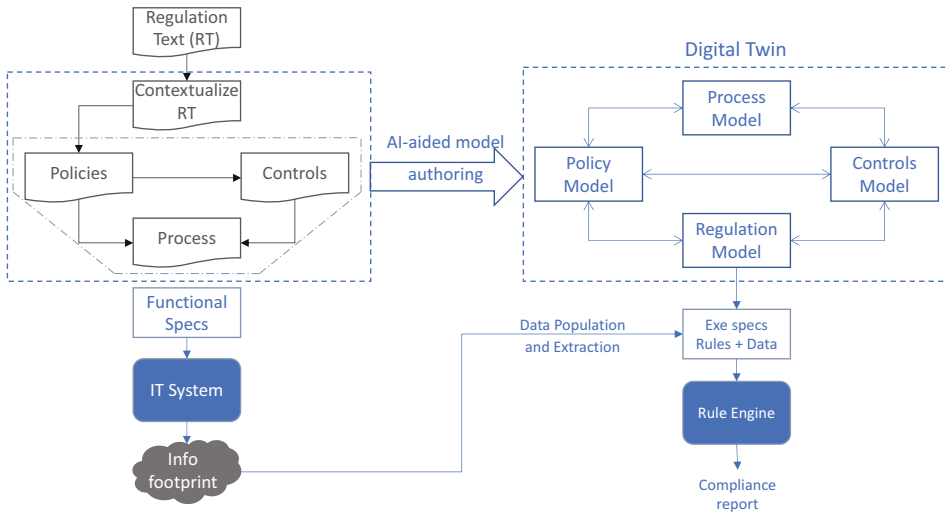


Fig. 3.4 AI-aided model-based approach to automated regulatory compliance

automated reasoning techniques are used to generate key representative scenarios to be presented to experts for further validation. The validated models are automatically transformed into an executable form with clear identification of the necessary and sufficient data for compliance checking. We provide technology to pull this data from wherever it exists in the enterprise. Proven technology is used to automatically check whether the data conforms to regulation rules. In case of non-compliance, we clearly identify the non-compliant data element and provide traceability links to the regulation text being violated.

The proposed approach leads to several advantages. Shifting the focus from text documents to models helps bring agility to compliance management process. Impact of a change in regulation can now be computed automatically and with precision. Further downstream process of compliance checking is cognizant of the change and hence more efficient. Analysis and synthesis burden on human experts is considerably reduced as models can be automatically checked for internal consistency, correctness and completeness. Furthermore, automatic transformation of these models to an executable form and subsequent automated checking results in highly responsive compliance process. Automated nature of our approach can eliminate the present need to resort to sample-based compliance, thus improving the reliability of compliance process. Furthermore, the compliance checking process can be always running in the background to enable the identification of deviation from compliance state as soon as it takes place, thus minimizing its ripple effects within the enterprise.

Technology Infrastructure to Support the Line of Attack

AI-Based Model Authoring

We use AI techniques to author model of a regulation from its NL text as shown in Fig. 3.5. As the first step, we help Subject Matter Eexpert (SME) author *Concept Model*, i.e. concepts being referred to in the regulation text and relationships between them. The concept model serves as a lens to focus on the relevant NL text containing obligations and data. As the same concept may get mentioned in multiple ways in the regulation document, these sets of *mentions* are constructed to eliminate redundancy. We provide a *Controlled Natural Language* (CNL) for SMEs to specify the obligations found. The CNL serves as an intermediate representation that's closer to the business domain (and hence intuitive to SME) but with well-defined syntax and semantics (and hence amenable to machine processing). We provide syntax-driven editing support for SME to organize the obligations and data in CNL which can be automatically checked for syntactic and semantic well-formedness. As CNL has well-defined syntax and semantics, it is possible to construct a translator from CNL to other languages using the principle of syntactic transformation under semantic in variance. We use the Semantics of Business Vocabulary and Business Rules (SBVR⁵) as the modelling language to specify regulations. We have chosen CNL such that it's quite close to the linearized representation of SBVR. As a result, text in CNL can be automatically transformed to a model in SBVR.

We author models for policies, controls and processes along similar lines. However, their languages are different yet relatable to each other and to the regulation model in

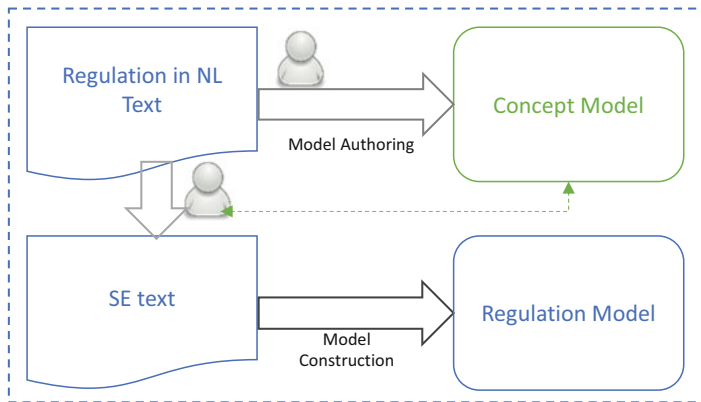


Fig. 3.5 AI-aided model authoring

⁵<https://www.omg.org/spec/SBVR>

SBVR. We thus shift the focus of regulatory compliance from text documents to formal models leading to several benefits [11].

Automation: Being in model form as opposed to NL text, it is now possible to ensure that policies, controls and processes are consistent with each other and with the regulation. Automation aids for validating the authored model help capture inconsistency in regulation. Also, it is possible to check if any aspect of regulation is not being captured in policies and controls. Moreover, this assurance comes with automation, thus eliminating fatigue-induced errors of omission and commission that are so prevalent in manual process.

Separation of concerns: SME having cursory knowledge of legalese text is independently able to author models from NL text. CNL provides an intuitive interface, thus hiding the technical complexities of NLP as well as modelling languages.

Completeness: Ensuring that no obligation is missed in authoring regulation model from its NL text is a hard problem. In concept model, SMEs get a lens that helps them focus on the relevant sections of regulation document only, thus reducing fatigue as regulation documents are typically rather large. In addition, our tool highlights the text from which obligations are extracted. These aids go some distance in helping ensure that no obligation is missed.

Therefore, it can be said that the proposed approach fares better on the counts of *hygiene and change management*.

Validating the Authored Model

Model-driven engineering (MDE) approach necessitates verification and validation (V&V) of the models used. Model verification seeks to check that the model does what it is expected to do, usually through an implementing program.⁶ We propose inconsistency checking, scenario generation and testing using a standard set of test cases and test data as verification techniques in our approach. The objective of model validation is to ascertain that the model is a faithful representation of the problem space it represents.³ We implement validation by having SMEs certify scenarios generated from the model for validity and coverage. Figure 3.6 depicts the overall approach.

We use SBVR as modelling notation and Answer Set Programming (ASP) as the logic paradigm for the verification of regulation model authored by SMEs [12].

SBVR is a fact-oriented modelling notation [13] that encodes rules as logical formulations over fact types that are relations between concepts.⁷ Rules are stated in SBVR SE to populate the model.

⁶<http://www.inf.ed.ac.uk/teaching/courses/pm/Note16.pdf>

⁷<https://www.omg.org/spec/SBVR/1.3>

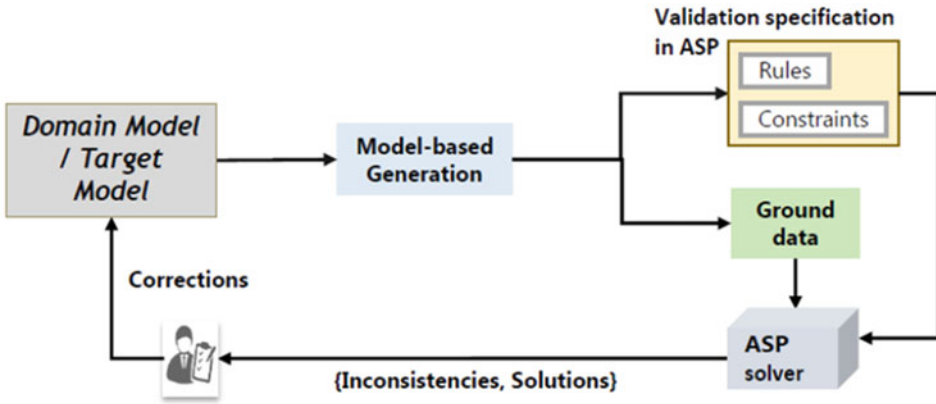


Fig. 3.6 Model validation and verification

We choose ASP for automated verification since (1) it is a logic programming paradigm that maps directly onto SBVR’s fact-oriented model and underlying first-order logic formalism and (2) it is a powerful, highly expressive notation supporting aggregates, functions and optimizations. We translate rules in SBVR to ASP rules and constraints.

We use the DLV system [14] as the solver for our generated ASP programs. The solver performs consistency checking between rules, constraints and ground facts, indicating conflicting rules or constraints and generating answer sets for paths where no conflicts exist. Answer sets, by definition, are minimal, i.e. no answer set can be contained in another [15], and represent the minimal set of scenarios for the input model. These are presented to SMEs to check whether the generated scenarios are valid and whether all important scenarios have been covered.

Automating Compliance Checking

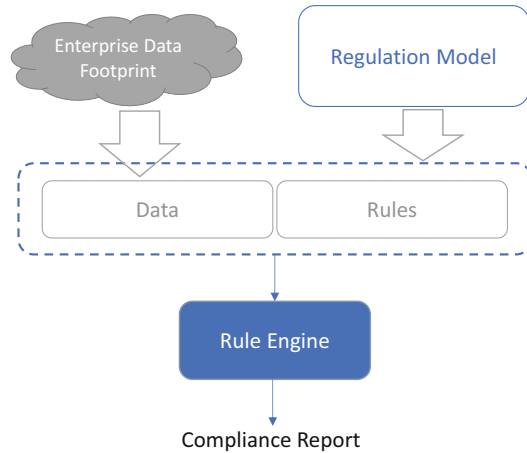
Figure 3.7 depicts our automation infrastructure for compliance checking. We transform the validated regulation model in SBVR to an executable form by using model transformation techniques.^{8,9} We support two executable specifications, namely, DR-Prolog [16] and Drools.¹⁰ For ease of adoption, we prefer generating rules. A rule is an executable specification of an obligation – a Horn clause over a set of variables.

⁸<https://www.omg.org/spec/MOFM2T/>

⁹<https://www.omg.org/spec/QVT/>

¹⁰<https://www.drools.org/>

Fig. 3.7 Automated compliance checking



For populating values of rule variables from structured data, we rely on the schema mapping, to identify the right record from appropriate relational tables [17]. For populating these values from the unstructured data, we use text mining techniques [8–10, 18].

With the rule variables populated, we use a rule engine to check whether the Horn clause is satisfied. If not, it's easy to figure out which variable is leading to non-satisfaction, i.e. non-compliance with respect to the obligation. We maintain traceability from variable all the way back to the NL text in the regulation text.

Benefits of the Proposed Approach

Purposive meta-model serves as a lens for looking at regulation text, thus helping filter out text unrelated to obligations. The modified PageRank algorithm extracts first-cut domain model that can be extended by SME if required. With domain model as a lens over NL text, NLP techniques help identify obligations in NL text. These obligations are presented in Controlled Natural Language (CNL) to be refined further by SMEs on a need basis. A formal model (SBVR) is automatically constructed from CNL using standard model authoring techniques. Answer Set Programming helps SMEs to validate the constructed model of regulation using a minimal set of scenarios. Validated regulation model is automatically transformed to executable form, say rules, using standard model-to-text transformation techniques. Extracted data model helps automatically pick the relevant data from enterprise information footprint. Rule engine checks if the data conforms to the rule, thus completing the compliance checking process.

Shift to model-based regulatory compliance also helps in *hygiene* (as the models are amenable to automated checking for consistency, completeness and adherence to domain constraints) and *change management* (as regulations, policies, controls and system specs exist in model form, it is possible to precisely compute the impact of a change in one model

onto others, thus facilitating change-driven compliance process where the cost of compliance management is proportional to the change). Where business processes and IT systems are architected for easy configuration, model-based compliance management also helps in making them compliant. Moreover, making these models simulatable addresses risk management task through scenario playing. Also, such a compliance-specific digital twin can serve as an Early Warning System to help enterprise prepare better for outlier situations.

Illustrative Use Cases of Automated Regulatory Compliance

We describe a representative sample of use cases that illustrate utility and efficacy of the proposed approach in industry context.

Assurance of Hygiene

A compliance hygiene solution was used to demonstrate P&C regulation rationalization for a US insurance company.

Business Problem

The insurer needed to create a centralized and rationalized obligation library of the code books across all the US jurisdictions to stay compliant to obligations. This needed processing of all the code books across state jurisdiction in order to classify citation/law/obligations as per categories of insurer's business functions. Current practice involves legal and domain experts scanning documents manually, which is time- and effort-intensive, without assurance of full coverage, completeness and consistency.

Scope

There are more than 250 categories associated with the insurer's business functions. Definitions of these categories constitute input text. Obligations as per laws, statutes and regulations of 55 diverse states were to be mapped to these business categories and rationalized.

Approach

As shown in Fig. 3.8, the regulation/law text was mapped to the pre-defined category and sub-category definitions using the navigable ontologies extracted from the inputs using the Automated Regulatory Compliance toolset supporting the proposed approach (ARC). ARC enabled category definitions to be used as query text to search across all the input regulations, laws and statutes for each jurisdiction. This enabled experts to map obligations that were relevant to the category definitions and capture citations that confirm this mapping.

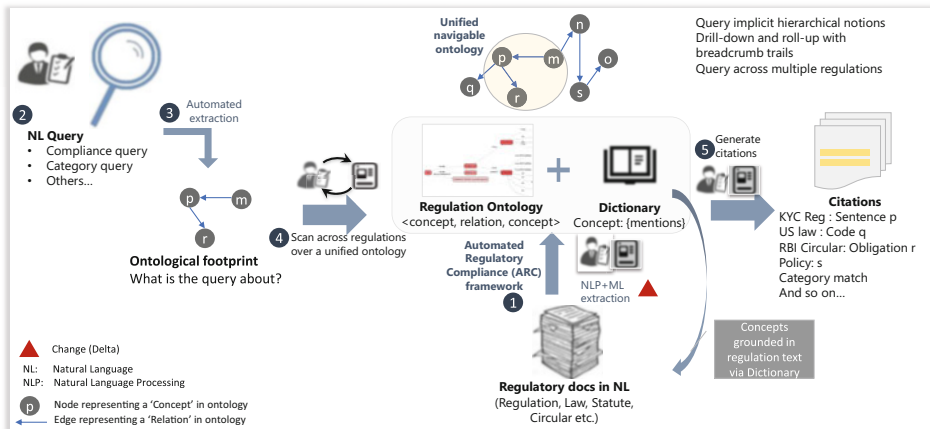


Fig. 3.8 P&C regulation rationalization

Benefits

The following benefits were observed:

- Comprehensive coverage – using ARC, the categorization and mapping of citations for 55 states in the USA was completed in less than 6 months' time frame.
- Over 90% accuracy in suggested mappings of obligations and its categorization.
- Effort and time savings resulting in estimated 90% cost savings (USD450K vs USD4.5Mi projected).
- Reduced burden on SMEs and reduction in human errors in the process.
- One-time effort for categorization enables the management of regulatory, legal changes as a delta.

Compliance Hygiene and Change Impact Management

Business Problem

A large EU bank initiated an effort to redesign the target operating model for their compliance function and was looking to leverage “RegTech” designs to drive increased automation in compliance management value chain. The focus area for PoC was regulatory intelligence.

Objectives

The principal objectives were (1) to establish lineage from any given regulation to bank's policy and control documents and (2) to identify the impact of specific regulatory changes on policy and control documents.

Scope

ARC solution was leveraged to drive human-guided automation in regulatory intelligence and policy/control lineage aligned to target operating model. Regulation covering Basel Committee on Banking Supervision guidelines related to Anti Money Laundering (AML)/Counter Financing of Terrorism (CFT) was used for the proof-of-concept (PoC) exercise. Anti-Money Laundering Client Onboarding Policy and Internal Control libraries associated to Know Your Customer (KYC)/AML functions were received as in-scope documents from the bank.

Approach

Broad approach used was to apply ARC to identify obligations from the regulatory guidelines and establish lineage from obligations to policy and operating controls. Baseline ontology for the documents in scope was created automatically. Then ontology was extended for the regulation, policy and control documents sequentially as in Fig. 3.9, to prepare navigable ontologies. Obligations were automatically extracted from the regulation document and subsequently reviewed by SMEs. ARC aids allowed the identification of a ranked list of matched sentences in the target document (policy or control) which are similar to a selected sentence in the source (regulation) document. Such aids also helped to establish a reason for their similarity as displayed in Figs. 3.10 and 3.11.

Benefits

The following benefits were observed:

- Human-guided automation in measuring the compliance implementations through policy and controls leading to 25% savings in efforts compared to similar endeavours in the past

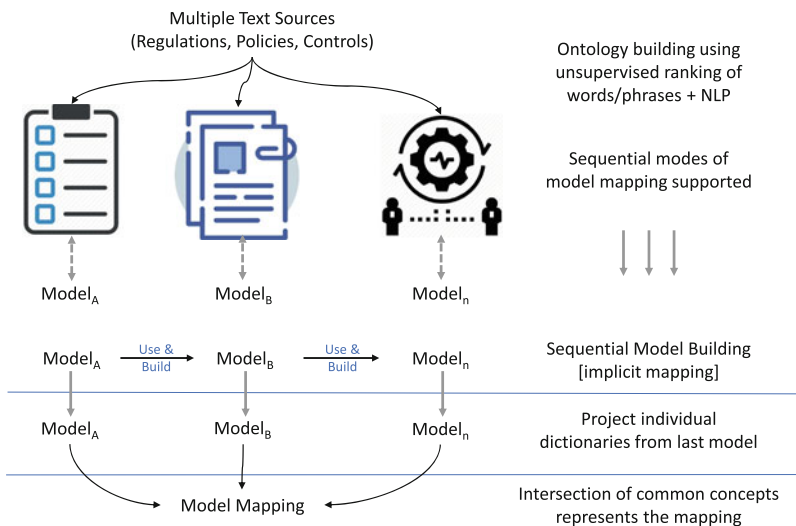


Fig. 3.9 Sequential model building

Account residesAt Bank.
 Bank requiredToApply Risk management.
 Risk management involves Customer Identification Procedure.
 Customer Identification Procedure carriesOut Identification and Verification.
 Identification and Verification basedOn Identity Information.
 Identity Information verifiedUsing Document.
 Bank mitigates Risk.
 Risk has Risk Level.
 Due diligence event merits Risk management.
 Risk management requiredToManage Risk.

Fig. 3.10 Model/ontology walking for the sentences selected in two documents

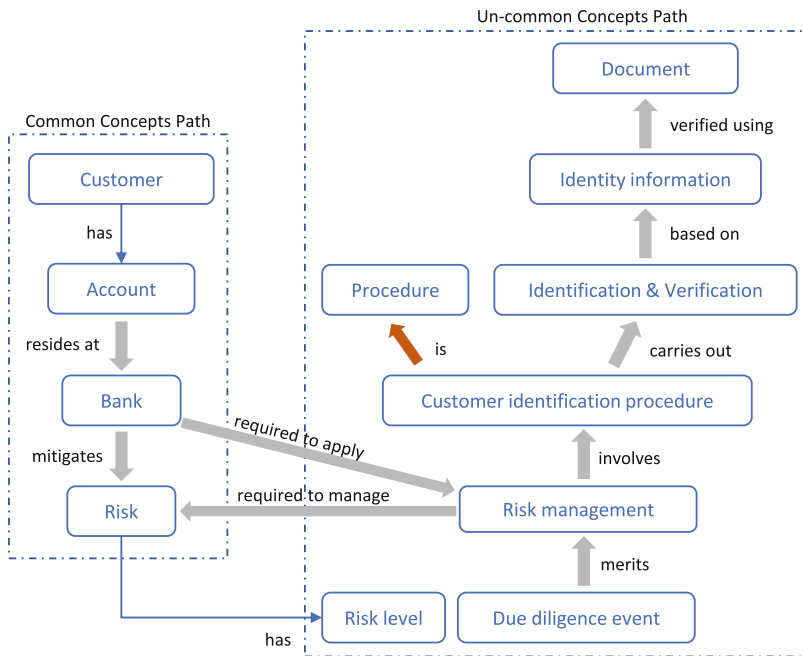


Fig. 3.11 Subset model for the sentences selected in two documents

- Provides evidence of compliance and evidence of gap for timely management interventions
- Seamless integration possible with the bank’s landscape for regulatory intelligence and change management
- AI-driven, model-centric automation to reduce burden on experts
- Reusable ontology models that can be extended to new documents and revised for changes in documents

Compliance Checking

Business Problem

A large bank was spending considerable time, effort and money in staying compliant. Its large size and multi-geography operation made use of off-the-shelf RegTech tools difficult. A purposive regulation-agnostic solution that can be tailored to the bank's specific need seemed the right solution.

Current Practice

Legal and domain experts tag specific interpretation of a regulation connecting rules to enterprise artefacts and data that are identified by IT and domain experts in collaboration. This process is manual with untagged rules at 50% and incorrectly tagged rules at 50%, resulting in high degree of gaps in tagging. Process of checking for compliance is manual. As a result, the process fares poorly in terms of correctness, completeness, agility and scalability.

Objectives

Demonstrate utility and efficacy of proposed approach on MIFID¹¹ regulation.

Scope

Money Market Statistical Reporting (MMSR) regulation

Approach

MMSR regulation and the relations between these concepts were extracted using ARC. SMEs review and extend the ontology to capture additional concepts and relations. The concepts and relations in the ontology serve as a vocabulary for authoring Structured English (SE) rules which is a controlled natural language and an OMG standard for capturing business rules. These rules were automatically converted into an SBVR (Semantics of Business Vocabulary and Business Rules) model, another OMG standard. This model is automatically transformed to generate Drools rules and a conceptual model of data to be checked for compliance. Using TCS EDI (Enterprise Data Integration) framework, the enterprise data is mapped to the conceptual model generated by ARC. Finally, the data is checked against the rules, and a final validation/compliance report is generated which contains the full traceability from NL text, to authored rules, and the data.

Benefits

The following benefits were observed:

¹¹<https://www.esma.europa.eu/policy-rules/mifid-ii-and-mifir>

- The framework provides enhanced controls for compliance breach monitoring and breach management to reduce the incidents while reducing analysis and synthesis burden on experts:
 - Autogeneration of 49 database tables + 97 SQL queries
 - Autogeneration of 36,000+ LoC fact code, 2700+ LoC POJO and ~700 LoC rules code
- Identification of right data to be checked for compliance.
- Human-guided process for automated compliance checking (100 MMSR messages checked for PoC).
- Full traceability in compliance reports.
- Quantitative benefits can be stated precisely only when the same compliance checking situation is addressed “with automation” and “without automation”. As such, a comparison is rather difficult, while working in real-life situation, we think the effort savings for “similar” situations are likely to be around 25% with the use of automation aids.

Change Management

Business Problem

A large MEA bank was interested in exploring automation mechanisms that could help identify the impact of a regulation or a regulatory circular on the banks’ internal policies.

Scope

Identify changes in policy document such as Access Management Policy due to RBI Circular RBI/2018-19/63 DCBS.CO.PCB.Cir.No.1/18.01.000/2018-19 and Cyber Security Framework regulation.

Approach

Baseline ontology and corresponding dictionaries were automatically extracted from the documents in scope. Ontology for the regulation was extended by the expert. This was further extended to capture concepts and relations from circular and policy sequentially. This resulted in a unified, navigable ontology covering all the documents, backed by a dictionary.

Reference to a concept in any two documents indicated conceptual correlations in the content. Concepts present in regulation or circular but missing in policy document indicated potential gaps. Common references to relation <concept, relation, concept> indicated a higher likelihood of correlation between sentences. A traceability report on sentences corresponding to common/uncommon concepts and their relations enabled the identification of policy statements impacted by obligations listed in the regulation/circular.

Results

PoT approach was effectively able to detect if the regulation and circular had an impact on policy document. It was also able to correlate content from the regulation/circular to policy document on the basis of common and uncommon concepts and identify relevant policy content for selected regulatory sentences.

Benefits

The following benefits were observed:

- Automated detection of enterprise policies impacted by regulatory changes eliminating the need for manual scanning of regulation, circular and policy content.
- The solution enables the effective management of internal control environment by identifying new obligations missing appropriate controls or existing obligations with weakness in control for driving timely management intervention.
- Helps fasten the process of obligation/risk taxonomy creation through automations.
- Reusable ontology models that can be extended for new regulations, circulars, policies and other documents and revised for changes in these documents.
- Quantitative benefits can be stated precisely only when the same change management situation is addressed “with automation” and “without automation”. As such, a comparison is rather difficult, while working in real-life situation, we think the effort savings for “similar” situations are likely to be around 25% with the use of automation aids.

Summary and Future Work

Regulatory compliance is a critical need faced by enterprises today. With enterprises likely to operate in an increasingly regulated regime, this need will be felt even more acutely as the time window continues to shrink. We argued the current document-centric manual practice of regulatory compliance that relies on human experts is falling short of helping enterprises achieve the goal of staying compliant at optimal cost and with minimal exposure to risk. We also stated the limitations of existing RegTech like GRC frameworks and regulation-specific point solutions.

We proposed an AI-aided model-driven approach to regulatory compliance. The key intuition is to shift the focus from documents to models so as to bring in automation, thus reducing the analysis and synthesis burden on human experts. We presented automation support for authoring regulation model from regulation document in NL text, validating the authored model, transforming the regulation model to executable rules, populating the rule variables from enterprise data footprint and compliance checking. We showed the efficacy of the proposed approach and the associated technology infrastructure to address the needs of hygiene, compliance checking and change management in real-world industry-scale contexts.

Thus, it can be argued that the proposed approach is a significant step towards achieving the goal of “staying compliant with minimal risk exposure and optimal cost”. However, some challenges remain unaddressed.

Risk management: Cost of compliance needs to be weighed against the risk of non-compliance. This calls for a means to quantify risk. With policies, controls, processes and regulations available in model form, they can be simulated to play out what-if scenarios to quantify risk using real-world data [16].

Addressing non-compliance: Can AI techniques like Machine Learning [17] and Reinforcement Learning [18] help suggest measures to fix a specific non-compliance? This seems a far-away goal as of now.

Staying compliant: Given the high pervasiveness of computing in modern enterprises, the compliance remediation decision results in the modification of software systems. Model-centric nature of the proposed approach can help identify the software systems that need to undergo a change. Even the exact place where the change needs to be introduced can be identified if detailed model of software system is available. Proven MDE techniques can be used to effect these changes in an automated manner [19, 20].

Compliance dashboard and early warning system: By keeping the compliance checking process continuously running in the background, it will be easy to present a comprehensive view of compliance. Moreover, compliance digital twin, through what-if simulation, can predict an undesirable state ahead of time. Alerts can then be issued to the relevant stakeholders, thus helping them be prepared to handle the situation.

References

1. Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P., & Cory, H. T. (1986). The British Nationality Act as a logic program. *Communications of the ACM*, 29(5), 370–386.
2. Antoniou, G., Dimareisis, N., & Governatori, G. (2007). A system for modal and deontic defeasible reasoning. In *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia, December 2–6, 2007, Proceedings*.
3. Kerrigan, S., & Law, K. H. (2003). Logic-based regulation compliance-assistance. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law, ICAIL 2003, Edinburgh, Scotland, UK, June 24–28, 2003* (pp. 126–135).
4. Zeni, N., Kiyavitskaya, N., Mich, L., Cordy, J. R., & Mylopoulos, J. (2015). GaiusT supporting the extraction of rights and obligations for regulatory compliance. *Requirements Engineering*, 20(1), 1–22.
5. Smullyan, R. M. (1995). *First-order logic*. Courier Corporation.
6. Wright, V., & Henrik, G. (1951). Deontic logic. *Mind*, 60(237), 1–15.
7. Nute, D. (2001). Defeasible logic. In *International Conference on Applications of Prolog* (pp. 151–169). Springer.
8. Sunkle, S., Kholkar, D., & Kulkarni, V. (2016). Informed active learning to aid domain experts in modeling compliance. In *2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE.

9. Sunkle, S., Kholkar, D., & Kulkarni, V. (2016). *Comparison and synergy between fact-orientation and relation extraction for domain model generation in regulatory compliance* (International Conference on Conceptual Modeling). Springer.
10. Roychoudhury, S., et al. (2018). A case study on modeling and validating financial regulations using (semi-) automated compliance framework. In *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer.
11. Sunkle, S., Kholkar, D., & Kulkarni, V. (2015). Model-driven regulatory compliance: A case study of “Know Your Customer” regulations. In *18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MoDELS 2015, Ottawa, ON, Canada, September 30 – October 2, 2015* (pp. 436–445).
12. Kholkar, D., Mulpuru, D., & Kulkarni, V. (2018). Balancing model usability and verifiability with SBVR and answer set programming. In *MODELS Workshops* (pp. 570–573).
13. Sjjir Nijssen. SBVR: Semantics for business. 2007.
14. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., & Scarcello, F. (2006). The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3), 499–562.
15. Lifschitz, V. 2008. What is answer set programming? In *Proceedings of the Twenty Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, IL, July 13–17*.
16. Dimarasis, N. (2007). A system for modal and deontic defeasible reasoning. *International Journal of Cooperative Information Systems*, 14(2–3), 181–216.
17. Yeddula, R. R., Das, P., & Reddy, S. (2015). A model-driven approach to enterprise data migration. In J. Zdravkovic, M. Kirikova, & P. Johannesson (Eds.), *Conference on Advanced Information Systems Engineering (CAISE), Stockholm, Sweden* (Lecture Notes in Computer Science) (Vol. 9097, pp. 230–243). Springer.
18. Feldman, R., & Sanger, J. (2007). *The text mining handbook: Advanced approaches in analyzing unstructured data*. Cambridge University Press.
19. Kulkarni, V., Barat, S., & Clark, T. (2019). Towards adaptive enterprises using digital twins. In *2019 Winter Simulation Conference (WSC)* (pp. 60–74). IEEE.
20. Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260.



Tony Clark 

Introduction

An AI-Enabled Enterprise relies on Digital Twin(s) to achieve adaptation. In most cases, it is useful to be able to model the digital twin system to understand the key features and gain confidence that appropriate adaptation will be achieved. This chapter reviews different types of digital twin and associated implementation technology before introducing a modelling approach for Digital Twin(s) in terms of several case studies.

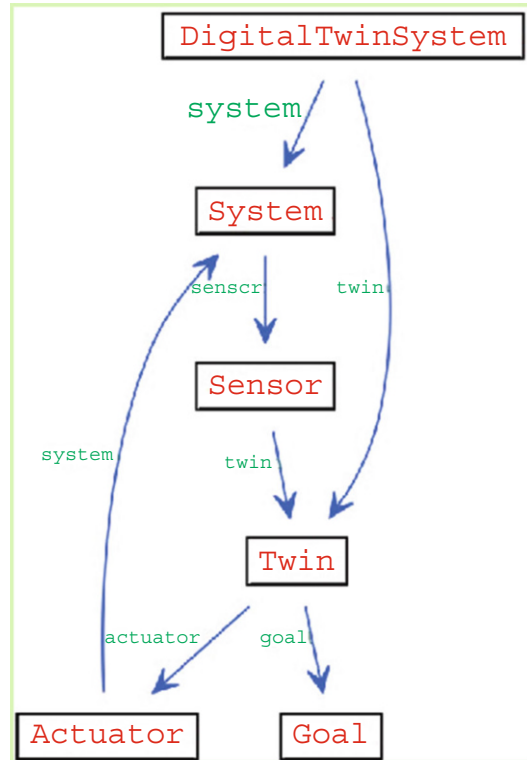
Digital Twin(s)

Figure 4.1 shows a typical digital twin architecture. There are several variations depending on the level of detail; however, most architectures involve a real system with an interface to a digital twin. The twin measures the system using sensors and controls it using actuators. Both the measurement and control are often partial meaning that the twin does not have complete knowledge of the state of the system and does not have complete control over its behaviour.

The twin is associated with a goal and tries to achieve the goal in terms of its measurements of the real system. Often, the goal may be to achieve optimal behaviour but may also be to achieve some sort of desirable final state. An interesting situation occurs when an algorithm is not available that will allow the twin to achieve the goal in terms of the controls. In this case, the twin must adapt.

T. Clark (✉)
Aston University, Birmingham, UK
e-mail: tony.clark@aston.ac.uk

Fig. 4.1 Digital twin architecture



In most cases, it is not possible to obtain historical data from the real system to adapt. This may be because the real system is not well understood (a black box) or because it is just too complex. In this situation, the twin must adapt dynamically as the real system executes.

Figure 4.1 is a very high-level conceptual view of a digital twin, and there may be many different variations on the terminology used and the activities of the key concepts. For example, the system may be a physical system, a digital system, a human system or a mix. Sensors and actuators may be automated or involve human intervention. A digital twin system may involve multiple Digital Twin(s) for the same system or multiple systems with their own collection of twins. The entire system may be tightly integrated or federated across a network and cloud-based technologies.

In all cases, a twin will contain a model of some aspect of the real system. The extent to which the model is in correspondence with the real system depends on the role of the twin system. Further, the information provided by the sensors and the level of control provided by the actuators will depend on what is offered by the real system. In extreme situations, the real system may not expose the detail of its inner workings, and a twin may have to build up a picture of the system incrementally from noisy and incomplete information.

There are several variations on the theme of Digital Twin(s) depending on the level of integration and automation:

Simulating A simulation does not connect the twin to a real system. There may be several reasons for this. A typical motivation for a *simulation twin* is to resolve design issues when building the real system. Another reason may be to try to understand the *as-is* or *to-be* states of the real system, particularly when maintaining or making system modifications. A simulation can be used to determine whether a particular system configuration will achieve the desired goals and to investigate configuration variations.

Shadowing A shadow receives sensor information from the real system to monitor its behaviour. A *shadowing twin* may be used to raise issues when the real system diverges from the expected behaviour; in this sense, it can advise when situations arise but cannot act on the basis of the advice.

Controlling A twin that monitors the real system via the sensors and performs control via the actuators is a *controlling twin*. Such a twin can be used to nudge a system back on track when it diverges from desirable behaviour. If the actuation involves a human, then the controlling twin can be used for *decision support* where several options for control are presented for the user to choose from. Typically, a controlling twin will have a fixed behaviour (often the ideal behaviour) which is measured against the information from the real system available via the sensors.

Adapting A twin that changes its behaviour to achieve a goal is an *adapting twin*. In comparison with a controlling twin, the adapting twin need not have a fixed behaviour – it may not know what the system should do. In such situations, we will refer to the behaviour of the twin as its *policy*. The twin must dynamically adapt in order to develop a policy that can be used to control the real system.

The variations above form a broad spectrum of sophistication in terms of twin behaviour. They are not intended to be entirely distinct, for example, a simulation may also adapt. However, the level of difficulty increases depending on the level of integration with a real system; the level of fidelity between the real system and the twin; the quality, quantity and frequency of communication between the twin and the system; whether the twin system is fully automated or relies on human intervention; and whether the twin policy is fixed and known in advance [3].

State of the Art

Adaptation in software systems relies on a feedback loop that monitors the system and, its environment and adapts the system against some specified goals [4]. The *autonomic control loop* shown in Fig. 4.2 [5] is equivalent to the MAPE (or MAPE-K) loop shown in Fig. 4.3 [6] since it collects data, compares data against expectations, decides how to intervene, and then applies actions. The specific technologies that are used to achieve these activities will depend on the application and the level of sophistication. The loop in Fig. 4.2 was designed by merging standard approaches from engineering control theory with knowledge-based control used in fields such as intelligent robotics and planning systems.

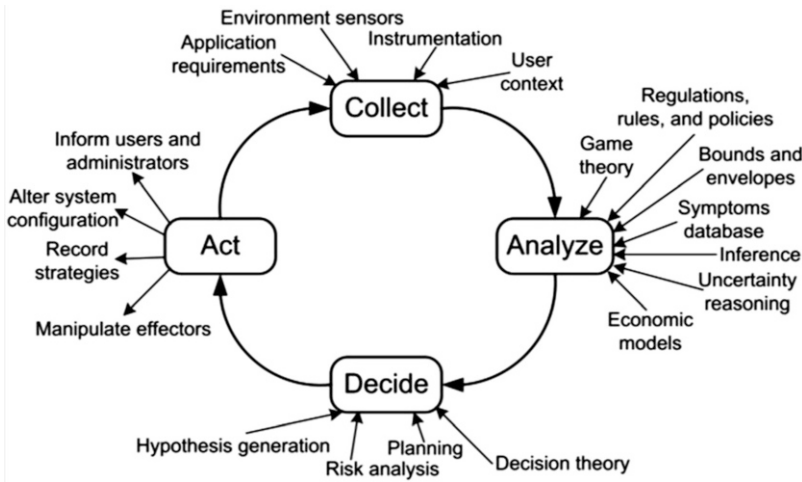
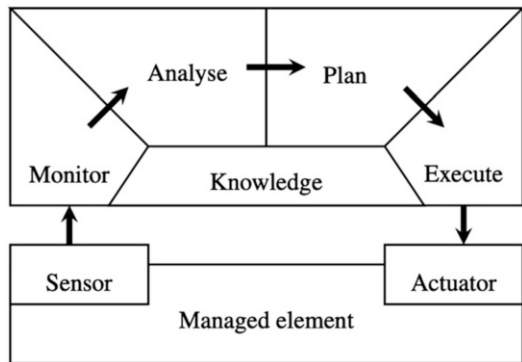


Fig. 4.2 Autonomic control loop [1]

Fig. 4.3 MAPE-K loop [2]



The engineering approaches of MIAC and MRAC both use *models* to implement the loop [4] which is key to a *model-driven* approach to Digital Twin(s) where a twin builds (MIAC) or maintains (MRAC) a model of the real system. Several types of model used in Digital Twin(s) are identified in MODA [7] in order to separate out the goals (prescriptive), the expectations (predictive) and the mirror of the system (descriptive). From this work, it is clear that there is scope for separating out the different types of models required by adaptive systems and Digital Twin(s) at both design time (when designing and prototyping twins) and at run-time (by processing the model dynamically).

Adaptation can be achieved through several technologies. Recent advances in Machine Learning provide an obvious choice for many types of non-trivial adaptation against goals [8]. Since Digital Twin(s) model aspects of real systems, the size of any state space associated with a twin policy is likely to be huge, and therefore, deep learning would seem appropriate. The MAPE-K architecture can be extended using Machine Learning techniques [9] as the adaptation mechanism leading to a variety of design and implementation questions regarding the correct ML technology to use.

A key feature of an adaptable digital twin is that it will typically adapt *dynamically* meaning that techniques for learning from existing data are not appropriate. Recent advances in Reinforcement Learning [10] are demonstrating that incremental learning techniques are appropriate for twins that need to discover policies. Reinforcement Learning can easily be included in the autonomic control loop [11] as shown in Fig. 4.4 where an additional reward stream provided by an *oracle* is added to the twin. The oracle monitors the controls and effects produced by the twin and measured from the system and associates the effects with a reward. The reward is used to fine-tune the choices made by the twin and thereby improve the internal policy.

Although there has been progress in identifying the architecture of Digital Twin(s), their control features, the types of models to be used and the technologies that might be used to achieve adaptation, little has been developed in terms of languages to express Digital Twin(s). As described in [12], such languages need to have explicit support for the

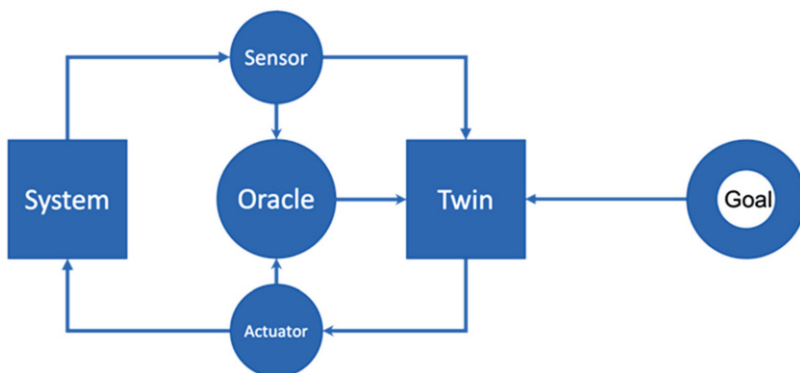


Fig. 4.4 Twin Reinforcement Learning

feedback loop that is intrinsic to adaptable systems and to provide explicit support for representing key features such as goals and adaptation mechanisms.

The rest of this chapter provides an approach to this challenge in the form of a language and associated toolset called TwinSim that provides dedicated support for Reinforcement Learning-based adaptation features that are required when designing and prototyping Digital Twin(s).

Modelling Twin Systems

The previous section has motivated the need for adaptive Digital Twin(s) within an AI-Enabled Enterprise. Several twin architectures are available together with adaptation technologies. However, there is a need for an approach for twin design. This section introduces an approach to the design of Digital Twin(s) leading to an executable twin simulation that can be used to provide confidence prior to implementation.

Case Study

Consider a system that processes jobs. Each job must be allocated to a *bucket* for processing. The processing takes a specific amount of time, and each bucket has a maximum capacity.

The system is *dumb* in the sense that it has no intrinsic knowledge of how to allocate jobs so that no bucket overflows. Our challenge is to construct a digital twin of the system to control it. In terms of the architecture shown in Fig. 4.1, we have:

Sensor The system informs the twin as to whether a bucket has exceeded capacity.

Actuators The twin allocates a new job to a bucket.

The task of the twin is to use the sensors and actuators to adapt to achieve the goal of preventing any overflow. The twin uses a model of the system to achieve this goal by learning two system parameters:

Capacity The capacity of each bucket

Duration The duration of processing for any job at a given bucket

The digital twin system can be modelled as shown in Fig. 4.5. A state of the digital twin system is either in sensor or actuator mode depending on whether the twin is receiving data from the system or controlling it. The twin system has a reference to the real system (which we need to model the real behaviour). However, the twin is separate and has no internal

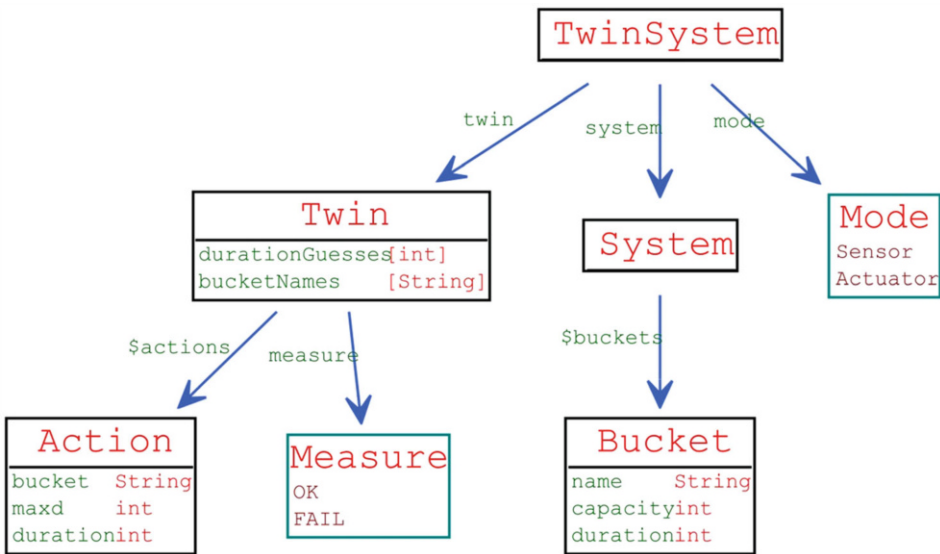


Fig. 4.5 Allocating packages: model

knowledge of the system. The system has a collection of buckets, each of which has a name, a capacity and a duration. The task of the twin is to learn the capacity and duration values of each bucket through behavioural adaptation. The twin contains a collection of duration guesses and bucket names to choose from when allocating jobs. It receives a measure of OK or FAIL from the sensor based on the state of the real system. At any time, it has a sequence of actions which describe the current state of job allocation. The system is informed of each new job allocation via the actuator.

Each time the actuator requests an action from the twin, a new job is allocated. While the twin does not have a policy that is consistent with the goal, it must choose actions at random based on knowledge of the names of available buckets and the possible job duration.

Twin System Execution

Execution involves making transitions based on the model shown in Fig. 4.5 and proceeds by alternating two phases:

Actuator Create an action based on the current policy defined by the twin. At the start of the execution, the policy is unknown; therefore, actions are created at random. Therefore, many of the actions will violate the goal, although some will be OK. As the twin adapts, the policy starts to develop, and the actions are more likely to satisfy the goal.

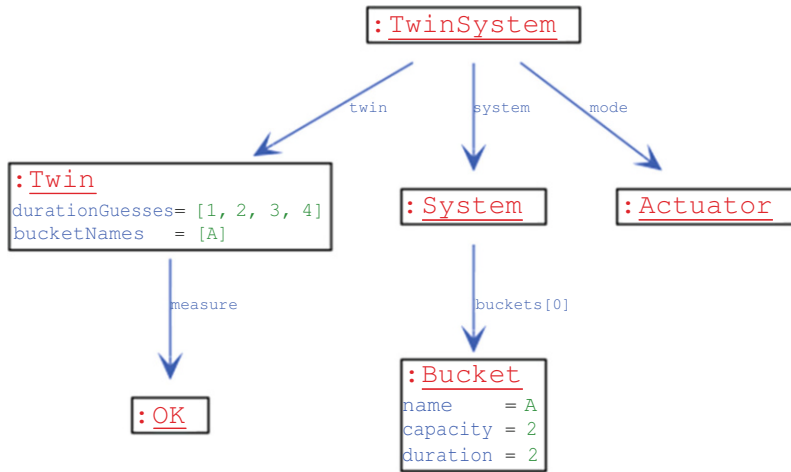


Fig. 4.6 Allocating packages: a simple starting state

Sensor Actions that are sent to the system may cause the system constraint to be violated either by allocating a job to a bucket with an inappropriate duration or by allocating a job to a bucket that is already full. The twin receives either OK or FAIL from the system via the sensor.

The simplest twin system involves a single bucket as shown in Fig. 4.6 where bucket A has a capacity of two jobs and a duration of two for each job. The twin knows that there is a single bucket but does not know the capacity or duration. It will guess the duration by selecting numbers from 1 to 4.

Figure 4.7 shows what happens when the twin is used without any training. Step 0 is the initial state of the system. At step 1, an action is allocated, and the mode of the twin system is set to Sensor. The system feeds back that the action is acceptable in step 2. Steps 3 and 4 repeat this where a second action is created which is acceptable. However, at step 5, a new action is created which has a duration of four which the sensor reports as unacceptable in step 6.

The twin can make some observations about the execution shown in Fig. 4.7. Firstly, several actions were allocated that the system reported as being acceptable. The twin can record these successful actions as being more likely to achieve success in the future. Secondly, step 6 shows an action that is unacceptable. The twin can record this and avoid making the same choice in the future. Over time, this will lead to a *twin policy* where the choices tend towards success and tend to avoid failure.

The twin system has arrived at an undesirable state in step 6. Depending on whether we are training the twin before deployment or dynamically training it will determine the next

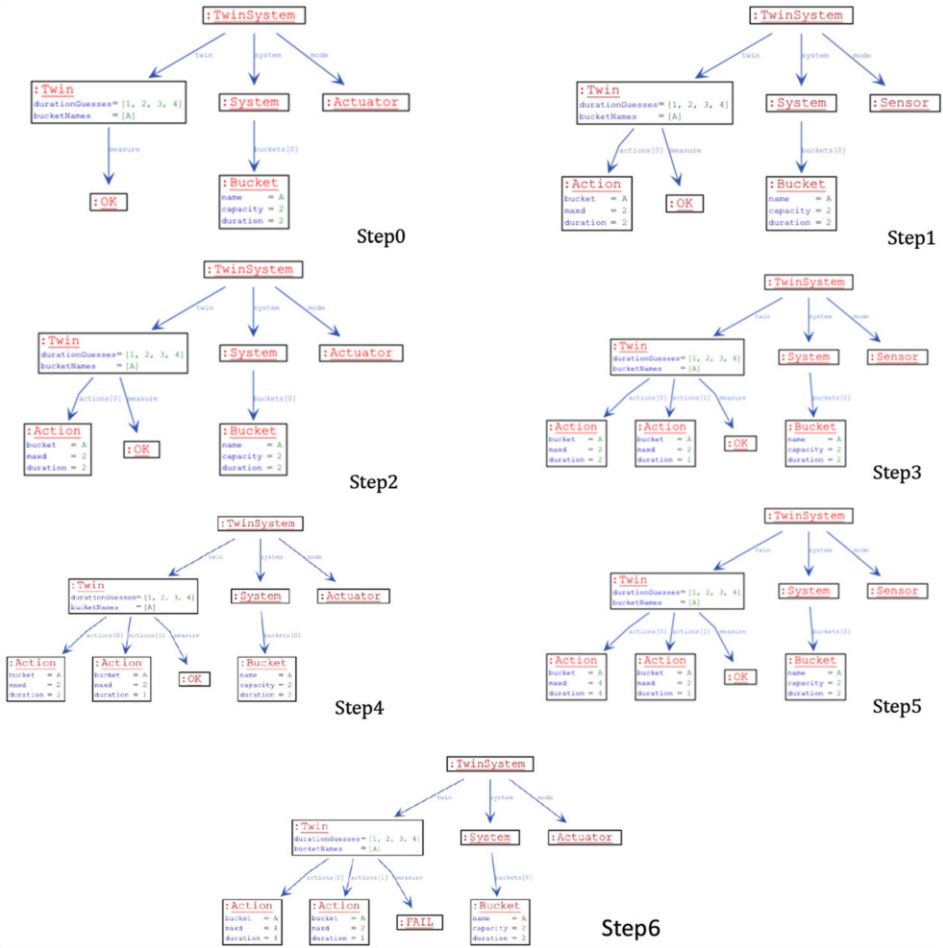


Fig. 4.7 A digital twin with adaptation

steps. In certain twin systems, creating an undesirable state is *terminal*, and the whole system must be reset. This is likely to be the case where the twin is trained a priori. Alternatively, the real system may *fail-safe* (perhaps we are trying to adapt to optimize something) in which case execution continues.

Twin Policies

Adaptation in the twin occurs by building up a *policy*. The policy is a graph that consists of nodes labelled by twin states and edges that are labelled with numbers. The policy can be

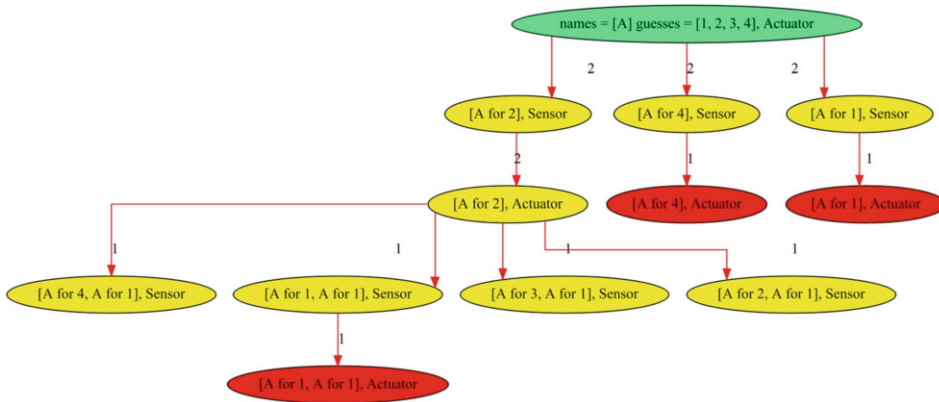


Fig. 4.8 A partial policy

used by finding the node labelled with the current state of the twin and then selecting the edge with the highest value. The next state of the twin is the label of the target node.

The nodes of the policy have three modes: *initial* which is coloured green, *intermediate* which is coloured yellow and *terminal* which is coloured red.

A policy is constructed incrementally. Figure 4.8 shows the policy of the job allocation twin as it is constructed. The initial node is labelled with the names, guesses and mode of the twin. All other nodes are labelled with the current actions.

Figure 4.9 shows a fully trained policy for the twin. Note that it contains a cycle and can therefore support twin system that runs forever.

Consider a twin system that contains four buckets as shown in Fig. 4.10. The twin can choose to create an action that allocated to any bucket each step. In addition, there are seven different job durations to choose from each time.

After the twin has adapted to the hidden parameters, it produces a policy as shown in Fig. 4.11. The interesting thing about this policy is that it satisfies the goal: all jobs are allocated, and no bucket ever overflows, but *the system does not require all four buckets*. In this case, the adaptation performed by the twin has achieved two things: (1) found a suitable policy and (2) caused the system to be optimized with respect to the number of buckets needed.

A technology that implements an adaptable twin is not limited to finding an appropriate policy. The technology can be used to perform experiments, by varying the system parameters in order to find a system configuration for which a policy exists and which satisfies some goal.

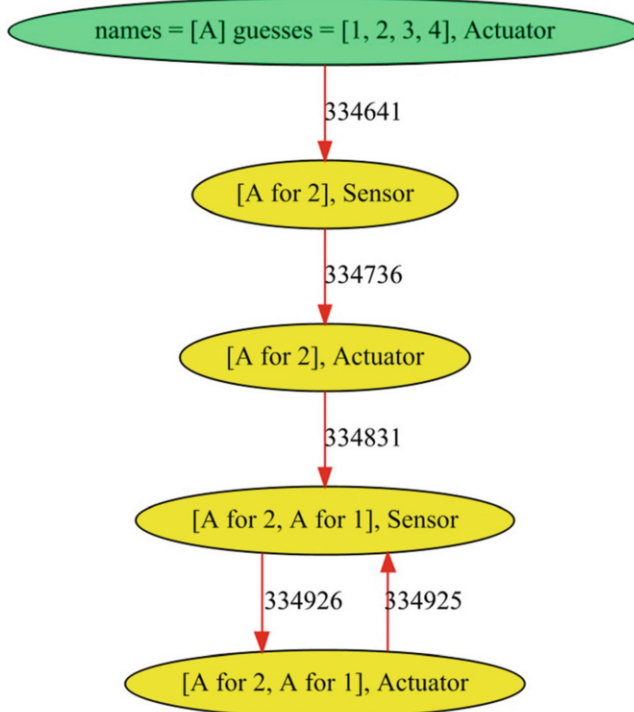


Fig. 4.9 A complete policy

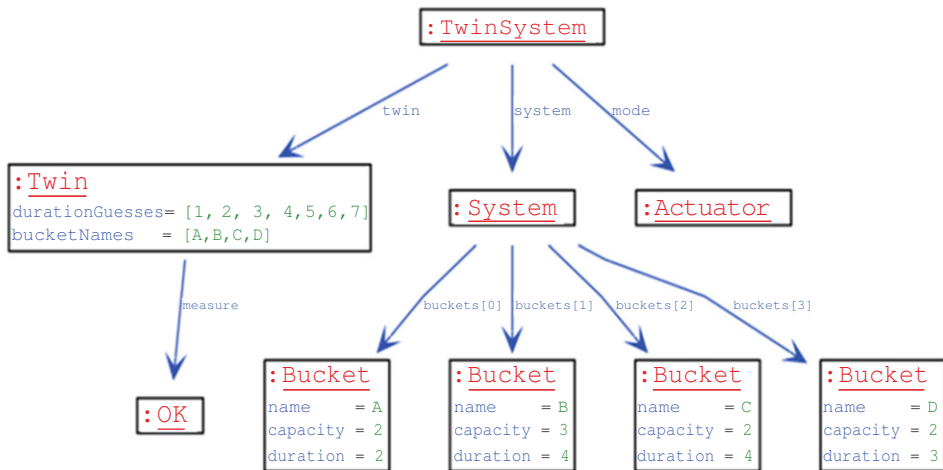


Fig. 4.10 Allocating packages: a starting state with four buckets

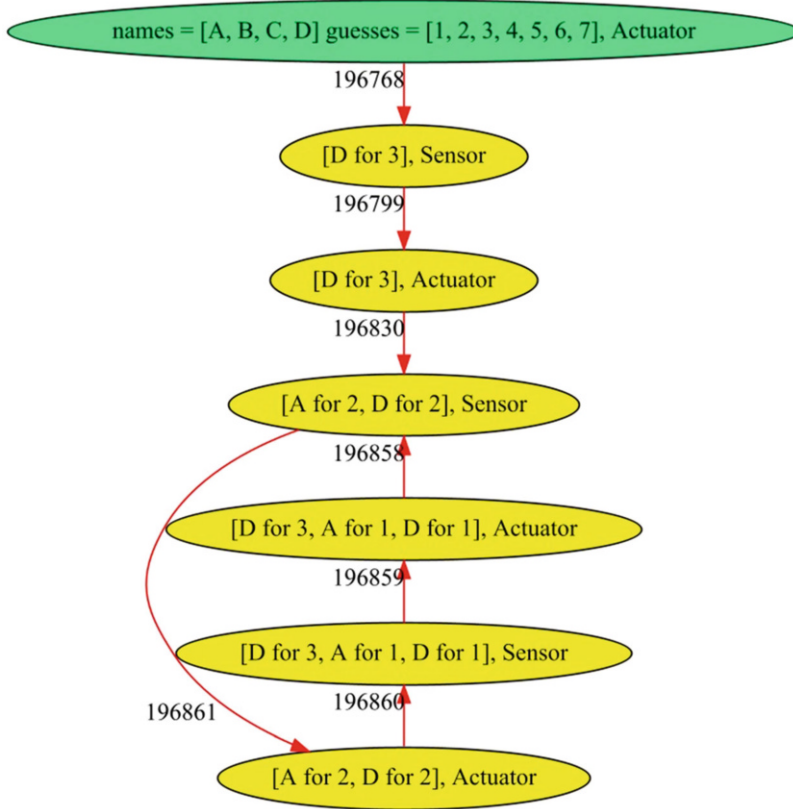


Fig. 4.11 A twin policy for four buckets

Implementation: TwinSim

TwinSim is a domain-specific language for modelling digital twin systems. The diagrams in the chapter have all been produced by TwinSim.

As noted above, Reinforcement Learning is a suitable technology that can be used to achieve twin adaptation. This is because it is not often possible to produce training data from the real system; therefore, the data used to adapt the twin is incrementally produced by sensors attached to the real system as shown in Fig. 4.1. A domain-specific language provides the following features:

State Model The execution of the digital twin system proceeds in using state transitions. Each state is an instance of a state model. An example of a state model is given in Fig. 4.5. The TwinSim language offers two constructs to define the state model:

Terms A term-definition is like a class in a UML model or a Java program. It introduces a data type with associated fields. A term-definition has the form:

```
term n(field-name:field-type,...) {
  fun fun-name(fun-arg,...) = fun-body;...
}
```

The short form of a term-definition omits the functions and the curly braces. The very short form can omit the field definitions if the term-definition has only one instance. An example of a term-definition is:

```
term Action(bucket:str,maxd:int,duration:int);
```

A term-value is created using the name of the term-definition and supplying arguments of the appropriate type:

```
a = Action("A",3,3)
```

The fields of a term value can be referenced using pattern-matching or using “.”, for example `s.buckets` where `s` is a term of type `System`.

Unions A union-definition introduces a data type and a collection of term-definitions. An instance of any of the term-definitions can be used when a value of the union data type is expected. A union-definition has the form:

```
union n { term-name(field-name:field-type,...); ...
}
```

An example of a union-definition is:

```
union Mode { Sensor; Actuator; }
```

Levers A lever is a value that controls the digital twin system and which may be changed at the start of the simulation or at any time during the simulation. Typically, a lever will be specified as a value definition in the form: `val name:type = value;`

The following are two examples of value definitions:

```
val theBuckets:[Bucket] = [bucketA,bucketB,bucketC,bucketD];
val theDurations:[int] = [1,2,3,4,5,6,7];
val theBucketNames:[str] = [ b.name | b ← theBuckets];
```

Functions Functions are defined in order to perform tasks as part of the simulation. A function-definition has the form:

```
fun name:(arg-type,...) → result-type fun n(arg-pattern,...) = fun-body; ...
```

The first line introduces the type of the function, and the subsequent lines provide pattern-directed cases for its definition. When the function is applied to argument values, each of the cases is tried in turn until all of the arg-patterns match the values, in which case the function body provides the return value. An example is shown below:

```
sensor:(TwinSystem) → TwinSystem;
fun sensor(TwinSystem(Twin(as,_,g,ns),s,m)) =
  TwinSystem(Twin(as,OK,g,ns),s,m) when actionsOK(as); fun
sensor(TwinSystem(Twin(as,_,g,ns),s,m)) =
  TwinSystem(Twin(as,FAIL,g,ns),s,m);
```

State Machine A digital twin system executes using a state machine. The states are instances of term-definitions, and the transitions are defined by rules. A state machine has the form:

```
name:(type) → type;
machine name:type { rewardClause... terminalClause...
  constraintClause... ruleClause... initialClause
  learnClause training-parameter...
}
```

A state machine can be thought of as a function that maps a term-value to a term-value. The mapping is defined by the machine's policy which is initially empty. The various clauses within the machine definition (explained below) define how the policy is populated using Reinforcement Learning.

Rewards A reward is a mapping from a term-value to a real number that describes how well the value matches the goal of the machine. Typically, attractive values (machine states) will be mapped to higher values than less attractive values. The reward clauses of a machine definition are implemented to cover the goal of the digital twin. A reward clause has the form:

```
reward name {
  pattern → real-value
}
```

The following are examples of reward clause definitions:

```
reward ok {
  TwinSystem(Twin(_,OK,_,_),_,_) → 1.0
}
reward fail {
  TwinSystem(Twin(_,FAIL,_,_),_,_) → 0.0
}
```

Constraints The state model of the twin system has semantics that are defined by constraints which are conditions that hold at any time for the machine state. A constraint-clause is a pattern that must match the state: `constraint name { pattern }`

The following constraint requires the system to be correct, i.e. no bucket has exceeded capacity, and all jobs are allocated to buckets for the appropriate duration:

```
constraint legal {
    TwinSystem(Twin(_,OK,_,_),_,_)
}
```

Initial States Execution of a machine starts in one of a number of initial states. Once started, the execution may be intended to go on forever or may aim to reach a terminal state. The initial states are specified as a set: `learn set-expression`

For example, the following specifies a single starting state for the case study:

```
learn {
    TwinSystem(
        Twin([],OK,theDurations,theBucketNames),
        System(theBuckets),Actuator)
}
```

Terminal States The machine may have terminal states which, if reached, cause the machine to stop. There are two situations where this might occur: (1) when the state becomes illegal or (2) when the state becomes a target state. Terminal states are defined using terminal clauses of the form:

```
terminal name {
    pattern
}
```

For example, there is no point in continuing past the point when the system has failed:

```
terminal bad {
    TwinSystem(Twin(_,FAIL,_,_),_,_)
}
```

Transition Rules The machine is executed using pattern-directed transition rules. At any time, the machine is in a current state. An execution step is performed by matching the current state against all transition rules that are satisfied and creating a new state based on the target of the rule. This occurs *for all possible rules that are satisfied*. For each rule that is satisfied, there may be multiple ways in which the rule can apply to the current state and the rule may specify multiple possible target states. In that case, *all* target states are

created and added to the current policy, and the best choice is made. If the policy does not specify the best choice, then a target state is chosen at random. A rule has the following form:

```
rule name {
    pattern -exp
}
```

The following two rules define the sensor phase of a machine. They are mutually exclusive, so just one of them will apply:

```
rule sensorOK {
    TwinSystem(Twin(as,s,g,ns),sys,Sensor) →
        TwinSystem(Twin(as,OK,g,ns),sys,Actuator) when actionsOK(as)
}
rule sensorFAIL {
    TwinSystem(Twin(as,s,g,ns),sys,Sensor) →
        TwinSystem(Twin(as,FAIL,g,ns),sys,Actuator) when not(actionsOK(as)) }
```

The following example uses `select` to select an element at random from the bucket names and the durations. Therefore, an actuator phase of the machine will allocate a job to a bucket at random and for a randomly selected duration:

```
rule actuator {
    TwinSystem(Twin(as,_,guesses,ns),s,Actuator) → let b = select(ns) d =
        select(guesses) in
        let as = [ Action(b,m,n-1) | Action(b,m,n) ← as, ?n > 1 ]
        in TwinSystem(Twin([Action(b,d,d)] ++ as,OK,guesses,ns),s,Sensor) }
```

When the `actuator` rule is used, the policy is expanded with all the possible state transitions, and the best is chosen based on the constraints and rewards defined for the machine.

Training Parameters Where a policy is created a priori, the machine must be configured and executed multiple times. Each execution consolidates the machine's policy. The training parameters are used to control how long each execution should last (`steps`) and how many executions there should be (`epochs`). The policy can be sampled during the executions to display a graph (`trace`). Reinforcement Learning is performed in terms of *exploration* and *exploitation*, which is controlled using `decay`, and in terms of parameters `alpha` and `gamma`.

Figure 4.12 shows the result of training the case study machine. The y-axis shows cumulative rewards for each epoch which is represented on the x-axis. It demonstrates a typical profile arising from the trade-off between exploration and exploitation. The rewards

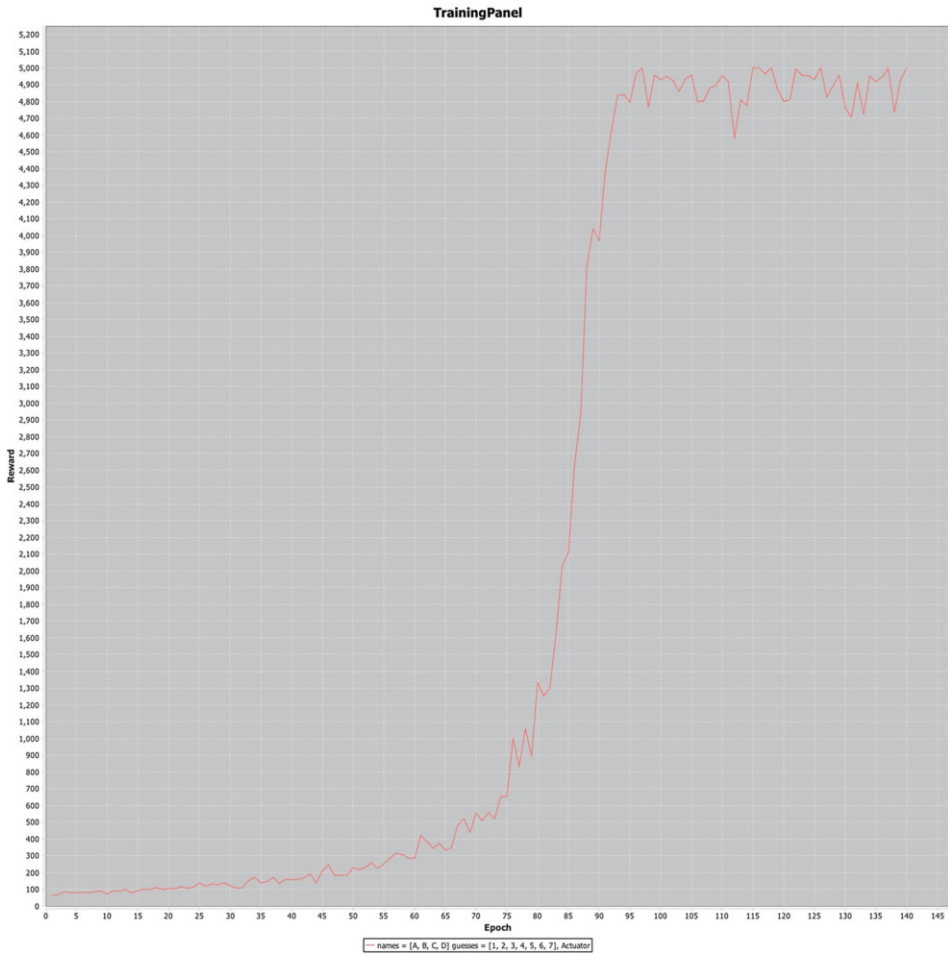


Fig. 4.12 A training profile

are low towards the left of the graph where the machine is performing a high degree of exploration and often *getting it wrong*. Towards the right of the graph, the machine is exploiting the policy that is emerging and therefore tending to *get it right*. After a certain point, the policy has reached a fixed point (although a limited amount of exploration still occurs).

The complete implementation of the bucket case study is shown below:

```

import "Lists.sys";
term System(buckets:[Bucket]); term Twin(actions:[Action],measure:Measure,
durationGuesses:[int],bucketNames:[str]) {
  fun toString() = actions;
}
term TwinSystem(twin:Twin,system:System,mode:Mode) {
  fun toString() =
    case twin.actions {
      [] → "names = " + twin.bucketNames +
          " guesses = " + twin.durationGuesses +
          ", " + mode;
      _ → twin + ", " + mode;
    };
}

term Action(bucket:str,maxd:int,duration:int) {
  fun toString() = bucket + " for " + duration;
}
Union Measure { OK; FAIL; }
union Mode { Sensor; Actuator; }
term Bucket(name:str,capacity:int,duration:int);

val bucketA:Bucket = Bucket("A",2,2);
val bucketB:Bucket = Bucket("B",3,4);
val bucketC:Bucket = Bucket("C",2,4);
val bucketD:Bucket = Bucket("D",2,3);

val theBuckets:[Bucket] = [bucketA,bucketB,bucketC,bucketD];
val theDurations:[int] = [1,2,3,4,5,6,7]; v
al theBucketNames:[str] = [ b.name | b ← theBuckets];
val theSystem:System = System(theBuckets);
val theTwin:Twin = Twin([],OK,theDurations,theBucketNames);
val theTwinSystem:TwinSystem = TwinSystem(theTwin,theSystem,Actuator);

getBucket:(str) → Bucket;
fun getBucket(name) =
  find b ← theBuckets { b.name = name } then b
  else "no bucket named " + name;

actionsOK:([Action]) → bool;
fun actionsOK(actions) =
  forall a ← actions {
    a.maxd = getBucket(a.bucket).duration } and forall b ← theBuckets {
    bucketOK(b,actions) };

bucketOK:(Bucket,[Action]) → bool; fun bucketOK(b,actions) = #[[ 1 | Action(n,m,_) ← actions,?n = b.name ]]
  <= b.capacity;

```



```

perform:([Action]) → [Action]; fun perform(as) = [ Action(b,m,n-1) | Action(b,m,n) ← as, ?n > 1 ];
allocator:(TwinSystem) → TwinSystem;
machine allocator:TwinSystem {

  reward ok {
    "stay OK."
    TwinSystem(Twin(_,OK,_,_),_,_) → 1.0
  }

  reward fail {
    "avoid fail."
    TwinSystem(Twin(_,FAIL,_,_),_,_) → 0.0
  }

  terminal bad {
    TwinSystem(Twin(_,FAIL,_,_),_,_)
  }

  constraint legal {
    TwinSystem(Twin(_,OK,_,_),_,_)
  }

  rule actuator {
    TwinSystem(Twin(as,_,gs,ns),s,Actuator) → let b = select(ns) d =
      select(guesses)
    in TwinSystem(Twin([Action(b,d,d)] ++ perform(as),OK,gs,ns),s,Sensor)
  }

  rule sensorOK {
    TwinSystem(Twin(as,s,g,ns),sys,Sensor) →
    TwinSystem(Twin(as,OK,g,ns),sys,Actuator) when actionsOK(as)
  }

  rule sensorFAIL {
    TwinSystem(Twin(as,s,g,ns),sys,Sensor) →
    TwinSystem(Twin(as,FAIL,g,ns),sys,Actuator) when not(actionsOK(as))
  }

  learn {
    TwinSystem(Twin([],OK,theDurations,theBucketNames),theSystem,Actuator)
  }

  train decay = 0.9999; train steps = 100; train epochs =
  7000; train trace = 50; train gamma = 1.0; train alpha
  = 1.0;
}

```

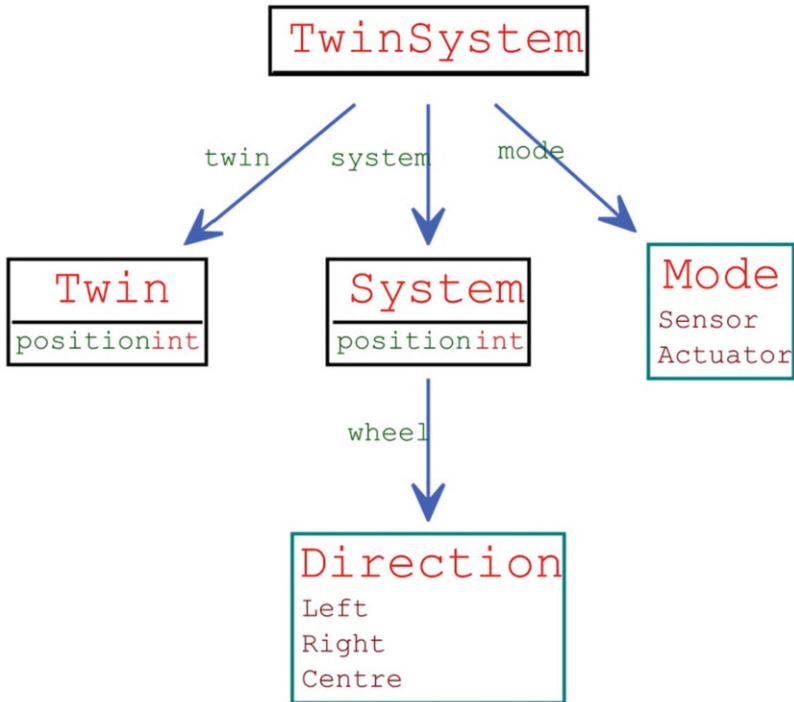
Training for Multiple Eventualities

A twin may have a goal to ensure that a system maintains a state within given parameters. The twin cannot directly update the state of the system and must use the actuator controls in order to *nudge* the system in the right direction. The twin must adapt in order to achieve a policy that uses the controls in the right way.

Consider a self-driving car (Fig. 4.13) system whose aim is to stay in the centre of a lane. If the road is straight and flat, then the car simply continues in its current direction. However, the road may bend and may present physical issues that must be addressed by modifying the current direction. A digital twin can learn a policy that controls the car. Sensors are used to measure the current distance to the centre of the lane, and actuators are used to change the position of the steering wheel. A simple digital twin system for the car is shown in Fig. 4.14. The twin can set the position of the driving wheel which in turn will affect the change in lane position.

Fig. 4.13 A self-driving car

Unlike the job-processing twin, the car twin must deal with changes in the car state that are not controlled by the actuator. These changes occur when the car encounters bends in the road or other physical challenges. In this case, the environment forces changes to the system state that must be addressed by the twin through adaptation.

**Fig. 4.14** A car digital twin

```

term System(wheel:Direction,position:int); union Direction { Left;
Right; Centre; } term Twin(position:int);
term TwinSystem(twin:Twin,system:System,mode:Mode); union Mode { Sensor; Actuator;
}

val maxPosition:int = 2;
val thePositions:[int] = (0-maxPosition)..(maxPosition+1);
val theDirections:[Direction] = [Left,Right,Centre];

selfDrive:(TwinSystem) -> TwinSystem;
machine selfDrive:TwinSystem {

  reward stayCentral { TwinSystem(Twin(0),_,_) - 1.0 }
  reward avoidNonCentral { TwinSystem(Twin(_,_,_) - -1.0 }
  terminal bad { TwinSystem(Twin(pos),_,_) when abs(pos) > maxPosition }

  constraint legal {
    TwinSystem(Twin(pos),_,_) when abs(pos) <= maxPosition
  }

  rule actuator {
    TwinSystem(Twin(pos),System(_,pos),Actuator) ->
      TwinSystem(Twin(pos),System(select(theDirections),pos),Sensor)
  }

  rule sensorRight {
    TwinSystem(Twin(_,System(Right,pos),Sensor) ->
      TwinSystem(Twin(pos+1),System(Right,pos+1),Actuator)
  }

  rule sensorLeft {
    TwinSystem(Twin(_,System(Left,pos),Sensor) ->
      TwinSystem(Twin(pos-1),System(Left,pos-1),Actuator)
  }

  rule sensorCentre {
    TwinSystem(Twin(_,System(Centre,pos),Sensor) ->
      TwinSystem(Twin(pos),System(Centre,pos),Actuator)
  }

  learn {
    TwinSystem(Twin(pos),System(dir,pos),Actuator) | pos - thePositions, dir -
      theDirections
  }
}

```

Fig. 4.15 A car digital twin system

To deal with such changes, Reinforcement Learning must be presented with all possible states that could be encountered. The resulting policy will contain all the possible states, and we can jump around the resulting state machine at will because of the values received from the sensor.

A digital twin system for a car is modelled in TwinSim as shown in Fig. 4.15. This is clearly a highly simplified model which elides many physical measurements provided by car sensors and represents the single measure of *position* in a simple discrete form where the real system would provide measures from a continuous data domain.

The key phases of the machine are:

Actuator The twin can command the car to move the steering wheel `Right`, `Left` or `Centre`. The twin will adapt by learning a policy that chooses the correct command.

Sensor The twin is provided with the current position of the car. The position is represented as an integer on a scale of $-\text{maxPosition}$ to maxPosition where 0 is the centre of the lane.

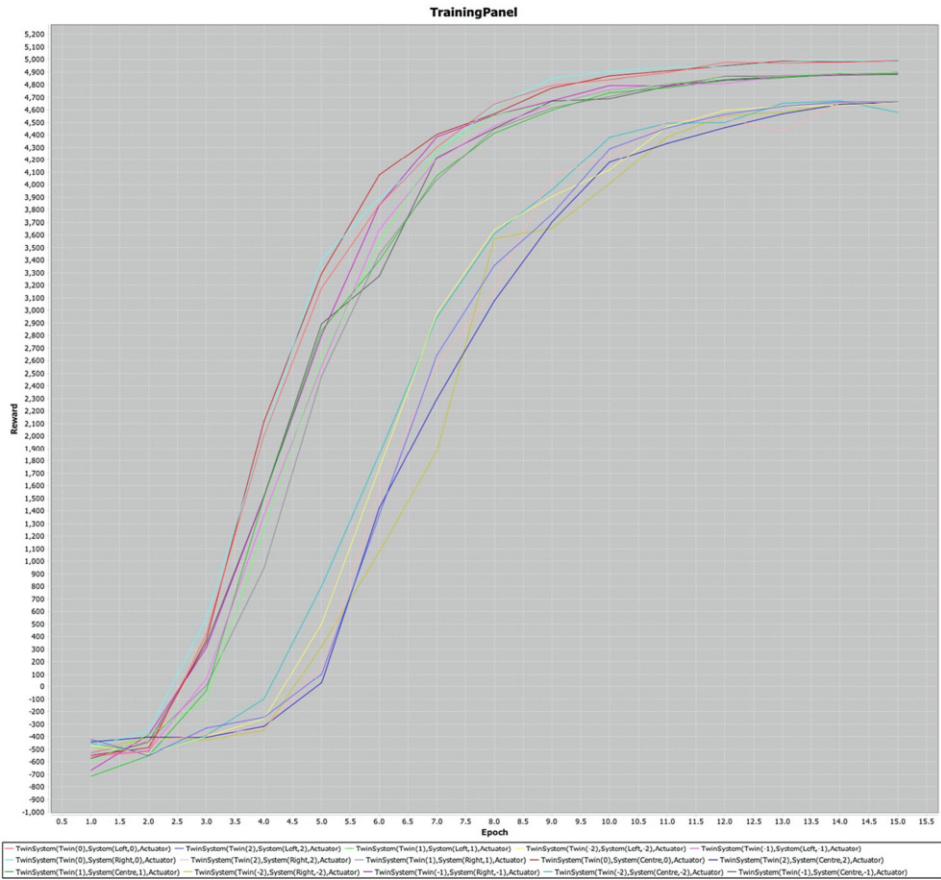


Fig. 4.16 Training the car twin

The machine is rewarded positively when the twin senses that the car is in the centre of the lane and is rewarded negatively otherwise. Training will stop when the car leaves the lane as defined by the terminal and the constraint-clauses.

In order to deal with all eventualities, the machine is trained using starting states that cover all possible car positions:

```
learn {
  TwinSystem(Twin(pos),System(dir,pos),Actuator) | pos ← thePositions, dir ←
  theDirections
}
```

Figure 4.16 shows the result of training against all the initial states leading to the policy shown in Fig. 4.17. Note that the policy guides all states towards the centre and then keeps it there. Of course, the car system may cause the twin to jump around the states as physical

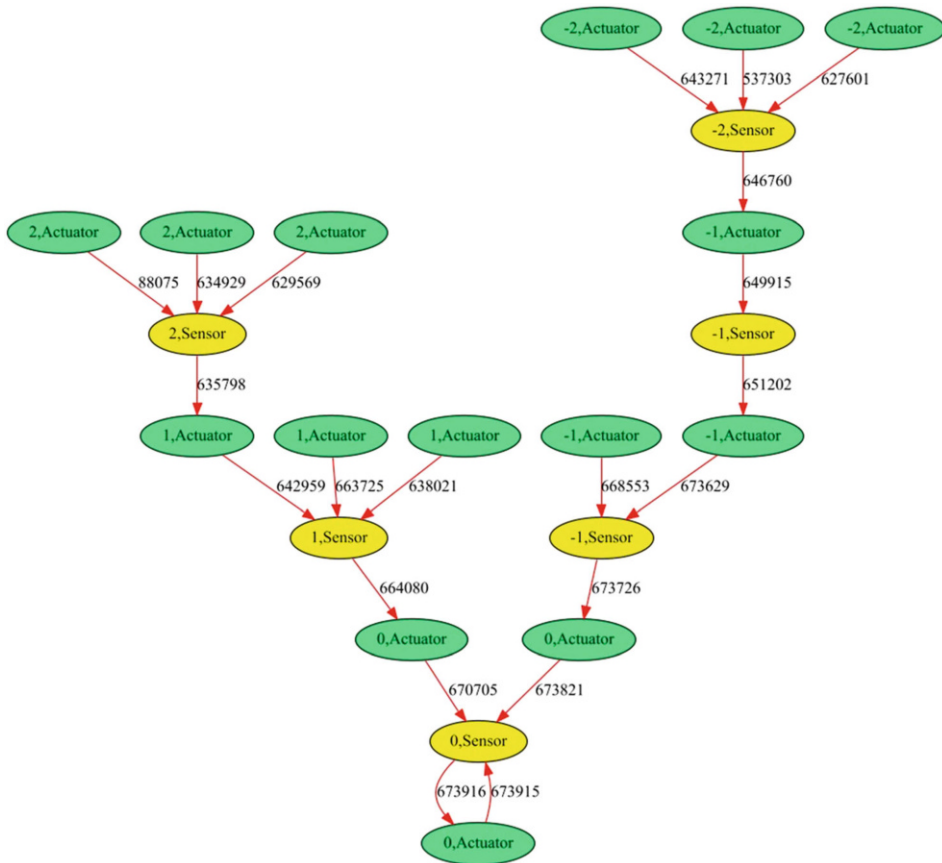


Fig. 4.17 Car twin policy

challenges are encountered, but the twin will ensure that the car always tends towards the centre of the lane no matter which state it ends up in.

Prototyping as Part of the Development Process

The AI-Enabled Enterprise relies on dynamic adaptation using knowledge about the domain and goals. Figure 4.18 shows the adaptive architecture for such an enterprise which relies on a model of that part of the system that is used to compare the current state with the ideal state as represented by the model.

To achieve an AI-Enabled Enterprise based on the architecture shown in Fig. 4.18, the learning must operate *at scale* since the state space of the model is likely to be huge. Perfect

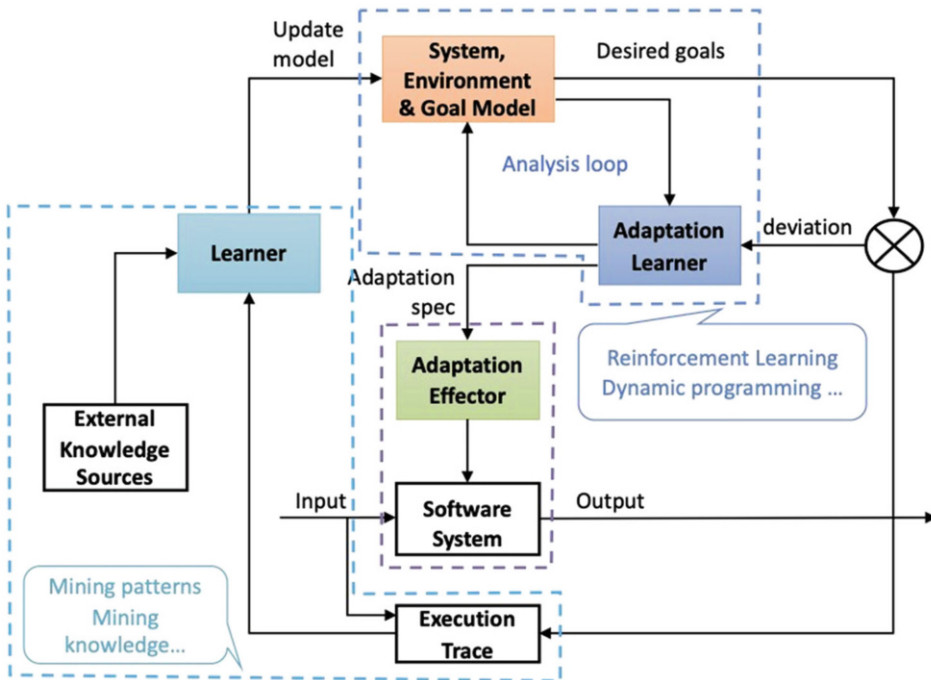


Fig. 4.18 Adaptation architecture

adaptation at such a scale is not possible due to the state space explosion: it is equivalent to learning a complete state machine that represents a function.

In order to be realistic, the state machine (and therefore the function) must be approximated using techniques such as deep learning based on neural networks. In this case, the function is approximated using feedback loops and weights within a network. However, direct use of this approach raises significant problems developing the resulting AI-Enabled Enterprise:

Semantics In order to use deep learning, the domain must be encoded in a way that is suitable for the network. This inherently loses information which is difficult to recover except in terms of the outputs of the network.

Verification Given the loss of semantics, it is difficult to analyse the resulting adaptive system in terms of completeness and consistency with respect to the goal. Furthermore, intermediate states of neural networks are not particularly useful and cannot be mapped onto intermediate state machines.

Explanation Explainable Machine Learning is an active research area which aims to provide feedback on *why* a given situation has arisen. The loss of semantics means that this is difficult for neural networks.

Given these drawbacks, it is useful to be able to prototype the key features of adaptation that will be used to drive the AI-Enabled Enterprise. TwinSim, as described in this chapter, is an approach that can be used to address the issue. TwinSim uses an algebraic representation of system states and pattern-directed rules to perform system execution. The resulting system is therefore *complete* in the sense that no information is intrinsically lost and is amenable to analysis including consistency checking. The system can be run to produce intermediate state machine that can be inspected and interrogated using queries. Execution runs from different initial states can be compared.

A drawback of the TwinSim approach is that it does not scale. However, our proposal is that many applications can be decomposed and reduced in terms of their complexity so that they can be expressed as a collection of representative independent adaptable subsystems, each of which can usefully be prototyped and analysed using the TwinSim approach.

Research Roadmap

As software systems grow in sophistication and connectivity, it is increasingly challenging to dictate a fixed behaviour prior to deployment. Thus, systems must *adapt* post-deployment against goals that may be fixed, but also may change over time. The mainstream engineering of adaptable software systems is a relatively recent topic as described in [13] where the authors conclude by listing key challenges that must be addressed.

A suitable architecture must be chosen when designing for adaptation that is supported by a suitable adaptation technology. The leading architecture is MAPE-K associated with Machine Learning; however, this raises questions about language support for MAPE-K, dealing with heterogeneity in large systems, and which style of Machine Learning to adopt. As systems become large and complex, agent-based architectures might be more appropriate leading to issues about distributed Machine Learning including game theory including adversarial approaches [9].

Given that adaptive software tends to address issues of uncertainty, there is a challenge regarding achieving the appropriate level of quality and in particular verification of such systems [14]. Digital Twin(s) typically involve some level of simulation which must be validated through analysis. The construction of adaptive systems lacks a methodology. Figure 4.18 shows the key concepts that are required at the heart of an AI-Enabled Enterprise; the question arises: how are these concepts acquired and developed into a system? Therefore, within the context of an AI-Enabled Enterprise, the following key research questions should be addressed to achieve adaptation:

Conceptual model What are the underlying concepts that must be represented to build such a system?

Application variations What is the space of purposes to which the adaptation can be applied? Decision support and error identification are examples of such variations. Can these variations be represented as a product line?

Architecture What is the architecture space for such adaptive systems? The MAPE-K architecture is an example.

Method What is the method by which such a system can be designed, built and deployed? Given the application area, there is likely to be heavy reliance on human knowledge; therefore, are there specific roles and knowledge acquisition techniques to be used?

Scale How to manage the state space associated with real-world adaptive systems?

Validation How to check that the model relates to the real-world system? How to check that the model of the goal is that which we want to achieve?

Verification How to check that the adaptive system will converge on the goal? Can executions of the system be interrogated and is there a way of determining if the system is complete?

Technology What are the appropriate technology platforms that can be used to implement the adaptive system?

References


1. Dobson, S., Denazis, S., Fernandez, A., Gaiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., & Zambonelli, F. (2006). A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2), 223–259.
2. Rutten, E., Marchand, N., & Simon, D. (2017). Feedback control as MAPEK loop in autonomic computing. In *Software engineering for self-adaptive systems III* (pp. 349–373). Assurances.
3. Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29, 36–52.
4. Brun, Y., Di Marzo, G., Serugendo, C. G., Giese, H., Kienle, H., Litoiu, M., Muller, H., Pezze, M., & Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In *Software engineering for self-adaptive systems* (pp. 48–70). Springer.
5. de Lemos, R., Giese, H., Bencomo, N., Cukic, B., Muller, H., & Weyns, D. (2009). Software engineering for self-adaptive systems: A research roadmap. In B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, & J. Magee (Eds.), *Software engineering for self-adaptive systems* (Lecture Notes in Computer Science) (Vol. 5525).

6. Autonomic Computing, et al. (2006). An architectural blueprint for autonomic computing. *IBM White Paper*, 31(2006), 1–6.
7. Eramo, R., Bordeleau, F., Combemale, B., van den Brand, M., Wimmer, M., & Wortmann, A. (2022). Conceptualizing digital twins. *IEEE Software*, 39(2), 39–46.
8. Almasan, P., Ferriol-Galmes, M., Paillisse, J., Suarez-Varela, J., Perino, D., Lopez, D., Perales, A. A. P., Harvey, P., Ciavaglia, L., Wong, L., et al. (2022). *Digital twin network: Opportunities and challenges*. arXiv preprint arXiv:2201.01144.
9. Gheibi, O., Weyns, D., & Quin, F. (2021). Applying machine learning in self-adaptive systems: A systematic literature review. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 15(3), 1–37.
10. Cronrath, C., Aderiani, A. R., & Lennartson, B. (2019). Enhancing digital twins through reinforcement learning. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)* (pp. 293–298). IEEE.
11. Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., Pineau, J., et al. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3–4), 219–354.
12. Jouniaux, G., Barais, O., Combemale, B., & Mussbacher, G. (2021). Towards self-adaptable languages. In *Proceedings of the 2021 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software* (pp. 97–113).
13. Salama, M., Bahsoon, R., & Bencomo, N. (2017). *Managing trade-offs in self-adaptive software architectures: A systematic mapping study* (pp. 249–297).
14. de Lemos, R., Giese, H., Muller, H. A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N. M., Vogel, T., et al. (2013). Software engineering for self-adaptive systems: A second research roadmap. In *Software engineering for self-adaptive systems II* (pp. 1–32). Springer.



Democratized Hyper-automated Software Development

5

Sreedhar Reddy 

Introduction

Enterprises are fast becoming complex system of systems that increasingly need to operate in dynamic operating environments while dealing with unforeseen changes along multiple dimensions. They must constantly innovate to survive and stay ahead in this environment, be it new products, services or business models. Moreover, enterprises are increasingly part of connected ecosystems, with offerings cutting across multiple ecosystem players. Software is central not only to the operationalization and integration of these products and services but, in many cases, central to the business models themselves. With increased pervasiveness of computing, the role of IT systems will not just be limited to deriving mechanical advantage through business process automation. Instead, these systems would be the key driver of the growth story. With the availability of exploding volumes and varieties of data, there is a growing demand for sophisticated applications that are capable of deriving business insights, predicting future trends, planning and decision-making. Enterprises are fast moving towards an “everything is software” scenario wherein a variety of information processing needs (as shown in Fig. 5.1) will have to be met in a short window of opportunity. The ever-changing technology landscape adds to this, with more and more sophisticated platforms, frameworks and libraries coming up all the time, which enterprises want to exploit to stay ahead. All these factors are expected to lead to an explosion in demand for the development and evolution of software across business verticals. This exploding demand is impossible to be met relying solely on trained manpower. Even today, we experience an acute shortage of STEM skills, and this shortage is expected to increase even further over the next decade. A new approach to software

S. Reddy (✉)

Tata Consultancy Services Research, Pune, Maharashtra, India

e-mail: sreedhar.reddy@tcs.com

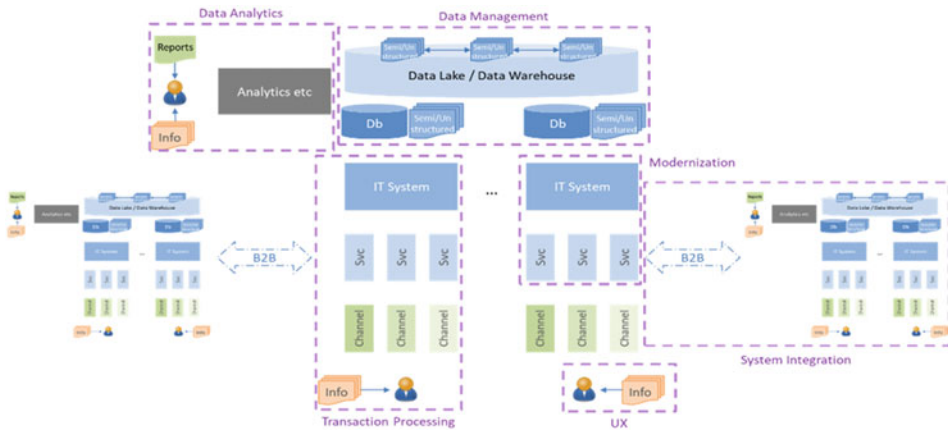


Fig. 5.1 Information processing needs of an enterprise

development is required that will enable significantly higher automation by leveraging recent advances in AI such as NLP, ML, AI-powered code completion and AI-powered testing and debugging. The approach should reduce dependencies on human expertise in capturing and using of knowledge in all phases of SDLC. Additionally, the approach should innovatively integrate relevant concepts from model-driven engineering (MDE) for code generation, formal methods for validation and verification, Digital Twin(s) for risk-free experimentation and control theory for dynamic adaptation.

Current Practice

Typical SDLC Today

Figure 5.2 depicts a typical software development life cycle (SDLC) today. The process starts with a business need. Domain experts analyse this need and come up with a set of software requirements. Key concerns here are the following: Is the requirements specification fit for purpose? Does it meet the business need? Is it complete? Consistent? This is domain knowledge-intensive where domain experts spend considerable amount of time getting the requirements specification right. Then we go through design, development, testing, etc., which is the purview of IT experts and programmers. Design addresses concerns such as how do we componentize the system; what can we reuse; what can we buy; what needs to be built afresh; what technology stack to use; which frameworks, libraries and languages; etc. The entire process is people heavy, requiring large teams with varied skills and expertise to work together. People with these skills are increasingly in short supply relative to the demand. The process also suffers from long latencies owing to

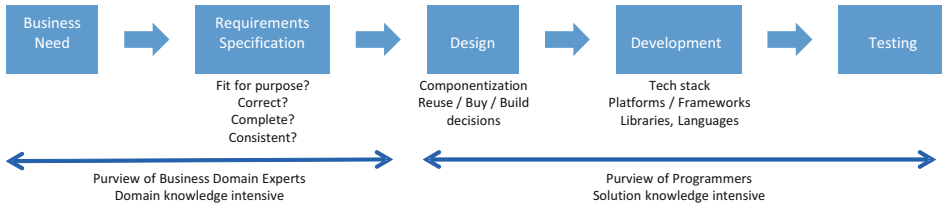


Fig. 5.2 Typical software development life cycle

complexities of communication and coordination among teams with disparate backgrounds and skills.

Over the years, we have seen several technologies that attempt to address different parts of this problem with varied degrees of success, such as model-driven development (MDD), low-code/no-code platforms and the more recent AI-powered software development.

Model-Driven Development

Model-driven development (MDD) is at least a couple of decades old, if not more. In MDD, we start with a business need, create a requirements specification and from this specification derive models and detailed business logic specifications. Up to this, it is typically a manual process. But once we have models, we can use model compilers to automate code generation. Typically, for each target platform, we have a platform-specific model compiler. These compilers need to be developed only once per platform. Since they are specified at the meta-model level, they will work for all problems that can be specified as instances of these meta-models. Figure 5.3 depicts the typical MDD pipeline.

MDD brings several benefits to software development [1]. It brings improved productivity since code generation is automated. It leads to uniform code quality since code quality does not depend on an individual’s coding skills. It provides platform independence since specifications are in terms of platform-independent models, providing an easy path to platform migration. All we need to do is use a different model compiler without having to rewrite the code. Also, since models are easier to verify compared to code, the system can be verified for various desirable properties, such as correctness, completeness, consistency, etc., at an early stage. On the minus side, modelling has a steeper learning curve since developers must understand the semantics of the underlying meta-models to do a good job of modelling. Also, the upper end of the SDLC, namely, translating a business need into software system specifications, remains manual. In some sense, while MDD addresses the need of “building systems right”, it does not address the need of “building right systems” that meet the business need.

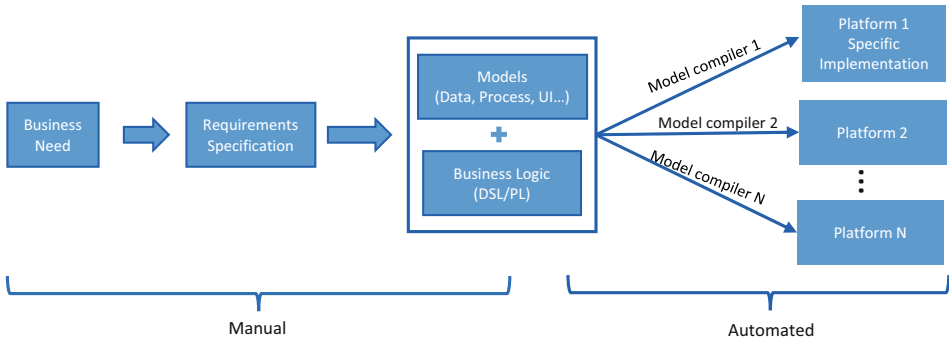


Fig. 5.3 Model-driven development

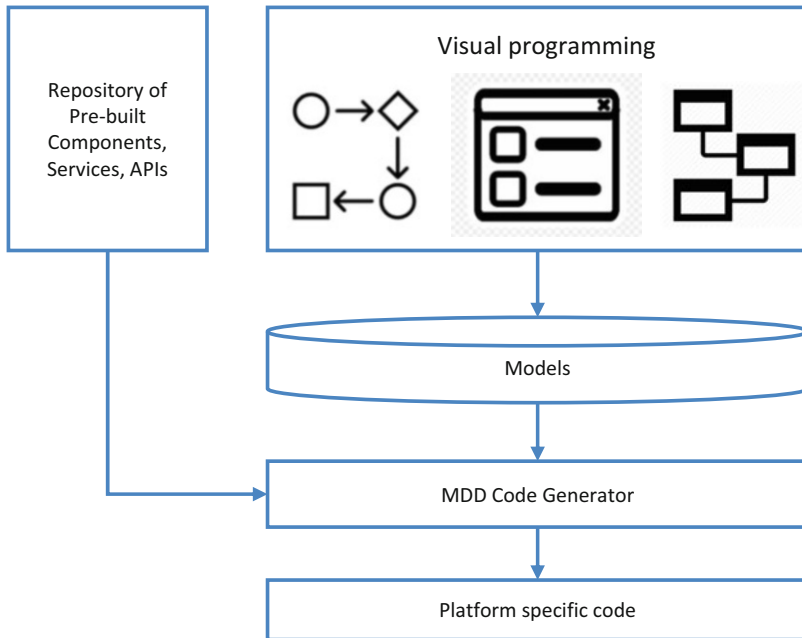


Fig. 5.4 Low-code/no-code platforms

Low-Code/No-Code Platforms

Low-code/no-code (LCNC) platforms [2] provide a drag-and-drop visual programming interface for users with little/no programming experience. LCNC is essentially a simplified version of MDD, where the visual specifications are mapped to underlying models which are then translated into platform-specific code by a bunch of model compilers (refer to Fig. 5.4). LCNC works well in situations where the applications can largely be composed

from pre-built components, with visual programming serving primary as an orchestration specification with simple rules. For applications that require a substantial amount of fresh code with modest to complex logic, visual programming quickly becomes cumbersome.

AI-Powered SDLC

AI-Powered Requirements

Recent AI-powered advances in NLP are being used to automate early stages of SDLC, for example, analysing quality attributes of requirements such as imprecision, incompleteness and escape clauses. More recent advances transform requirements spread across structured, semi-structured and unstructured information sources into formal purposive models leading to several benefits [3, 4], for example, (1) automated analysis of requirements for properties such as consistency and completeness, (2) automated computation of change and change impact, (3) automated transformation of requirements into a detailed specification and (4) end-to-end traceability via Digital Thread.

AI-Powered Testing

AI platforms can now aid in detecting, diagnosing and repairing certain classes of software errors such as incorrect usage of APIs and null dereferencing. Today, there are tools that can significantly automate user interface testing through test script generation from user interaction history and automated test script repair.

AI-Powered Coding

Recent advances in transformer-powered language models such as GPT-3, Codex and AlphaCode have opened the possibility of generating code from natural language descriptions. These are huge models (175B parameters) that have been trained on large public code repositories such as GitHub. These are being integrated into IDEs (such as GitHub Copilot) where they suggest code completions based on natural language descriptions of functions and current program context.

A language model is essentially a statistical model that predicts the next token in a sequence, i.e. it predicts the probability of the next token taking different values given the token sequence seen so far, $P(w_t | w_{t-k}, \dots, w_{t-1})$, and in a generation task, we typically take the token value that has the highest probability. In the case of code generation, given the natural language specification as the prior token sequence, it predicts the code sequence that has the highest probability of being consistent with the specification. Figure 5.5 shows a small example where the specification is incrementally extended in a series of steps to generate the code.

While language models have shown themselves to be remarkably good at learning abstract patterns and correlations among those patterns, being statistical models, fundamentally, they can only solve problems that are “similar” to the problems they have already seen. They cannot reason and synthesize a solution to a brand-new problem. What works in their favour is that they are trained on massive repositories that have solutions to most of

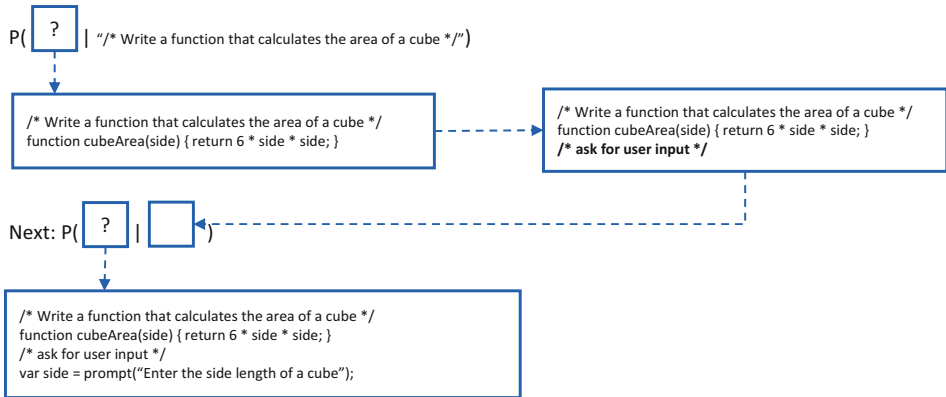


Fig. 5.5 Code generation using large language model

the common coding problems, written in several languages, using several libraries, APIs and frameworks. So, it is possible to see this technology mature to a stage where automation support of the following kind will exist for common coding problems:

As an intelligent, context-adaptive replacement for search-copy-paste-modify style of coding: Programmers routinely search the internet (Stack Exchange) for solutions. They issue a natural language query, look at the results, pick an appropriate solution, copy the code and modify it to adapt to their context. What if this entire process could be automated more intelligently and efficiently from within the IDEs? OpenAI Codex promises just that. From the text description of the problem and the coding context, Codex can (1) perform a better “search” than Google since it has the power of GPT-3 language model, (2) use much richer context for search than a simple query since it can process the program context and (3) adapt the “retrieved” code to the context since it “understands” the program context. To illustrate the last point, suppose we have a binary tree of COVID patient data in the program and we want a function to search patients in each age range. The binary tree search function (we are searching the binary tree) that Codex generates will use the COVID patient data type even if it has never seen such a data type in its training data. All this happens instantly from within the IDE.

Intelligent assistance on new/unfamiliar languages, libraries, APIs and frameworks: Suppose we have an ML programmer who knows TensorFlow (and hence the logical ML pipeline) but is new to PyTorch or we need to access stock market data but do not know which API to use. From the textual description of the task to be performed and the library to be used, Codex can automatically generate the required library and API calls.

Productivity aid to novice programmers: Tools such as Codex will be of great assistance to novice programmers on common coding problems involving various standard data types and algorithms. An assessment of Codex on LeetCode showed that it does a reasonable job. This can only get better with improving technology.

Automation of UI and DB ends of programming: Most of the mundane UI code is likely to be completely automated. For example, from a natural language description of the screen such as “A screen that has an edit box with title ‘Department’ to show department name and a list box with title ‘Employees’ to show employees of the department, and a Submit button that calls the API get-employee-list”, we can automatically generate the code to render the screen. Similarly, for a database schema and a natural language description of the query, we can generate an SQL code such as “Get the percentage of employees who have worked on AI projects”. All mundane code that does not require knowledge of the business domain and the underlying business logic is a candidate for automation.

However, this technology has some limitations too:

No guarantee of correctness: If these models remain purely statistical, there will never be any guarantee of correctness of the generated code. It will always have to be manually checked. With autoregressive language models such as Codex, the farther the prediction is from the input prompt, the greater the drift from the intent. So, longer code fragments require several iterations of generate-correct-regenerate to get them right.

Determining the level of specification detail: We do not have a clear answer to this at this stage. It obviously defeats the purpose if the natural language description needs to be as detailed as the pseudocode. One can even argue that it is much harder to specify a program precisely in natural language than to write the code. There needs to be an empirical study to assess this aspect.

Incapable of generating code for business logic: AI models we see today are trained on public repositories. Since enterprise code repositories are private and will not be made public anytime soon, it is unlikely that these models will generate code for business logic at present or in the near future.

Computational cost: Today, OpenAI’s GPT-3 and Codex are both available only as APIs, to be used in as-is form, with the model hosted on OpenAI’s hardware. However, to be more useful to enterprise applications, Codex needs to be trained on domain-specific knowledge and code. The cost of training the model is likely to be high, estimated to cost OpenAI about 3.114E23 FLOPS of compute power to train its GPT-3 model.

Proposed Line of Attack

While MDE, LCNC and AI-driven code generation address different aspects of the SDLC, none of them provides a comprehensive end-to-end solution for deriving software system specifications from business needs and then deriving software system implementations from these specifications. As discussed earlier, the former is a domain knowledge-intensive activity and the latter solution knowledge-intensive, and none of these approaches provides

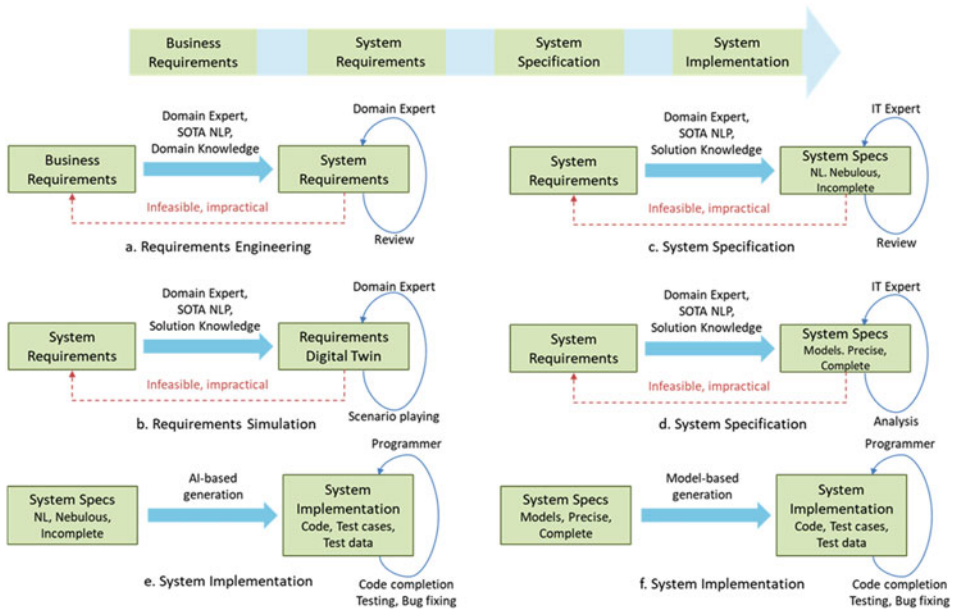


Fig. 5.6 Knowledge-guided AI-aided model-driven software development

a systematic means to integrate knowledge and knowledge-based reasoning into the SDLC process.

To address this, we outline a software development approach that brings together knowledge, AI and MDD to significantly enhance the automation in SDLC, bringing the process closer to domain experts. Figure 5.6 outlines the high-level approach. The approach leverages domain knowledge and state-of-the-art NLP to help business domain experts refine high-level business requirements into detailed software system requirements that are complete with respect to meeting the stated business needs (refer to Fig. 5.6a, Requirements Engineering). These can then be reviewed for feasibility and practicality, perhaps using a requirements digital twin (refer to Fig. 5.6b, Requirements Simulation). We leverage solution knowledge and state-of-the-art NLP to help IT experts refine the system requirements into a software system specification, catering to functional and non-functional requirements (refer to Fig. 5.6c, System Specification). These specifications may either be precise formal models using model-based generation (refer to Fig. 5.6e, System Implementation) or natural language text (with associated ambiguity and imprecision) using AI-based generation (refer to Fig. 5.6f, System Implementation) to derive implementation artefacts such as code, test cases and test data. We leverage AI-based testing, bug detection and bug fixing, wherever possible. The various implementation artefacts sit in a knowledge repository, thus facilitating further training of AI-based SDLC tools.

We delve into details of these steps below.

Knowledge-Guided, AI-Aided Refinement of Business Requirements into Software Requirements

The first step towards implementing a successful software system is to arrive at right software requirements that will meet the business need. This requires deep business domain knowledge. It also requires some amount of systems knowledge, i.e. knowledge of what systems solutions work for what problems in what context. This knowledge is typically derived from solutions developed in the past. To provide automation support, we need machinery that captures both kinds of knowledge in a machine-processable form. To aid business experts, we need an automated process that can reason with this knowledge to systematically transform a high-level business requirement, given in natural language, into a set of software requirements. We also need a validation mechanism that can check the generated requirements indeed meet the business needs. Digital Twin(s) can be of value here. They can potentially simulate the requirements to see how the system behaves in its operating environment.

Domain Ontology

Domain ontology describes the core concepts and relations that underlie a business domain along with the rules and constraints that govern them (Fig. 5.7). The concepts encompass business domain entities, events, actions and operations. The ontology serves as the semantic base and provides the vocabulary in terms of which domain knowledge is expressed and interpreted. It also serves as the common reference point for all stakeholders involved in system development to align their respective artefacts. Ontological alignment of all artefacts involved in software development, namely, business requirements, software

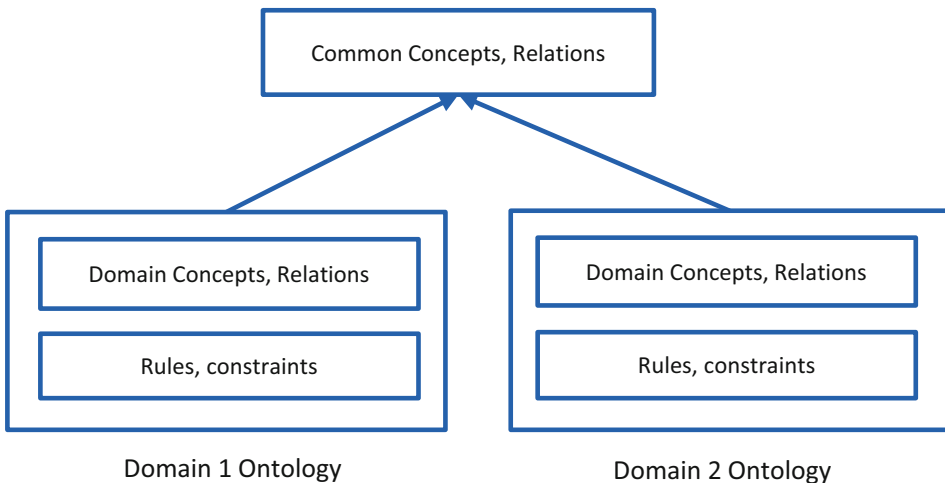


Fig. 5.7 Domain ontologies

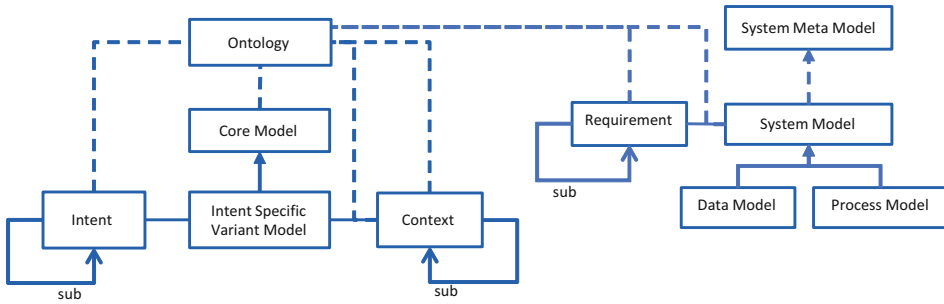


Fig. 5.8 Systems knowledge

specifications, models and code, paves the way for transfer learning across related problems in a given domain and across related domains.

Systems Knowledge

Systems knowledge consists of knowledge of different systems that serve different intents in different contexts. A system is described in terms of its intent and context, their decomposition into detailed requirements and a set of associated structural and behavioural models that meet the stated intent in the stated context. This can be organized into a core part and a set of variants corresponding to variations in intent and context. A comprehensive knowledge repository also contains code artefacts corresponding to the structural and behavioural models. All elements of the repository (i.e. intent, context, requirements, model elements, their natural language descriptions and code) are mapped to the domain ontology to semantically align them with domain concepts. Figure 5.8 schematically describes the various models and their relationships. The key thing to note here is that all systems are described in terms of a common meta-model and all model elements and their descriptions and coding artefacts are aligned with a common domain ontology.

AI and NLP

Such a knowledge repository facilitates training of AI models. From it, we can generate training data tuples of the form $\langle ontology, system-meta-model, intent-description, context-description, system-model, system-code \rangle$. AI learns to use the common meta-model and ontology as the basis to generalize from problem-solution instances captured in the knowledge repository.

In a typical SDLC, we start with a high-level business requirements specification given in natural language that describes the business intent and context. Then we align this specification with the ontology (see example in Fig. 5.9) using NLP. This alignment enables the AI to relate the specification with prior knowledge (which is already ontology aligned) and enable it to carry out the required transformations, namely, from business requirements to software requirements and further on from software requirements to software implementations. To be able to do this, we need AI models that can combine

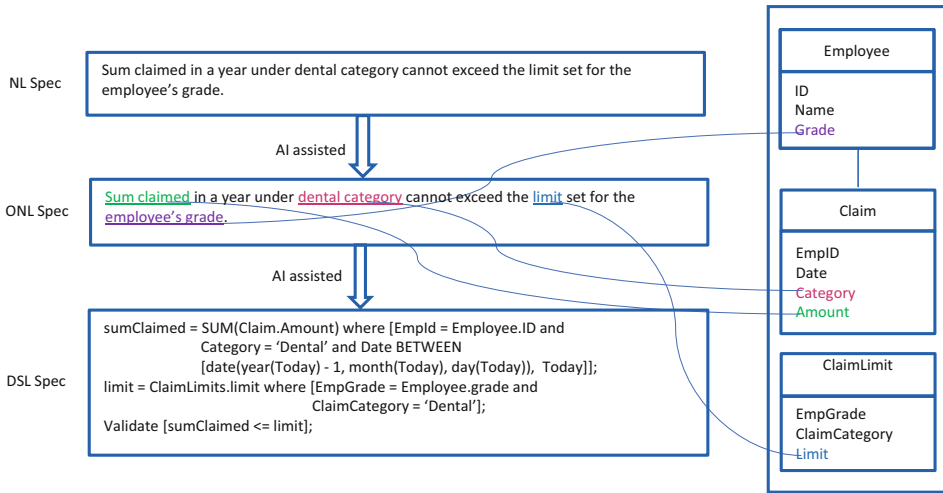


Fig. 5.9 AI-assisted transformation via ontology-aligned natural language (ONL) specifications

language processing with knowledge-based reasoning. While language models themselves are quite mature, integrating structured knowledge with them is still an on-going research problem. Recent work on integrating knowledge graphs into language models [5–7] is a step in that direction. This needs to be further extended with neuro-symbolic approaches to integrate axiomatic knowledge as well.

Digital Twin(s)

The idea of Digital Twin(s) has been gaining ground in both industrial and enterprise spaces. Digital Twin is a purposive virtual representation of reality that responds to automated analysis typically through what-if scenarios. While Digital Twin(s) have been used as design aids in physical and cyber-physical spaces, their use in enterprises has been limited to a decision-making aid, that is, to identify “in silico” which among the available options is the best response in a given situation. However, in enterprise software development, the concept of digital twin can be very useful in arriving at the right system specification, starting from high-level requirements [8]. A simulation-capable digital twin enables a business domain expert to explore the solution space by playing out various scenarios of interest under a multitude of design options [9]. A software system exists in a business context interacting with several internal and external actors. It is useful to simulate this whole context to see which solution option has what effect, thereby helping arrive at the right system specifications in an informed manner. For this to be feasible, it should be possible to generate a light-weight digital twin (stripped of all non-functional concerns that a typical software system would have) from requirements specifications. We believe a lighter version of the knowledge-guided, AI-aided, model-driven system implementation approach discussed below will do the job.

Knowledge-Guided, AI-Aided Refinement of Software Requirements into Software Specifications

From software requirements, we derive detailed software system specifications. This involves deriving detailed functional specifications such as data models, process models, business logic, etc. This also involves deriving the right software architecture (components/services, etc.) and technical architecture (GUI/middleware/DB, cloud, etc.) to meet the non-functional requirements. This part of the SDLC is solution knowledge-intensive. By solution knowledge, we mean knowledge that helps in deciding which design and architectural patterns to use for what problem in what context, and what platform-specific choices to make in what situation, based on non-functional requirements. Here, system specifications might take two different forms.

In one case, we might derive specifications in the form of models and domain-specific languages (DSLs) and use model-based code generators to translate them into platform-specific implementations. However, these specifications might be harder for domain experts to verify and correct. Also, this approach presupposes that we have the necessary modelling languages and DSLs which are complete for the purpose. This need not always be the case. But where they do, this is a more reliable and well-proven path to system implementation.

In other cases, we may derive specifications in the form of controlled natural language which is easier for domain experts to verify and correct. However, being expressed in natural language, these specifications are likely to contain some amount of inherent ambiguity and incompleteness from a computer implementation perspective. This is where AI-based code generation technologies excel, as they are better at dealing with ambiguity and incompleteness in inputs. As discussed earlier, language models such as Codex demonstrate great potential, targeting a wide range of languages, platforms and frameworks. However, they need to be trained and fine-tuned on domain-specific knowledge and content to be able to generate domain-specific business logic from natural language specifications. This is where the knowledge repository architecture discussed in previous sections helps, where all systems are described in terms of a common meta-model and all content is aligned with a common domain ontology. AI can learn to use the common meta-model and ontology as the basis to generalize from problem-solution instances captured in the knowledge repository. For instance, we can generate training tuples of the form `<ontology, meta-model, model, NL-functional-spec, NL-non-functional-spec, code>` to train the AI model and then use the model on a new specification (i.e. `<ontology, meta-model, model-new, NL-functional-spec-new, NL-non-functional-spec>` as input) to generate platform-specific business logic compliant with the spec. As mentioned, to be able to do this, we need AI models that can combine language processing with knowledge-based reasoning. Recent work on integrating structured and symbolic knowledge into language models looks quite promising in this regard [5–7].

Architecture for Software Adaptation

Software implementation is not a one-time activity. Over time, requirements change, operating environment changes, system goals themselves change, and so on. A software system needs to evolve suitably to cater to these changes. To address this, we propose a novel architecture for continuous dynamic adaptation of software systems as discussed in Chap. 4.

Technology Infrastructure to Support the Line of Attack

To support the proposed approach, we need the following key technology enablers:

1. Domain ontologies and automation aids to construct purposive domain ontologies by mining structured/semi-structured/unstructured text resources
2. Knowledge-enhanced NLP capable of processing nebulous incomplete requirements
3. Requirements digital twin, automation aids for its construction from detailed requirements in text form and the associated simulation machinery
4. Modelling infrastructure to capture system specifications in model form and machinery to author these models from NL text
5. Model validation and verification machinery
6. Model-based code generation machinery
7. AI-based code generation machinery

While some of these technologies are already mature, others are still at a research stage. The following are the more mature ones:

Authoring models from NL text: We presume that the necessary information concerning functional and non-functional characteristics of the desired application is available, albeit in a fragmented form, distributed over several text documents. This information may cover the known knowns (i.e. static and fully known information) and the known unknowns (i.e. fully known possible variations). Moreover, the documents containing this information may conform to a certain superstructure (organized in terms of sections and subsections whose names give a definitive hint of their text content). Several algorithms exist that can process this NL text content to identify domain concepts and the relationships between these concepts, that is, the domain model which SMEs can further refine, if required.

For instance, we have an automated regulatory compliance (ARC) solution that enables SMEs to author a purposive domain model based on the regulation document released by the regulatory authority. This model serves as a lens to sift the relevant text from a regulation document. This text containing obligations is presented in a Controlled Natural Language (CNL) which is more intuitive for SMEs to refine. The refined

obligations in CNL are translated to structured English (SE), a linearized representation of Semantics of Business Vocabulary and Business Rules (SBVR). Thus, text in SE can easily populate the SBVR model. This $NL \rightarrow CNL \rightarrow SE \rightarrow SBVR$ is a human-in-the-loop refinement process supported by automation aids [10]. While we have used these underlying NLP and ML techniques to author a variety of models from NL text, they need to be enhanced to help SMEs author the solution architecture model.

Modelling infrastructure: As we would be using different models to specify different aspects of the system, we need a meta-modelling infrastructure so that purpose-specific meta-models can be defined. Several such meta-modelling solutions exist. For example, TCS MasterCraft™ provides a meta-modelling infrastructure complete with multi-user, multi-site repository having in-built versioning and configuration management capability [1].

Validating the authored models: The authored models must be consistent with the domain ontology and must be correct with respect to the intent. The former is a solved problem with several automation aids across a wide spectrum of sophistication. We can use simulation-based techniques to address the latter. For instance, Enterprise Digital Twin (EDT) models support what-if and if-what scenario playing to ascertain correctness. We can also use Answer Set Programming (ASP)-based techniques to help SMEs ensure the authored model is correct, as done in the regulatory domain [11].

Model-based code generation machinery: System specification in model form can be automatically transformed into various implementation artefacts such as code, test cases, test data and deployment descriptors. System specifications are kept agnostic of technology platform, thereby enabling them to be retargeted to multiple technology platforms by changing the code generators suitably. In TCS MasterCraft™, we have comprehensive model-based code generation infrastructure comprising model-to-model and model-to-text transformers. These transformers are specification-driven and can be easily customized to cater to different technology platforms and purposive meta-models [12].

Knowledge repository for actionable contextual intelligence: One of the key strengths of any business enterprise is its deep domain and contextual knowledge, gained over decades of experience. It should be possible to capture this knowledge formally in a machine-processable form, so that it can be systematically exploited in solution development processes. Knowledge exists at multiple levels of abstraction, in multiple forms (facts, rules, models) to serve multiple intents (purposes). A knowledge model should capture all these aspects of knowledge and enable context-based reasoning to identify and reason according to a given problem context. At TCS, we have developed a similar knowledge modelling framework in our work in materials engineering [13]. This framework has proven to be versatile in supporting a diverse range of applications in the engineering space. We believe it can be easily extended to support software development as well.

Digital Twin: Digital Twin is a virtual purposive, high-fidelity representation of reality that is responsive to what-if and if-what scenario playing. A purposive requirements digital

twin can help SMEs arrive at the right requirements efficiently through scenario playing. Also, a digital twin of the software system can evolve and adapt the software system efficiently in response to changes in the desired objectives as well as operating environment. At TCS, we have developed actor-based infrastructure (EDT) that helps SME construct a purposive digital twin and use it as a risk-free experimentation aid to arrive at right decisions. We use the purposive digital twin as an experience generator to train a Reinforcement Learning agent, thus considerably automating the decision-making process [14].

There are also several new investigations that are required, such as:

Domain Specific Languages: The key objective is to empower SMEs to play a significantly greater role in SDLC, and eliminating accidental complexity is a significant step towards this objective. General-purpose programming languages pose a serious problem for SMEs. They need more intuitive and closer-to-business means for specifying system requirements. There is a need to strike the right balance between expressiveness, intuitiveness and pragmatism. As these means would vary from problem to problem and domain to domain, there is a need for a platform that can help quickly define suitable Domain Specific Language (DSL). While the well-researched space of language engineering can provide primordial concepts, some effort is needed to come up with purposive DSLs. Once the DSL is ready, the model authoring infrastructure can be easily repurposed to author system specifications in DSL.

Leveraging AI for SDLC: The intent of our approach is to leverage AI in all stages of SDLC. Manual developer-centric SDLC can benefit from AI in terms of IntelliSense, intelligent code completion and proactive search-cut-n-paste. Specification-based code generative SDLC can benefit in terms of increased automation in early SDLC stages, which are document-centric and manual. AI techniques such as Reinforcement Learning (RL) can enable smart Integrated Development Environments (IDEs) that can provide highly personalized user experience based on user profile, intent and past interactions. RL seems to be a key enabler to impart continuous improvement (and adaptiveness) to IDEs over time. Recent AI advances such as language models, transformers and program synthesis give hope of generating implementations from descriptions in NL text and input-output data. At present, these AI-based automation aids lack in domain specificity, having been trained on public repositories such as GitHub. We believe their effectiveness can be significantly improved if they are trained on content that is domain-specific and are capable of using domain knowledge in their operation.

References

1. Kulkarni, V., Reddy, S., & Rajbhoj, A. (2010). Scaling up model driven engineering—experience and lessons learnt. In *International conference on model driven engineering languages and systems* (pp. 331–345). Springer.

2. Bock, A. C., & Frank, U. (2021). Low-code platform. *Business & Information Systems Engineering*, 63(6), 733–740.
3. Nistala, P. V., Rajbhoj, A., Kulkarni, V., Soni, S., Nori, K. V., & Reddy, R. (2022). Towards digitalization of requirements: Generating context-sensitive user stories from diverse specifications. *Automated Software Engineering*, 29(1), 1–35.
4. Rajbhoj, A., Nistala, P., Kulkarni, V., Soni, S., & Pathan, A. (2022). DizSpec: Digitalization of requirements specification documents to automate traceability and impact analysis. In *2022 IEEE 30th International Requirements Engineering Conference (RE)* (pp. 243–254). IEEE.
5. Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., & Liu, Q. (2019). *ERNIE: Enhanced language representation with informative entities*. arXiv preprint arXiv:1905.07129.
6. Peters, M. E., Neumann, M., Logan, R. L. IV, Schwartz, R., Joshi, V., Singh, S., & Smith, N. A. (2019). *Knowledge enhanced contextual word representations*. arXiv preprint arXiv:1909.04164.
7. Yu, D., Zhu, C., Yang, Y., & Zeng, M. (2022). Jaket: Joint pre-training of knowledge graph and language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10), 11630–11638.
8. Kulkarni, V., & Reddy, S. (2017). From building systems right to building right systems. In *Federation of International Conferences on Software Technologies: Applications and Foundations* (pp. 184–192). Springer.
9. Kulkarni, V., Barat, S., & Clark, T. (2019). Towards adaptive enterprises using digital twins. In *2019 winter simulation conference (WSC)* (pp. 60–74). IEEE.
10. Sunkle, S., Kholkar, D., & Kulkarni, V. (2016). Informed active learning to aid domain experts in modeling compliance. In *2016 IEEE 20th International Enterprise Distributed Object Computing Conference (EDOC)* (pp. 1–10). IEEE.
11. Kholkar, D., Mulpuru, D., & Kulkarni, V. (2018). Balancing model usability and verifiability with SBVR and answer set programming. In *MoDELS (Workshops)* (pp. 570–573).
12. Kulkarni, V., & Reddy, S. (2008). An abstraction for reusable MDD components: Model-based generation of model-based code generators. In *Proceedings of the 7th international conference on Generative programming and component engineering* (pp. 181–184).
13. Yeddula, R. R., Vale, S., Reddy, S., Malhotra, C. P., Gautham, B. P., & Zagade, P. (2016). A knowledge modeling framework for computational materials engineering. In *SEKE* (pp. 197–202).
14. Barat, S., Khadilkar, H., Meisheri, H., Kulkarni, V., Baniwal, V., Kumar, P., & Gajrani, M. (2019). Actor based simulation for closed loop control of supply chain using reinforcement learning. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 1802–1804).



Coordinated Continuous Digital Transformation

6

Henderik Proper  and Bas van Gils 

Introduction

The overall focus of this book is on the transformation of enterprises towards *AI-Enabled Enterprises*, involving a strong role for both AI and digital twin technologies. At the same time, it is important to realize that for enterprises, the transformation towards AI-Enabled Enterprises is “just” a logical, albeit important, next phase in the continuous flow of digital transformations which enterprises are (and need to be) engaged in. In this chapter, we therefore specifically zoom in on both the challenges facing enterprises regarding digital transformations in general and the transition to AI-Enabled Enterprises in particular. In doing so, we will review, and integrate, both insights from practice and insights from research results.

Since digital transformations have (by definition) a profound impact on the structure of an enterprise, it is important to ensure that such (enterprise) transformations are well-coordinated [1, 2]. Enterprise (architecture) models are traditionally regarded as an effective way to enable such informed coordination and decision-making [1, 3]. In line with this, we take a model-enabled perspective on the needed coordination, in particular in the context of what we call *enterprise design dialogues* [4].

In the second section, we start by defining more precisely what we mean by digital transformation. The third section then reflects on the fact that digital transformations should be seen as a continuous process. This is then complemented in the fourth section with the

H. Proper (✉)
TU Wien, Vienna, Austria
e-mail: e.proper@acm.org

B. van Gils
Antwerp Management School, Antwerp, Belgium
e-mail: bas.vangils@strategy-alliance.com

observation that it is essential for these continuous digital transformations to happen in a coordinated way, involving coordination among many different actors. The fifth section reviews the concept of *enterprise design dialogues* that we see as being at the heart of the needed coordination of transformations. In the sixth section, we then attend to the crucial role of models (including the virtual model included in a Digital Twin(s)) to support enterprise design dialogues. Finally, before concluding, the seventh section reviews challenges and opportunities towards future research.

Digital Transformation

Our society has transitioned well and truly from the industrial age to the digital age. As a result, “digital” has become an integral part of our lives. Tasks in our common lives that used to be completely “analogue” are now increasingly “digital”: ordering pizza, ordering a taxi, booking a vacation trip, dating, etc. Similarly, in business, we see an increased transition from “analogue” via “digitized” (i.e. replacing paper with PDF) to “digital” (redesign of value proposition and operating model) business models [5]. The *on-going* development and maturation of “digital technologies”, such as mobile computing, pervasive computing, cloud computing, big data, AI, robotics, social media, low-code, Digital Twin(s), etc., drive enterprises to transform. Even more, non-IT infrastructures, such as electricity networks, water networks, transportation networks and even cities and buildings, increasingly become IT-intensive infrastructures. As a result, it is now humans and IT, who are jointly the driving agents in an enterprise, where IT is increasingly also fulfilling the role of the “operating system” of the enterprise. The increased use of different forms of AI in conjunction with digital twin technologies now ushers in a further transition for enterprises, from being “digital” to being *AI-enabled*.

When we speak about “digital transformation”, we do so primarily in the context of “enterprises”. An “enterprise” is a “unit of economic organization or activity” [6] such as a company, a government agency, a factory, etc. It is also, at a more fundamental level, a purposeful system (i.e. its enterprise) in the sense of conducting (possibly as part of a network of enterprises) a particular business in the sense of a “particular field of endeavour” [6]. In some areas, this is stated as systems having a *function* in their environment (e.g. [7]). With this in mind, we [8] define *digital transformation* as follows:

The deliberate effort to transform the architecture of the enterprise, with a significant impact on its digital capabilities.

The phrase “digital capabilities” refers to those business capabilities [9] of an enterprise that are digitally driven or at least highly digitally reliant. In terms of [10, 11], digital transformations may not only change the *operational capability* (needed to execute the business and operating model) of enterprises but specifically also their *dynamic capability*

(needed to continuously improve and innovate the business and operating model in relation to new opportunities and challenges).

To expand on the above definition of digital transformation, it should be noted that the term *architecture* has different meanings. The way it is used in the above definition should be taken in line with the general definition of architecture as reported in our earlier work (e.g. [12]):

Those properties of an artifact that are necessary and sufficient to meet its essential requirements; or in more colloquial terms it is about ‘what (should) keep(s) stakeholders awake at night’.

In line with [13], the latter is usually operationalized for *system architectures* by the assertion that, for systems, architecture concerns (1) the fundamental properties of a system (in terms of components and their relations) and (2) the principles guiding design and evolution. In light of the definition provided by [12], this implies that digital transformation initiatives are aimed at changing the *essence* of the organization. The corollary is that (relatively) minor changes that leave the essence of the organization intact do not count as digital transformation initiatives.

A further observation with regard to the definition is digital transformation is the focus on *digital capabilities*. The implication is that transformation initiatives that do not have a significant impact on the digital capabilities of the organization do not count as digital transformation initiatives. As before, we are not claiming that these do not occur nor that they are not important. We simply do not consider these to be digital transformation initiatives.

Continuous Digital Transformation

The *on-going* development and maturation of “digital technologies” certainly drives enterprises to change. However, this is certainly not the only source for change in enterprises. Market dynamics, new regulations, opportunities offered by other (non-digital) new technologies, etc. force modern-day enterprises to change almost continuously. This is sometimes referred to as the “VUCA” world (volatile, uncertain, complex and ambiguous; see, e.g. [14, 15]).

At the same time, we argue that enterprises have always had a need to change. Before the Industrial Revolution, such changes might have (in general) occurred at a slow pace. Social and political developments (including wars and revolutions) may have caused a temporary increase in the pace of change. The technological advancements driving, and causing, the Industrial Revolution added more speed to change. Enterprises could innovate at a higher pace due to the technological developments, while society at large also became more demanding regarding products and services. During this period, companies with the

“best” innovations “won” in the market (first-mover advantage), often leading to extravagant market positions.

We are now experiencing the “Digital Revolution” [5, 16–18], which is increasing the speed of change even more. Our observation is that this is a *reinforcing loop*. Market developments drive the need to innovate, which is faster in a digital space. Demand creates supply: organizations innovate at a higher and higher pace. This in turn drives market developments, which sets in motion the next “cycle”.

What also strengthens the reinforcing loop is the fact that, as mentioned above, digital transformations can not only be used to transform the operational capabilities of an enterprise but their dynamic capabilities as well. Initial examples of the digital transformation of dynamic capabilities include the use of workflow engines, business rule engines and low-code solutions. The introduction of AI to support different tasks in digital transformations (see, e.g. [19–21]) is a prelude towards things to come for the dynamic capabilities in AI-Enabled Enterprises.

As a result, we would argue that one needs to increasingly consider digital transformation to be a continuous process and certainly not as a “one-off” project.

Coordinated Continuous Digital Transformation

So far, we have discussed what digital transformation is and that it should be considered as a continuous process. We now shift perspective to emphasize the fact that digital transformation requires strong *coordination* to be successful.

We start with two related observations based on the previous discussion. As the definition of digital transformations stipulates, digital transformations have a significant impact on the digital capabilities of an enterprise. We can also observe how, over the past decades, the role of IT in enterprises has increased from the mere automation of information processing, via the automation of actual business processes, to now being a core element of their business models. As the role of IT in enterprises increased, so did the need to ensure a coherent design between IT and all other aspects of an enterprise [22], from the operational alignment between human and IT-based activities to the longer-term strategic alignment [3, 23]. A second pertinent observation is that experience shows that digital transformation requires a *deliberate effort* to achieve an outcome [24, 25].

Our position is that both of these observations, i.e. (1) the need for *coherent design* and (2) the fact that digital transformations require a *deliberate effort*, point towards the need for a *coordinated* [1] approach to digital transformation to ensure that the *profound impact* of these transformations pushes the enterprise in the right direction in a coherent and deliberate manner.

In the remainder of this section, we argue that there is another fundamental reason for requiring a coordinated approach: social and technical *complexity*. Different frameworks exist to classify, and reason about, *complexity* of problems in general. For our discussion below, we primarily rely on the Cynefin framework (see, e.g. [26–29]). In this *sense-*

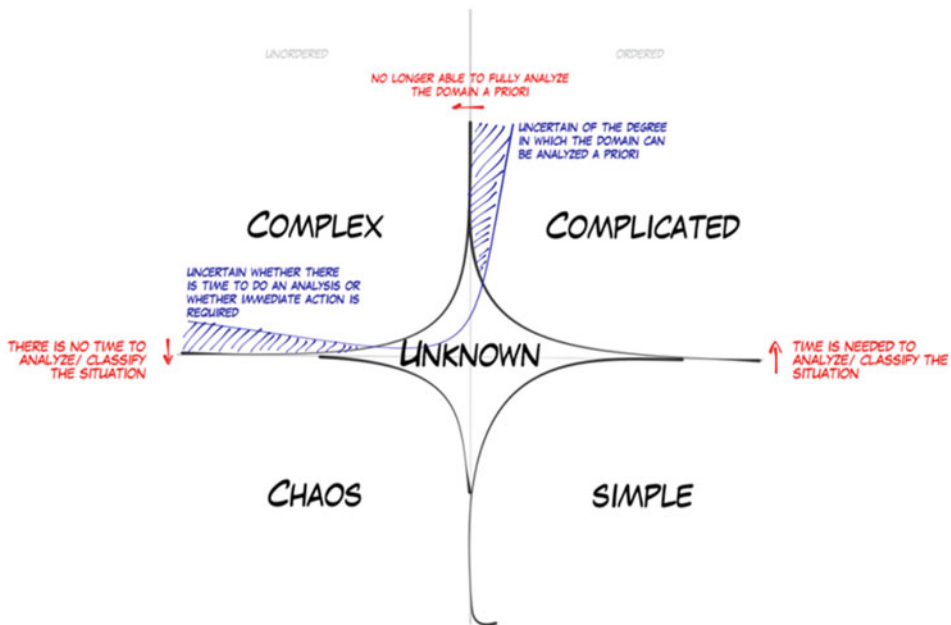


Fig. 6.1 The Cynefin framework [28], enhanced with our own interpretation

making framework, “problems” are classified into different *domains*. The framework, enriched with our interpretation, is visualized in Fig. 6.1. To understand why it is important to understand in which domain a problem at hand fits, we use a quote from the GUM:¹

Throughout the day, we are forced to deal with numerous new impressions and experiences. In order to get to grips with the chaos that characterises the world around us, we are constantly on the lookout for connections and patterns. Based on those, we are able to classify reality and create order. Scientists draw up similar classifications. However, these need to be underpinned by clearly defined criteria which will determine in a straightforward way whether or not something belongs to that classification, and, if so, where. Now, you might wonder: does that order truly exist? Alternatively, do scientists impose said order on reality?

The domain of *simple* problems is characterized by the fact that challenges are recognized as belonging to a certain class, so the solution to these challenges is immediately obvious, while there is none to moderate time pressures to realize the solution. A good example would be the update of the operating system as used on the desktop or the migration of e-mail services to a new e-mail platform.

¹Gents Universitair Museum <https://www.gum.gent/en/collection-album/chaos-1>; seen on 27-Dec-2022

In the domain of *chaos*, the complete opposite is true; whenever there is chaos and a (life-threatening) crisis emerges, an immediate response is usually lacking. In situations of this type, it is suggested [28] that decisive leadership is required in order to stabilize the situation. In practical terms, this would entail returning to one of the other three problem domains. This is why, e.g. officers in the military speak of *the terrible burden of command*: when the proverbial shit hits the fan, they still have an army to lead with potentially lethal consequences [30].

This leaves the *complicated* and *complex* domains. The former refers to situations for which an a priori, provably correct solution can be developed. This does not mean that these are simple, or trivial, problems. They are, however, *complicated* [30]. Usually, these situations involve challenging *engineering problems*, such as the design of bridges or other intricate water works. The generally used approach in these kinds of situations is to *analyse* the situation, *design* a solution and then *implement* it. Note that in these situations, a *reductionist* approach is common: the “essential” properties of the problem domain are analysed (whatever these may be), and the irrelevant aspects are (and are assumed safe to be) ignored.

In contrast, for problems in the domain of *complex* problems, no a priori correct solution can be found. The situation is characterized by the fact that the interplay between variables is so complex that cause and effect can only be analysed (fully) a posteriori. Problems in this domain are also called *wicked problems* [31]. This is the realm of emergent change, where a hypothesis of the situation is the input for deciding about potential action which has to be evaluated a posteriori to see if it delivered the expected results. In software engineering, this usually entails to the use of an *agile* approach. Note that the approach here does not favour reductionism; the whole point in this domain is that a full analysis is not possible. The emphasis is, indeed, on *probing* the organization and *evaluating* results – something that is often referred to as *situational awareness* [32, 33].

Note further that:

- There is a fifth “unknown” domain of problems in the centre of the framework. This is used to signal situations here we do not yet know in which of the four main domains we are.
- The left-right “split”: both the simple and complicated domains are said to be *ordered*. This is intended to signify that a correct solution can be derived a priori. The complex and chaos domains are said to be *unordered* and do not have this property.
- Going from the simple domain to the complicated domain signifies a clear distinction between situations where a full understanding is immediately apparent (simple domain) versus situations where time for analysis is needed.
- Going from the complicated to the complex domain signifies a clear distinction where time is available for analysis (complex) versus situations where it is not (complicated).
- The “squiggle” at the bottom, between the chaos and simple domains, is intended to signify a rift/barrier: it is not possible to go from the chaotic domain to the simple domain; one will have to go “up” to the complicated or complex domain.

- The blue areas are “in-between” areas. These signify the areas of doubt and uncertainty where we are unsure in which of the two adjacent areas we are.

Our claim is that (continuous) digital transformations are primarily in the *complex space*. We motivate this claim by referring to the earlier observation that, in our view, the *profound* impact suggests that many parts of the enterprise will be impacted. These “parts” come in many shapes and forms that are intricately intertwined: people in their roles, processes (structured and creative), data, information systems, infrastructure, team meetings and perhaps even culture are all considered [34]. Furthermore, AI-based actors/components will add even more complexity to the mix, especially, when taking the complex interplay between multiple human and AI-based actors into account.

In [35], the complexity that originates from people and their different interests and backgrounds is referred to as *social complexity*. Inspired by this, in [34], the following pseudo-formula for *social complexity* is suggested:

social complexity = #stakeholder roles × diversity of stakes × diversity of cultures

This could be complemented further with *technical complexity* due to the interplay between the different components and relations involved in a digital transformation. As mentioned before, the increased use of AI and the potential interplay (and associated uncertainty regarding causes and effect).

The social complexity and technical complexity involved in digital transformation put more stress on the need for coordination. To further illustrate this, we introduce two new notions: (1) a single-effort digital transformation initiative refers to a situation where a single initiative attempts to achieve a digital transformation outcome, and (2) a multi-effort digital transformation refers to a situation where a group of *parallel* initiatives does the same. We deliberately use the “vague” term *initiative* to avoid a (waterfall) project versus agile discussion while also embracing the earlier observation that digital transformations (be it single-effort or multi-effort) should be thought of as continuous processes.

Let us now, briefly, examine each of these kinds of digital transformations in turn. In a single-effort digital transformation initiative, one attempts to change the core/architecture of the enterprise in a single initiative. Our earlier claim is that digital transformation initiatives are in the complex space, which suggests that a full a priori understanding of the domain is, by its very nature, not possible. We argue that multiple stakeholders are involved in such an initiative, also requiring the balancing of “local” interests (e.g. at business unit level) and “global” interests (e.g. at company-wide level) [1] (including, for instance, the need to comply to regulations). This emphasizes the need for coordination: both within the group of stakeholders that shape and execute the transformation initiative and with the stakeholders that are impacted by it. Indeed, in agile methods (e.g. SCRUM), there is much focus on communication and rituals (daily stand-up, retrospective, etc.) to arrange for this kind of coordination.

In a multi-effort digital transformation initiative, these coordination challenges become even more pressing. Here, we not only see the need for coordination (among stakeholders)

within a transformation initiative but also *across* different (parallel) initiatives. A more flexible approach seems to be called for, i.e. also less “Big Design Up Front” [36, 37]. At the same time, concerns, such as regulatory compliance, risk management, security, etc., do require an integrated view (and design) of all relevant aspects of an enterprise [3, 38]. In practice, we tend to see variations of *scaled agile* emerge.² Scaled agile methods are characterized by coordinating mechanisms “on top of” agile initiatives, ensuring that their goals and efforts align sufficiently.

To summarize our point, digital transformations are highly complex (and will be even more complex when AI is involved). They are also continuous and require strong coordination either within or across initiatives. In the next section, we will investigate the role of models in light of this point.

Enterprise Design Dialogues

We [4] take the view that, in general, the design of the structure (processes, hierarchies, (IT) infrastructures) of an enterprise is (re)shaped by a continuous flow of (top-down and bottom-up) *enterprise design dialogues* between the different involved human actors. (Coordinated and continuous) digital transformations are no exception to this.

This may sound abstract, but in practice, such design dialogues occur all across enterprises. Or in the words of [39]: “Design literally shapes organizational reality”. Each time co-workers discuss “how to” divide work or conduct a (new) task, they essentially engage in an enterprise design dialogue. When process engineers discuss with senior business management how to shape a business process, they are having a design dialogue. When database engineers discuss with domain experts what information needs to be captured in the database, they are having a design dialogue. When the enterprise architects that are involved in a digital transformation coordinate with different stakeholders regarding the future direction of the enterprise, they are having a design dialogue. These examples show how design dialogues occur across an enterprise, meanwhile (re)shaping the design of the enterprise.

As a more concrete example of (1) what a design dialogue looks like and (2) what the value of such a dialogue can be, consider the situation at a utilities company that we³ consulted to. This company had had a “best of breed” software strategy in the past. They ended up with a set of systems from different vendors that were only loosely connected. The situation served them well for years on end, but the need for a more integral approach to data access (both for operational and for business intelligence purposes) had arisen. It had been decided that “we need an integral platform” to make that happen, without having

²Several frameworks to scale agile methods have emerged. A full listing would be beyond the scope of this chapter.

³One of the authors works as an enterprise architect for different clients.

a good discussion of what that really is/means. This task fell to the project team which “should be able to figure that out”. After the project team had struggled with the issue for several months, it became apparent that this was not as easy as it seemed. A cursory root cause analysis showed that (1) there were too many disparate perspectives on the problem, (2) the language that stakeholders used to talk about the problem space varied greatly and (3) some politics were going on in the background as well. It was decided that a smaller group (three professionals) were to come up with a proposal for a definition of what an integral platform is, an ontological question, and a by-and-large overview of what it could look like for this organization – a practical question. Over a period of 3 weeks, we had three meetings (lasting 1.5–2 h each) where we carefully explored the problem area. Within this smaller group, we were able to strictly separate the two questions. The biggest hurdle was to standardize terminology, with questions such as “What is a process?”, “What is a platform?” and “What does ‘integral’ mean?”. With those questions answered, the ontological question was quickly resolved. We also managed to draft a rough outline of (the design of) what we think an integral platform should be and used that to align the views within the larger project team. The project is still on-going, but we can already conclude that a deliberate and brief design dialogue helped to align the views within our project team.

The notion of *enterprise design dialogue* also intends to reflect notions such as authoring “authoring of organizations” [40], as well as views from organizational design [41]. It also acknowledges the fact that an enterprise is certainly not a “machine” (in the sense of [42]) that can be “engineered” as such. In our view, a perspective on an enterprise as an “organism” or even a “learning system” (e.g. [43]) makes more sense, especially when dealing with continuous digital transformations. We do, however, assume that these dialogues result in some artefact that represents some abstraction of some aspect(s) of the design of the enterprise, i.e. an enterprise model in the broadest sense.

In dealing with the many levels and speeds of change that confront enterprises, it will become increasingly important for enterprises to be aware of all relevant activities and activities inside, and outside, the organizational boundaries. Even more, the different actors involved in/impacted by these changes need to (1) have an insight into the existing structures and operations of an enterprise; (2) be able to express, assess and evaluate different design options for their future; and (3) have instructions on how to make the necessary changes to these structures and operations and (4) how to operate in the future.

Mirroring the fact that digital transformation can occur in a top-down as well as a bottom-up fashion, *enterprise design dialogues* may occur bottom-up, but they may also take place as part of an orchestrated enterprise development/transformation process. In the latter case, one may explicitly develop a conversation strategy [44], spanning multiple design dialogues. As suggested in [44], the different steps (i.e. distinct design dialogues) involved in a conversation strategy can serve more specific goals with regard to “enterprise knowledge”, such as *share* (or create) knowledge, *agree to* the shared knowledge and *commit* to the consequences actions resulting from the shared/created knowledge.

Figure 6.2 illustrates our way of thinking. First, note that stakeholders have transformation goals (which may or may not conflict) with regard to a domain. In order to achieve

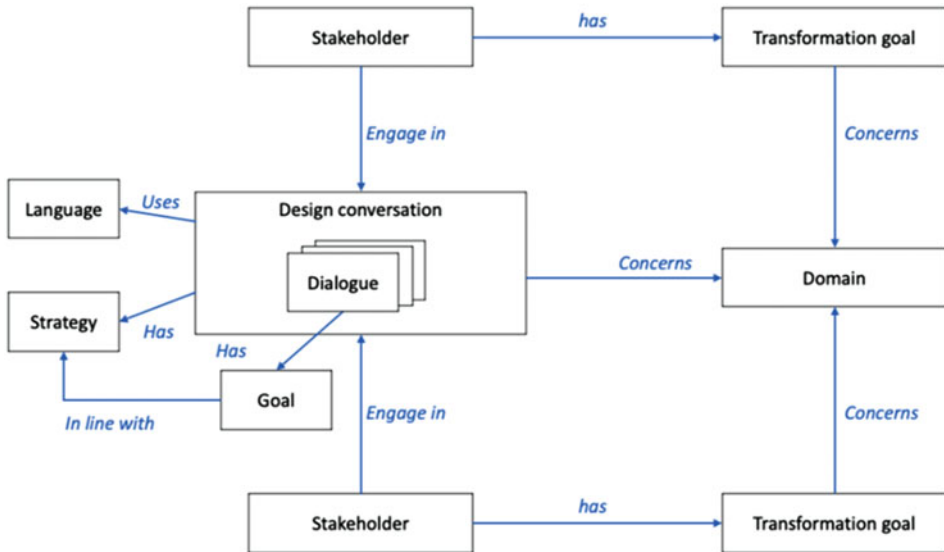


Fig. 6.2 Design conversation and dialogues

these goals, they engage in a design conversation. This conversation uses a (standardized) language and follows a strategy. As part of the conversation, they have different dialogues, each with a specific goal that is in line with the overall strategy. Based on our claim that digital transformation is a *deliberate* effort, we also claim that *design* is an activity that is part of digital transformation initiatives.

The Role of Models

Whenever we, as humans, have a need to (jointly) reason/reflect about some part of an existing/imagined domain, we essentially use models to express our understanding of this (part of the) domain [45], i.e. *domain models*. We take the view that such *domain models* have an important role to play in the design dialogues that shape an enterprise.

Based on the foundational work by, e.g. Apostel [46] and Stachowiak [47], more recent work on the same by different authors [48–51] as well as our own work [45, 52–57], we currently [58] understand a *domain model* to be:

A social artifact that is acknowledged by a collective agent to represent an abstraction of some domain for a particular cognitive purpose.

With *domain*, we refer to “anything” that one can speak and/or reflect about, i.e. the domain of interest. As such, *domain* simply refers to “that what is being modelled”. A model is seen as a *social artefact* in the sense that its role as a model should be recognizable by a

collective agent (e.g. people). The *collective agent* observes the domain by way of their senses and/or by way of (collective) self-reflection and, based on this, should acknowledge/accept the artefact as indeed being a model of the domain (for a given purpose). A model must always be created for some *cognitive purpose*, i.e. to express, specify, learn about or experience knowledge regarding the modelled domain. Finally, a model is the representation of an *abstraction*. This implies that, in line with the *cognitive purpose* of the model, some (if not most) “details” of the domain are consciously filtered out.

In the context of digital transformations, different aspects of an enterprise, including its structures, purpose, value proposition, business processes, stakeholder goals, information systems, etc., can be captured in terms of (interconnected) domain models. The latter also enables “cross-cutting” analysis [3, 59] across different aspects and perspectives.

Models pertaining to any aspect of an enterprise are, by definition, *enterprise models*. Enterprise models typically take the *form* of some “boxes-and-lines” diagram. As argued in, e.g. [60–62], enterprise models should also be understood from a broader perspective than mere “boxes-and-lines” diagrams. As such, domain models can, depending on the *purpose* at hand, take other forms as well, including text, mathematical specifications, games, animations, simulations and physical objects.

Enterprise models are traditionally regarded as an effective way to enable such informed coordination and decision-making. Just as senior management uses financial modelling to *enable* decision-making from a financial perspective, (enterprise) models covering the other aspects of an organization can be used to enable informed decision-making regarding the other aspects as well [36, 38, 63], as well as operational use in Digital Twin(s) and advanced rule-based systems (i.e. for tax law execution, [64, 65]). More generally, as suggested in [61, 66], high-level purposes for the creation of enterprise models include *understand* the current affairs on the enterprise, *assess* the current affairs, *diagnose* possible problems in the current affairs, (re-)*design* changes towards the future, *realize* such changes, provide guidance/direction for (human or digital) actors who *operate* in the enterprise and enable regulators to express regulations in order to *regulate* the activities of the enterprise.

More specifically, enterprise models potentially capture important enterprise knowledge [67]. This can, e.g. pertain to knowledge in relation to the well-known interrogatives (*why, who, whose, when, how, with*), be positioned in time (*as-was, as-is, as-planned, to-be*, etc.), be nuanced in terms of modalities (*must, ought, desired*, etc.), take a prescriptive or a descriptive perspective, etc. As a result, enterprise models can be used to support design dialogues and/or capture the results of design dialogues. Some concrete examples, across different objectives, would be:

- *Coherence of the enterprise*: Models can be used to capture different aspects of an enterprise, as well as their coherence. This was actually also one of the key drivers [68] in the development of the ArchiMate standard for enterprise (architecture) modelling.

- *Engagement of stakeholders*: Models can be used to capture requirements and/or regulations reflecting the needs from different stakeholders. They can also be used to express balanced compromises regarding the positive/negative impacts on the respective goals and concerns (security, regulatory compliance, environmental impact, flexibility, etc.) of stakeholders.
- *Evidence-enabled decision-making*: Models can be used to represent past and current design(s) of an enterprise, its desired future design as well as different options for its future design, all in relation to its (evolving) context. Such models enable among others (1) the analyses of the current, or future, affairs of the enterprise and its environment and (2) the evaluation of the potential impact of design decisions on (new) concerns or (3) assess the compliance of a design with regard to requirements or regulations.
- *General design knowledge*: Models can capture general design knowledge in terms of, e.g. construction theories, design patterns and reference designs (leading to reference models/architectures).

In line with the earlier discussed Cynefin framework, the potential role of models needs to be nuanced towards the specific domain in which the problem fits. For instance, if a problem can be classified as complicated, then the key properties of the problem (and its potential solution) can easily be caught in a model, which – in such a setting – has the interpretation of a simplified (yet relevant) version of reality. In the case of a problem that is classified as complex, models may still be used, but in a more humble role as hypothesis to decide about potential action. In the latter case, the emphasis is on *probing* the organization and *evaluating* results – something that is often referred to as *situational awareness* [32, 33].

Challenges and Opportunities

In this section, we will discuss the challenges and opportunities related to (the use of) design dialogues (as part of modelling initiatives) for digital transformation. We will first discuss the challenges – roughly following the line of reasoning in this chapter – and then the opportunities.

We started this chapter with our definition of digital transformation. This definition, loosely, boils down to changing the *core* of the enterprise with a significant impact on the digital capabilities of the enterprise. To see where the main challenge lies, from this perspective, one has to realize that the organization has been shaped – deliberately or not – the way it is for a reason. We can assume that the designer (i.e. management) of the organization made decisions in the past with the future of the enterprise in mind. The enterprise is the way it is because stakeholders have made their decisions with a bright future in mind, and now we are about to change that. Worse, we are about to embark on a digital transformation journey with uncertain outcome *with the doors open*: we still have to perform the main functions of the enterprise. The consequence is that there is a period of *fluid organization* – meaning parts of the enterprise conform to the old architecture and

parts conform to the new architecture. This is a sub-optimization and may hamper service levels towards customers.

We then argued that digital transformation efforts are *continuous* in nature, which appears to lead to many organizations adopting agile methods with shorter turn-around times to realize (initial version of) capabilities. The challenges related to this aspect are closely related to the previous point, for it entails that the enterprise is continually in flux. There is no such thing as moving from one stable situation to the next, as the transformation is a continuous, on-going process. The consequence is that, at any point in time, we do not have a full understanding of the enterprise. We only have a by-and-large understanding based on what we know of a past situation (which we can analyse because it is in the past) and our knowledge of the on-going transformation initiative(s). This means that, when we make a decision about transformation initiatives or their implications on the enterprise, we do so on an incomplete information position. We base our decisions on what we believe to be true, rather than on what we know to be true. In our view, this emphasizes once more that a thorough understanding of the architecture of the enterprise is crucial: the architecture will change/evolve slowly, whereas (implementation) details change more rapidly as a result of transformation initiatives. Models, of course, are a key enabler to mitigate the risks around this challenge, as they are intended to capture the shared understanding of what we believe to be true.

This brings us to the third point in this chapter: the need for coordination. As we have seen, the enterprise is in a constant state of flux, with uncertainty for all stakeholders involved. We believe that there is no such thing as “the” future of the enterprise that we are working towards: each of the stakeholders has their unique view of what the future should look like. From the perspective of attaining a bright future for the enterprise, the challenge is to align the views of stakeholders as much as possible which is the definition of coordination as used in this chapter. Based on our experience in the field (both authors are/have been active as consultants), we feel justified to conclude that this is rarely the case: in many organizations, stakeholders engage in politics to further their own agenda and maximize their own power/influence rather than achieving the best possible future as seen by the community at large within the enterprise. We should add that “decisive leadership” is sometimes useful or even necessary – but perhaps not at the level that we sometimes see.

As a small example, consider the tension that might occur between stakeholders with a more “risk-averse” mindset and with a more “innovation-driven” mindset. The former group of stakeholders may want to move cautiously from one semi-stable state to another, whereas the latter may desire a more bold approach, taking bigger steps to achieve success. When this tension is not (sufficiently) managed, then conflict/strife and sub-optimal results are bound to occur in the enterprise.

Solving this stakeholder puzzle is beyond the scope of this chapter. Yet, we do believe that models can aid in resolving these puzzles: they offer a focus to key discussion points as part of enterprise design dialogues.

The point of a dialogue is that stakeholders engage in thoughtful and purposeful conversation about the enterprise. This emphasizes the next challenge that is addressed

in this chapter: language. It is well-known that language – particularly getting a shared understanding of an utterance – is notoriously difficult (see, e.g. [69]). When shaping a digital transformation, seemingly small differences in the interpretation of important concepts may have large consequences. For example, take the notion of *causation* (*a* causes *b*). Someone with an engineering background is likely to have a more strict interpretation of what causation really means. When, in a design dialogue, it is claimed that *a* causes *b* and we know that *a* will be changed, then it may occur that one stakeholder logically infers that since *a* is no longer the case, it must be the case that *b* is also no longer the case, whereas the other stakeholder might have a more loose interpretation and conclude that *b* still could be (somewhat) the case. Worse, it may *appear* that these stakeholders are in agreement on some course of action, whereas in fact they are not (since the exact interpretation of their commitments is unclear). This is why we believe that design dialogues should be explicit and argue that the meaning of any key term must be clarified – and models are a good way to do so. We are aware that (a) this takes time and (b) this goes against what agile practitioners are accustomed to – yet we also argue that there is a potentially high “return on modelling effort” [58, 70].

Last but not least, there are challenges related to the notion of *models* and *modelling*. As noted, we see models as a social artefact; stakeholders should be able to examine it and assert whether the model (of a domain) can stand for that domain. We also observed that models are an *abstraction*: details that – according to the modeller(s) – are not relevant are left out. And here lies the challenge: how does one decide what is relevant and what is not? It may appear that this is a trivial choice made by the modeller(s). Yet, the work of Bjeković [71] shows that there is more to it than that: what is relevant is determined by the *goal* of the modeller. Going back to the previous points on design dialogues, we can see how the point on shared understanding of modelling objectives as well as key concepts “propagates”: without the shared understanding, it is hard to decide *collectively* which details to include/leave out.

This leaves the discussion of the opportunities that are to be reaped. In light of the overall theme of this book, we focus on AI-related opportunities and leave other opportunities for future research and exploration.

Recall that we spoke of digital transformation of an *enterprise*, which we defined as “unit of economic organization or activity”. We also expressed that enterprises are *organized* in the sense that actors and other means of production (data, materials, etc.) are used to achieve specific outcomes. In AI-Enabled Enterprises, these actors come in the form of humanoids and AIs which interact to create value. We have noted that, in terms of the Cynefin framework, digital transformations tend to belong to the *complex domain* which implies that no a priori full understanding of that domain can be obtained. This is where an opportunity for AIs comes in: correctly trained AIs may be able to take over a large part of the modelling effort (particularly the “complicated part”) so that human modellers can focus on the truly complex parts.

As an illustration, consider software bots that can “crawl” a network to discover application interfaces (see, e.g. [72]) or mine data to discover how processes work (see,

e.g. [73]). This type of bots exists for other domains as well. It seems safe to assume that in the foreseeable future, bots can be trained to not only discover/mine for processes but also to connect them to form a (detailed) model of the “things that exist” as well as “how they are related”. Perhaps the ability to create “useful abstractions” is a bit far-fetched, but it seems only a matter of time before we are able to achieve such results. This would be of tremendous help for human actors attempting to build up an understanding of the existing enterprise as well as shape a future enterprise: it takes away the burden of having to do a lot of background research.

This brings us to the second opportunity. Recent advances in *chat bots* and related technologies (e.g. [74]) show that meaningful conversations with an AI are available in specific domains. Anyone who has tried to talk their way to an AI over a phone line attempting to resolve business issues is probably well aware that the technology is not yet perfect. Let us assume that the domain we are applying this type of technology to is the domain of digital transformation initiatives. In this case, the AI could be a meaningful and valuable partner that would assist us in creating models with likely future state scenarios and assess impact in terms of digital transformation initiatives [20]. Given the big computational power that an AI has, it should be able to run scenarios and apply heuristics to test which scenarios are most feasible. While useful, we do expect that human judgment remains imperative – even with a well-trained AI (e.g. to ensure that there are no issues around bias/ethics, something we do not expect an AI to resolve for itself).

Conclusion

This chapter started with the observation that the transformation of enterprises towards *AI-Enabled Enterprises* is a logical next phase in the continuous flow of digital transformations which enterprises are (and need to be) engaged in. In line with this, this chapter zoomed in on both the challenges facing enterprises regarding digital transformations in general and the transition to AI-Enabled Enterprises in particular.

In doing so, we argued that digital transformation should be seen as a continuous process while also needing coordination among many different involved stakeholders and activities, as such resulting in *coordinated continuous digital transformation*. We then positioned *enterprise design dialogues* as being at the heart of the needed coordination of transformations while then also positioning enterprise models as a key artefact in support of enterprise design dialogues. Finally, we reviewed some of the challenges and opportunities towards future research.

References

1. Proper, H. A., Winter, R., Aier, S., & de Kinderen, S. (Eds.). (2018). *Architectural coordination of enterprise transformation* (The Enterprise Engineering Series). Springer. <https://doi.org/10.1007/978-3-319-69584-6>. ISBN 978-3-319-69583-9.
2. Proper, H. A., van Gils, B., & Haki, K. (Eds.). (2023). *Digital enterprises – Service-focused, digitally-powered, data-fueled* (EE Series). Springer. ISBN 978-3-031-30213-8.
3. Land, M. O., Proper, H. A., Waage, M., Cloo, J., & Steghuis, C. (2008). *Enterprise architecture – Creating value by informed governance* (The Enterprise Engineering Series). Springer. <https://doi.org/10.1007/978-3-540-85232-2>. ISBN 978-3-540-85231-5.
4. van Gils, B., Hoppenbrouwers, S. J. B. A., & Proper, H. A. (2022). Conceptual modeling in digital transformations – Enabling enterprise design dialogues. In *PoEM 2022 Forum Proceedings*. CEUR-WS.org.
5. Ross, J., Beath, C. M., & Mocker, M. (2019). *Designed for digital: How to architect your business for sustained success* (Management on the cutting edge). MIT Press.
6. Meriam–Webster. (2003). *Meriam–Webster Online, Collegiate Dictionary*. <http://www.merriam-webster.com>
7. Veeke, H. P. M., Ottjes, J. A., & Lodewijks, G. (2008). *The Delft systems approach: Analysis and design of industrial systems*. Springer Science & Business Media.
8. Proper, H. A., & van Gils, B. (2019). Enterprise modelling in the age of digital transformation. *IFIP Select*, 1(2). http://ifip.org/select/select_1_2/proper%201_2.pdf.
9. Scott, J. (2009). Business Capability Maps – The missing link between business strategy and IT action. *Architecture & Governance*, 5(9), 1–4.
10. Teece, D. J., Pisano, G., & Shuen, A. (1997). Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7), 509–533.
11. Teece, D. J. (2007). Explicating dynamic capabilities: The nature and of (sustainable) enterprise performance. *Strategic Management Journal*, 4(28), 1319–1350.
12. Greefhorst, D., & Proper, H. A. (2011). *Architecture principles – The cornerstones of enterprise architecture* (The Enterprise Engineering Series). Springer. <https://doi.org/10.1007/978-3-642-20279-7>. ISBN 978-3-642-20278-0.
13. IEEE Computer Society. (2000). *IEEE std 1471-2000: IEEE recommended practice for architecture description of software-intensive systems*.
14. Bennett, N., James, G., & Lemoine. (2014). What VUCA really means for you. *Harvard Business Review* (January-February), 27.
15. Johansen, B., & Euchner, J. (2013). Navigating the vuca world. *Research-Technology Management*, 56(1), 10–15.
16. Horan, T. A. (2000). *Digital Places – Building our city of bits*. The Urban Land Institute (ULI). ISBN 0-874-20845-9.
17. Negroponte, N. (1996). *Being digital*. Vintage Books. ISBN 0-679-76290-6.
18. Proper, H. A., Guédria, W., & Sottet, J.-S. (2020). Enterprise modelling in the digital age; Chapter 3. In V. Kulkarni, S. Reddy, T. Clark, & B. S. Barn (Eds.), *Advanced digital architectures for model-driven adaptive enterprises* (pp. 46–67). IGI Global. <https://doi.org/10.4018/978-1-7998-0108-5.ch003>. ISBN 9781799801085.
19. Burgueño, L., Cabot, J., & Gérard, S. (2019). An LSTM-based neural network architecture for model transformations. In *MODELS 2019* (pp. 294–299). IEEE Explore.
20. Feltus, C., Ma, Q., Proper, H. A., & Kelsen, P. (2021). Towards AI assisted domain modeling. In I. Reinhartz-Berger & S. W. Sadiq (Eds.), *Advances in Conceptual Modeling ER 2021 Workshops CoMoNoS, EmpER, CMLS, St. John's, NL, Canada, October 18-21, 2021, Proceedings* (Lecture Notes in Computer Science) (Vol. 13012). Springer. ISBN 978-3-030-88357-7.

21. Snoeck, M., Stirna, J., Weigand, H., & Proper, H. A. (2019). Panel discussion: Artificial intelligence meets enterprise modelling (summary of panel discussion). In C. Feltus, P. Johannesson, & H. A. Proper (Eds.), *Proceedings of the Practice of Enterprise Modelling 2019 Conference Forum (short papers), Luxembourg, November 27-29, 2019* (CEUR Workshop Proceedings) (Vol. 2586, pp. 88–97). CEUR-WS.org. <http://ceur-ws.org/Vol-2586/paper8.pdf>
22. Proper, H. A., Wagter, R., & Bekel, J. (2022). On enterprise coherence governance with gea: A 15-year co-evolution of practice and theory. *Software and Systems Modeling*. <https://doi.org/10.1007/s10270-022-01059-0>
23. Henderson, J. C., & Venkatraman, N. (1993). Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, 32(1), 4–16.
24. Mulder, H., Johnson, J., Meijer, K., & Crear, J. Investigating 5,140 digital transformation projects. In Proper et al. [2]. Forthcoming.
25. Mulder, J. B. F., & Johnson, J. (2016). The next step of IT project research in practice: The CHAOS university system. In *The 10th International Multi- Conference on Society, Cybernetics and Informatics: IMSCI 2016, July 5 – 8, Orlando, FL*.
26. Kurtz, C. F., & Snowden, D. (2003). The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM Systems Journal*, 42(3), 462–483.
27. Mark, A., & Snowden, D. (2006). Researching practice or practicing research: Innovating methods in healthcare – The contribution of cynefin. *Innovations in Health Care*.
28. Snowden, D., & Rancati, A. (2021). *Managing complexity (and chaos) in times of crisis. A field guide for decision makers inspired by the cynefin framework*. *Information Society*.
29. Snowden, D. J., & Boone, M. E. (2007). A leader’s framework for decision making. *Harvard Business Review*, 85(11), 68–76.
30. Edwards, J. B. (2014). *The burden of command*. CreateSpace Independent Publishing Platform. ISBN 978-1495240447.
31. Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, 4, 155–169.
32. Endsley, M. R. (2001). Designing for situation awareness in complex systems. In *Proceedings of the Second International Workshop on symbiosis of humans, artifacts and environment* (pp. 1–14).
33. Nofi, A. A. (2000). *Defining and measuring shared situational awareness*. Technical report. Center for Naval Analyses, Alexandria, VA.
34. Wagter, R., & Proper, H. A. Involving the right stakeholders – Enterprise coherence governance. In Proper et al. [1], chapter 10, pp. 99–110. ISBN 978-3-319-69583-9. <https://doi.org/10.1007/978-3-319-69584-6>.
35. Conklin, J. (2005). *Dialogue mapping: Building shared understanding of wicked problems*. Wiley. ISBN 978-0-470-01768-5.
36. Proper, H. A., & Lankhorst, M. M. (2014). Enterprise Architecture – Towards essential sensemaking. *Enterprise Modelling and Information Systems Architectures*, 9(1), 5–21. <https://doi.org/10.18417/emisa.9.1.1>
37. Waterman, M., Noble, J., & Allan, G. (2015). How much up-front? A grounded theory of agile architecture. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 1, pp. 347–357). IEEE.
38. Harmsen, A. F., Proper, H. A., & Kok, N. (2009). Informed governance of enterprise transformations. In H. A. Proper, A. F. Harmsen, & J. L. G. Dietz (Eds.), *Advances in Enterprise Engineering II – First NAF Academy Working Conference on Practice-Driven Research on Enterprise Transformation, PRET 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 11, 2009. Proceedings* (Lecture Notes in Business Information Processing) (Vol. 28, pp. 155–180). Springer. https://doi.org/10.1007/978-3-642-01859-6_9. ISBN 978-3-642-01858-9.

39. Junginger, S. (2015). Organizational design legacies & service design. *Design Journal*. Special Issue: Emerging Issues in Service Design.
40. Taylor, J. R. (1996). The communicational basis of organization: Between the conversation and the text. *Communication Theory*, 6(1), 1–39.
41. Magalhães, R. (Ed.). (2014). *Organization design and engineering: Coexistence, cooperation or integration*. Palgrave-Macmillan. ISBN 13 9781137351562.
42. Morgan, G. (1998). *Images of organization*. Sage. ISBN 0-761-91752-7.
43. Senge, P. M. (1997). The fifth discipline. *Measuring Business Excellence*.
44. Proper, H. A., Hoppenbrouwers, S. J. B. A., & Veldhuijzen van Zanten, G. E. (2017). Communication of enterprise architectures. In *Enterprise Architecture at Work – Modelling, Communication and Analysis* (The Enterprise Engineering Series) (4th ed., pp. 59–72). Springer. https://doi.org/10.1007/978-3-662-53933-0_4. ISBN 978-3-662-53932-3.
45. Proper, H. A., & Guizzardi, G. (2021). On domain conceptualization. In D. Aveiro, G. Guizzardi, R. Pergl, & H. A. Proper (Eds.), *Advances in Enterprise Engineering XIV – 10th Enterprise Engineering Working Conference, EEWC 2020, Bozen-Bolzano, Italy, September 28, October 19, and November 9-10, 2020, Revised Selected Papers* (Lecture Notes in Business Information Processing) (Vol. 411, pp. 49–69). Springer. https://doi.org/10.1007/978-3-030-74196-9_4. ISBN 978-3-030-74195-2.
46. Apostel, L. (1960). Towards the formal study of models in the non-formal sciences. *Synthese*, 12, 125–161.
47. Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Springer. <https://doi.org/10.1007/978-3-7091-8327-4>. ISBN 3-211-81106-0.
48. Harel, D., & Rumpe, B. (2004). Meaningful modeling: What’s the semantics of “semantics”? *IEEE Computer*, 37(10), 64–72. <https://doi.org/10.1109/MC.2004.172>
49. Rothenberg, J. (1989). The nature of modeling. In L. E. Widman, K. A. Loparo, & N. Nielson (Eds.), *Artificial intelligence, simulation & modeling* (pp. 75–92). Wiley. ISBN 0-471-60599-9.
50. Sandkuhl, K., Fill, H.-G., Hoppenbrouwers, S. J. B. A., Krogstie, J., Matthes, F., Opdahl, A. L., Schwabe, G., Uludağ, O., & Winter, R. (2018). From expert discipline to common practice: A vision and research agenda for extending the reach of enterprise modeling. *Business & Information Systems Engineering*, 60(1), 69–80. <https://doi.org/10.1007/s12599-017-0516-y>
51. Thalheim, B. (2011). The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In *Handbook of conceptual modeling* (pp. 543–577). Springer.
52. Bjeković, M., Proper, H. A., & Sottet, J.-S. (2014). Embracing pragmatics. In E. S. K. Yu, G. Dobbie, M. Jarke, & S. Purao (Eds.), *Conceptual Modeling – 33rd International Conference, ER 2014, Atlanta, GA, USA, October 27-29, 2014. Proceedings* (Lecture Notes in Computer Science) (Vol. 8824, pp. 431–444). Springer. https://doi.org/10.1007/978-3-319-12206-9_37. ISBN 978-3-319-12205-2.
53. Guarino, N., Guizzardi, G., & Mylopoulos, J. (2020). On the philosophical foundations of conceptual models. *Information Modelling and Knowledge Bases XXXI*, 321, 1.
54. Guizzardi, G. (2006). On ontology, ontologies, conceptualizations, modeling languages, and (meta)models. In O. Vasilecas, J. Eder, & A. Caplinskas (Eds.), *Databases and Information Systems IV – Selected Papers from the Seventh International Baltic Conference, DB&IS 2006, July 3-6, 2006, Vilnius, Lithuania (Frontiers in Artificial Intelligence and Applications)* (Vol. 155, pp. 18–39). IOS Press. ISBN 978-1-58603-715-4.
55. Hoppenbrouwers, S. J. B. A., Proper, H. A., & van der Weide, T. P. (2005). A fundamental view on the process of conceptual modeling. In L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, & O. Pastor (Eds.), *Conceptual Modeling – ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings* (Lecture Notes in Computer



- Science) (Vol. 3716, pp. 128–143). Springer. https://doi.org/10.1007/11568322_9. ISBN 3-540-29389-2.
56. Proper, H. A., & Guizzardi, G. (2020). On domain modelling and requisite variety – Current state of an ongoing journey. In J. Grabis & D. Bork (Eds.), *The Practice of Enterprise Modeling. PoEM 2020* (Lecture Notes in Business Information Processing) (Vol. 400, pp. 186–196). Springer. https://doi.org/10.1007/978-3-030-63479-7_13. ISBN 978-3-030-63479-7.
57. Proper, H. A., Verrijn-Stuart, A. A., & Hoppenbrouwers, S. J. B. A. (2005). On utility-based selection of architecture-modelling concepts. In S. Hartmann & M. Stumptner (Eds.), *Conceptual Modelling 2005, Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005), Newcastle, NSW, Australia, January/February 2005 (Conferences in Research and Practice in Information Technology Series)* (Vol. 43, pp. 25–34). Australian Computer Society. ISBN 1-920682-25-2. <http://crpit.scem.westernsydney.edu.au/abstracts/CRPITV43Proper.html>.
58. Proper, H. A., & Guizzardi, G. (2022). Modeling for enterprises; Let's go to RoME Via RiME. In *PoEM 2022 Forum Proceedings*. CEUR-WS.org.
59. Jonkers, H., Lankhorst, M. M., Quartel, D. A. C., Proper, H. A., & Iacob, M.-E. (2011). ArchiMate for integrated modelling throughout the architecture development and implementation cycle. In B. Hofreiter, E. Dubois, K.-J. Lin, T. Setzer, C. Godart, H. A. Proper, & L. Bodenstaff (Eds.), *13th IEEE Conference on Commerce and Enterprise Computing, CEC 2011, Luxembourg-Kirchberg, Luxembourg, September 5-7, 2011* (pp. 294–301). IEEE Computer Society Press. <https://doi.org/10.1109/CEC.2011.52>. ISBN 978-0-769-54535-6. <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6046209>.
60. Magalhães, R., & Proper, H. A. (2017). Model-enabled design and engineering of organisations. *Organisational Design and Enterprise Engineering*, 1(1), 1–12. <https://doi.org/10.1007/s41251-016-0005-9>
61. Proper, H. A. (2021). *On model-based coordination of change in organizations* (pp. 79–98). Springer. https://doi.org/10.1007/978-3-030-84655-8_6. ISBN 978-3-030-84655-8.
62. Proper, H. A., & Bjeković, M. (2019). Fundamental challenges in systems modelling. In H. C. Mayr, S. Rinderle-Ma, & S. Strecker (Eds.), *40 Years EMISA 2019, May 15-17, 2019, Evangelische Akademie Tutzing, Germany* (Lecture Notes in Informatics) (Vol. P-304, pp. 13–28). Gesellschaft für Informatik e.V.. <https://dl.gi.de/20.500.12116/33130>
63. Proper, H. A. (2014). Enterprise architecture: Informed steering of enterprises in motion. In S. Hammoudi, J. Cordeiro, L. A. Maciaszek, & J. Filipe (Eds.), *Enterprise Information Systems – 15th International Conference, ICEIS 2013, Angers, France, July 4-7, 2013, Revised Selected Papers* (Lecture Notes in Business Information Processing) (Vol. 190, pp. 16–34). Springer. https://doi.org/10.1007/978-3-319-09492-2_2. ISBN 978-3-319-09491-5.
64. Ausems, A., Bulles, J., & Lokin, M. (2021). *Wetsanalyse voor een werkbare uitvoering van wetgeving met ICT*. Boom. ISBN 978-9-462-90937-3. In Dutch.
65. Corsius, M., Hoppenbrouwers, S., Lokin, M., Baars, E., Sangers-Van Cappellen, G., & Wilmont, I. (2021). RegelSprak: A CNL for executable tax rules specification. In *Proceedings of the Seventh International Workshop on Controlled Natural Language (CNL 2020/21)*. Special Interest Group on Controlled Natural Language.
66. Proper, H. A. (2020). Digital enterprise modelling – Opportunities and challenges. In B. Roelens, W. Laurier, G. Poels, & H. Weigand (Eds.), *Proceedings of 14th International Workshop on Value Modelling and Business Ontologies, Brussels, Belgium, January 16-17, 2020* (CEUR Workshop Proceedings) (Vol. 2574, pp. 33–40). CEUR-WS.org. <http://ceur-ws.org/Vol-2574/short3.pdf>
67. Lillehagen, F., & Krogstie, J. (2010). *Active knowledge modeling of enterprises*. Springer. ISBN 978-3-540-79415-8.

68. Lankhorst, M. M., Proper, H. A., & Jonkers, H. (2010). The anatomy of the ArchiMate language. *International Journal of Information System Modeling and Design*, 1(1), 1–32. <https://doi.org/10.4018/jismd.2010092301>
69. Ogden, C. K., & Richards, I. A. (1923). *The meaning of meaning – A study of the influence of language upon thought and of the science of symbolism*. Magdalene College, University of Cambridge.
70. Guizzardi, G., & Proper, H. A. (2021). On understanding the value of domain modeling. In G. Guizzardi, T. P. Sales, C. Griffo, & M. Furnagalli (Eds.), *Proceedings of 15th International Workshop on Value Modelling and Business Ontologies (VMBO 2021), Bolzano, Italy, 2021* (CEUR Workshop Proceedings) (Vol. 2835). CEUR-WS.org. <http://ceur-ws.org/Vol-2835/paper6.pdf>
71. Bjeković, M. (2017). *Pragmatics of enterprise modelling languages: A framework for understanding and explaining*. PhD thesis, Radboud Universiteit Nijmegen.
72. Shestakov, D. (2009). On building a search interface discovery system. In *International Workshop on Resource Discovery* (pp. 81–93). Springer.
73. van der Aalst, W. (2012). Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 3(2), 1–17.
74. Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. In *IFIP International Conference on Artificial Intelligence Applications and Innovations* (pp. 373–383). Springer.



A Case Study: Wellness Ecosystem

7

Vinay Kulkarni  and Sreedhar Reddy 

Introduction

Modern businesses operate in a dynamic, uncertain environment where they are continually subjected to unpredictable changes in the business domain, consumer needs, laws and regulations and even technology. Business enterprises need to be able to adapt to these changes, continue to operate, stay relevant and thrive despite the dynamism and uncertainty [1]. In understanding the dynamism in their environment, enterprises need to be viewed in the context of the larger business ecosystems they are part of.

As shown in Fig. 7.1, an ecosystem comprises multiple interacting stakeholders such as business enterprises, suppliers, service, sales and other partner organizations, consumers, regulators, etc. [2]. Each stakeholder has a set of objectives to be met, devises a set of strategies to achieve these objectives and brings to the table a set of capabilities to realize these strategies. Working collaboratively enables them to deliver greater value to the customer than they could individually, giving them greater market access and brand value. In a dynamic world characterized by partial information and uncertainty, stakeholder strategies and capabilities need to be suitably adapted on a continuous basis while also being mindful of the fact that stakeholder objectives could be complementary or conflicting. This calls for sharing of the right information between the various stakeholders. This won't scale up if done individually where each stakeholder needs to approach the rest of the stakeholders for the relevant information. In the least, we need a value integrator that will acquire and share the right information among the stakeholders. In a dynamic world where stakeholders as well as their objectives can change rapidly, this integration needs to happen continuously. Going beyond just information sharing, one can think of the value

V. Kulkarni (✉) · S. Reddy

Tata Consultancy Services Research, Pune, Maharashtra, India

e-mail: vinay.vkulkarni@tcs.com

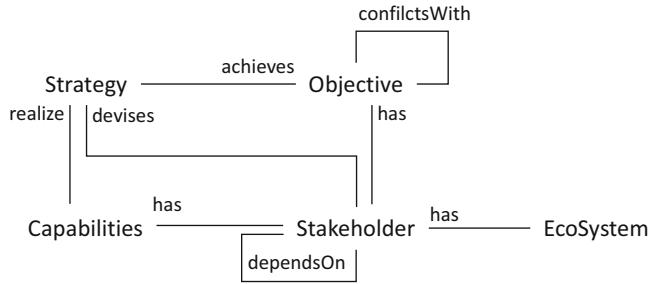


Fig. 7.1 Enterprise ecosystem

integrator providing adaptation support to stakeholders for devising the right strategies and their effective operationalization through the right capabilities [3]. This can be in the form of advisory via a recommender system or automated via an adaptation engine. This adaptation also needs to happen on a continuous basis in a dynamic world.

We illustrate these ideas using an example from the wellness domain.

Wellness Ecosystem

Wellness needs of individual consumers encompass fitness, healthcare, nutrition, leisure as well as health insurance. The wellness ecosystem comprises service providers for these needs; the consumer population consisting of individuals, families and corporates or communities; government and regulatory bodies; and manufacturers of wellness products as well as their networks of partner organizations, some of which are depicted in Fig. 7.2.

Wellness Stakeholders

We summarize the objectives, capabilities and adaptation needs of various stakeholders in wellness ecosystem.

Individual

Objectives An individual would want to maximize fitness and minimize hospital expenses while achieving good work-life balance.

Capabilities of interest include overall health, ability to handle stress, capacity to exercise and discipline.

Adaptation drivers include age, health disorders, financial situation, fitness aspirations and lifestyle changes.

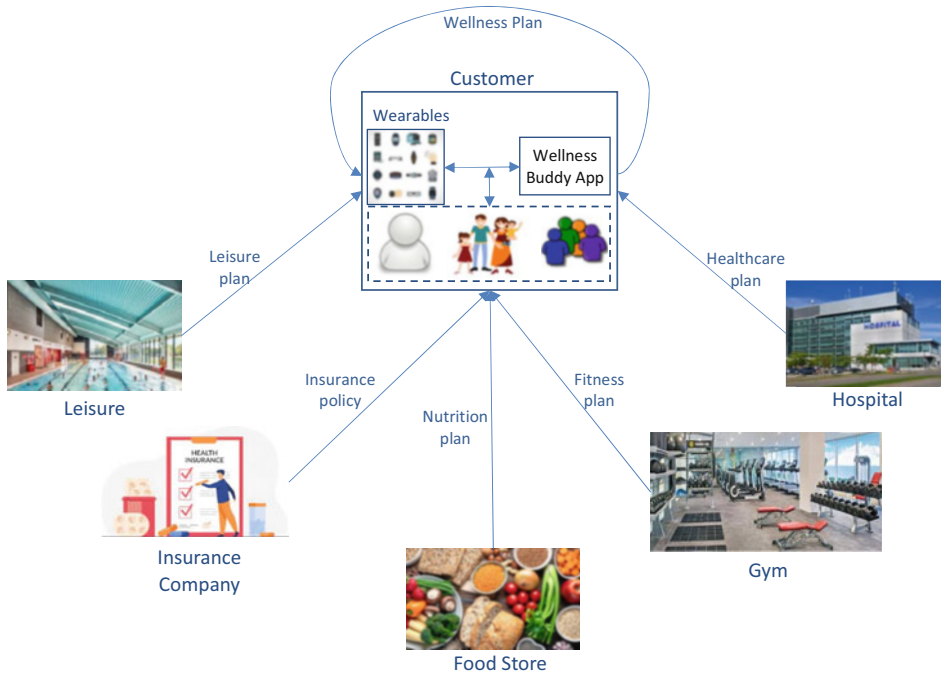


Fig. 7.2 Wellness ecosystem

Adaptive response An individual may want the wellness plan to be refined which includes choice of fitness regime, nutrition plan, health check-ups, health insurance, leisure activities and corresponding service providers.

Gym

Objectives A gym would want to maximize customer footfall and satisfaction, maximize RoI in fitness equipment and facilities and provide the right set of services to achieve these objectives.

Capabilities include fitness equipment and facilities, fitness packages and staff.

Adaptation drivers include demographics, fitness needs, competitor actions and black swan events like wars, pandemics, etc.

Adaptive response A gym may want to alter its size and layout, quantity and quality of fitness equipment, size and skill profile of training staff, fitness packages, etc.

Food Store

Objectives Principal objectives are maximizing customer satisfaction, maximizing sale and profit, eco-friendly packaging, minimizing energy consumption and wastage towards sustainability goals, etc.

Capabilities include shelf space, number of brands and products, strategies for product placement, pricing and promotion and size and quality of salesforce.

Adaptation drivers are demographics, consumer needs, economic profile of the neighbourhood, regulatory regime, competitor actions, etc.

Adaptive response of a food store comprises changes to shelf space and layout, product pricing and promotion strategies, supply chain and size and skill profile of salesforce.

Hospital

Objectives A hospital would want to service as many patients as possible without compromising on the quality of the healthcare; maximize RoI in healthcare facility, diagnostic centres, medical equipment and size as well as skill profile of staff; and provide the right set of services to achieve these objectives.

Capabilities include size and skill profile of staff, healthcare packages, medical equipment and diagnostic centres and facilities.

Adaptation drivers include demographics, competitor actions and black swan events like war, pandemics, etc.

Adaptive response A hospital may want to alter its size and profile in terms of diagnostics and equipment, quantity and quality of medical equipment, size and skill profile of healthcare staff, healthcare packages, etc.

Insurance Company

Objectives Principal objectives are maximization of sales, profit, customer lifetime value, average revenue per customer and customer satisfaction while minimizing customer churn and rationalization of product portfolio.

Capabilities include insurance products covering various types of risks; services such as underwriting, claims processing, policy administration, etc.; and product creation, marketing and servicing staff.

Adaptation drivers include demographics, health conditions, fitness levels and insurance products.

Leisure Provider

Objective Principal objectives are maximization of measures such as sales, profit, customer satisfaction and RoI in facilities.

Capabilities include leisure facilities such as resorts, theatres, cinema halls, restaurants, etc., size and skill profile of staff and the right set of services around these facilities.

Adaptation drivers include demographics, state of economy, competitor actions and black swan events like war, pandemics, etc.

Adaptive response A leisure provider may want to alter the leisure packages and their price points, expand facilities and resize and reskill the staff.

Services in each of the areas of wellness such as healthcare, fitness, nutrition and leisure complement one another and are also inter-dependent, requiring action to be taken on multiple fronts to satisfy an individual's wellness needs. For example, the health of an

individual is impacted by his fitness and nutrition habits. Similarly, fitness regime needs to cater to individual's fitness needs while taking into consideration overall health and dietary habits. Dietary needs are a function of individual's overall health and fitness regime. A change in any one aspect impacts the other related aspects, requiring them to change appropriately so that overall wellness objectives continue to be met.

A customer who aspires for wellness would have to interact separately with a fitness provider, food stores and healthcare provider for his needs in each of these areas. He has to manage the inter-dependence and dynamism among these aspects himself. For example, a switch from active to sedentary job would necessitate commensurate changes to fitness regime and nutrition intake. The customer might consult the appropriate nutrition and fitness experts for guidance; however, it is apparent that their advice would not work in isolation. Rather, healthcare, nutrition and fitness experts need to *collaborate* to decide upon the course of action that would truly work to satisfy the customer's needs. Although individual service providers may fulfil their goal of providing healthcare/nutrition/fitness services, their value proposition is complete only in conjunction. Changes introduced in their services by any of the collaborating participants are another potential source of dynamism in the ecosystem. Moreover, wellness needs of individuals change over time with age, change in lifestyle, family commitments, etc. As a result, the wellness plans as well as their implementations need to adapt suitably and continuously over time.

A service provider aspires to provide fit-for-purpose services aimed principally at maximizing the return on investment which in turn depends on customer satisfaction and maximum utilization of facilities. Current practice is to classify the target clientele into broad buckets and defining services catering to these buckets. Typically, these buckets are formed by taking into consideration multiple dimensions of interest, e.g. an insurance company will use the dimensions of age, occupation, medical history and fitness level to define buckets of interest. This definition will make use of data when available and will resort to guesstimate when data is not available. In addition, they arrive at correlations across these dimensions, e.g. individuals in sedentary occupations tend to have low fitness levels and low fitness level correlates to onset of health conditions such as coronary artery disease, diabetes and hypertension later in life. Insurance company designs insurance products for this bucket taking into consideration probability of onset of these health conditions at certain age. Access to correct data thus minimizing guesswork will lead to definition of insurance product that's beneficial to both the insurer and the insured. However, this still has some lacunae. This strategy is ignorant of the fact that correlations may change over time and hence bucket-level analysis-based product definition needs to adapt over time. Also, this strategy of product definition may not fit all individuals in the bucket equally well. It would be desirable if the product definition is customisable to individual's profile and needs.

Similarly, a gym may decide to invest in a set of exercise equipment based on broad bucketing of the target clientele such as youngsters, middle-aged white-collar workers, retired professionals, etc. Here, the assumption might be that only youngsters will opt for vigorous exercises such as weight training, retired professionals will opt for yoga and

middle-aged clients will opt for light jogging. However, it's quite common that some youngsters will choose yoga, whereas as some middle-aged customers may choose weight training. Such mismatches between the assumptions and ground reality will lead to sub-optimal provisioning and utilization of exercise equipment. Moreover, the gym needs to be cognizant of the overall health and health conditions of individual and the effect the prescribed fitness regime may have on the individual. First, the gym may not have access to this information, and second, the gym may not be in a position to assess health ramifications of the prescribed fitness regime. Thus, in short, the gym by itself can at best cater to only one aspect of wellness and that too in a sub-optimal manner. Finally, the gym also needs to be cognizant of other competing fitness facilities in the neighbourhood. This too is fraught with uncertainty as only partial information about the competitors is available. As wellness needs of individuals change over time, services offered by the gym need to adapt suitably and continuously over time.

All service providers face similar situations characterized by partial and uncertain information about customers and competitors and changing customer needs and competition landscape over time. Would it not be nice if there exists a value integrator that facilitates:

- Sharing of relevant information among the service providers
- Integration of the services offered by the various service providers to meet the wellness needs of individual customers
- Effective planning of capabilities and devising strategies for the various service providers to help maximize ROI

We posit such value integrator, wellness provider, as an adaptive enterprise that helps effective adaptation of all stakeholders in the wellness ecosystem [4].

Wellness Provider

The wellness provider is a central entity that provides an integration platform for all stakeholders and services so that customers now need to interact with a single entity for their wellness needs. The wellness provider orchestrates services from individual service providers, taking care of inter-dependencies and ensuring necessary information is made available to each entity while safeguarding security and privacy policies. As a facilitator, it tracks customer and service provider needs and issues at each step and ensures optimum collaboration between them to achieve objectives of each entity.

In order to realize this vision, the wellness provider will need to:

- Onboard a wide range of service providers in each category to cater to needs and preferences
- Capture information about each service provider and their services and inter-dependencies
- Acquire knowledge about customers' needs, preferences and behaviours

- Provide appropriate choices to customers based on their preferences
- Schedule services optimally for meeting customer requirements while honouring ecosystem-wide constraints
- Monitor customer and service providers continuously to learn from their behaviour and interactions
- Make intelligent adaptation decisions based on available information
- Alter its orchestration appropriately to continue meeting objectives

As a result, the wellness provider helps each stakeholder benefit from the collaboration as they gain access to the right information and insights and get advice to suitably adapt their capabilities, thus leading to improved services. The wellness provider also enables the ecosystem as a whole to deliver far greater value to all stakeholders through intelligent orchestration. Customers get a cohesive solution for all their wellness needs customized to their individual goals and preferences that will adapt with changing situations. Being part of the ecosystem, all service providers have the added advantage of enhancing their customer base. Uncertainty in the environment may cause disruption affecting various stakeholders in the ecosystem. The wellness provider facilitates a coordinated adaptive response to this disruption, thus ensuring a smooth and speedy transition to a stable state.

We propose these capabilities be supported through a wellness platform that provides the means to:

- Collect and disseminate the relevant information of stakeholders
- Capture stakeholder needs and their objectives in a machine-processable form
- Support decision-making via simulation, knowledge-based reasoning, statistical reasoning, etc.
- Effect the decisions through software systems in an efficacious manner
- Capture the relevant knowledge and keep it up to date
- Learn from past executions and external knowledge sources
- Integrate the above capabilities to support dynamic adaptation

Figure 7.3 depicts such a platform which we now elaborate in detail.

Decision-Making in Dynamic and Uncertain Environment

In essence, decision-making is the process of choosing the right course of action from a set of possible alternatives so as to meet the stated goals. The decision-making in a system is called upon when there is a change either in its capabilities or environment or goals. This typically requires exploration among the alternatives. Ideally, this is best supported “in silico”, thus optimizing on time, cost and effort. Moreover, this eliminates consequences of incorrect choices in the real world.

We propose the decision-making process to be viewed as a navigation over Goal-Measures-Levers (GML) graph shown in Fig. 7.4a. It depicts the goal decomposition

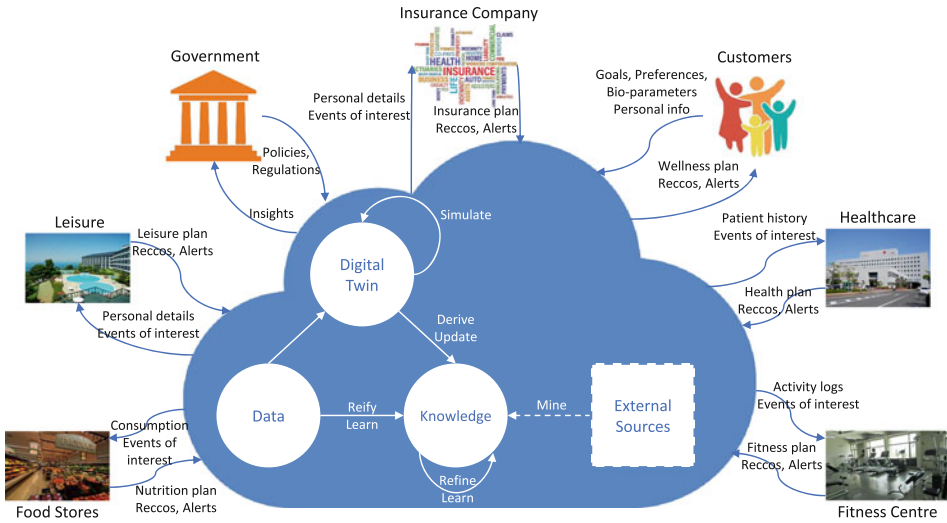


Fig. 7.3 The wellness platform

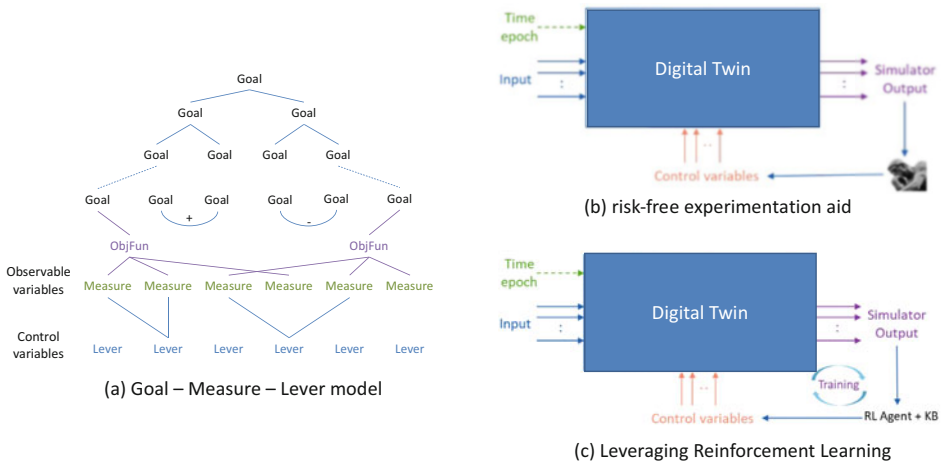


Fig. 7.4 Digital twin-based decision-making. (a) Goal—Measure—Lever model, (b) risk-free experimentation aid, (c) leveraging Reinforcement Learning

making explicit the relationships between goals. Goal is viewed as an objective function over a set of measures, i.e. observable output variables of the system. These measures are influenced by a set of levers, i.e. controllable input variables (or capabilities) of the system. Thus, supporting the decision-making process involves varying the levers, observing the effect on measures, checking if the stated goals are met and iterating till the goals are met or the capabilities are fully explored [5].

We propose digital twin as a purposive representation of the system that supports this decision-making process “in silico” [6]. The digital twin can be used as a risk-free experimentation aid (refer to Fig. 7.4b) wherein a Subject Matter Expert (SME) subjects the digital twin to a set of what-if scenarios to identify the right settings for the right levers that lead to satisfaction of the stated goal [7]. Selecting the right set of scenarios, checking if the goals are met and identifying the right levers are intellect-intensive activities. This analysis and synthesis burden on SME can be reduced by bringing in Reinforcement Learning (RL) and knowledge-guided exploration of solution space [8] as shown in Fig. 7.4c.

Effecting the Decisions in Software

With enterprises relying more and more on software for automating business operations, decisions are implemented in software systems. Hence, software needs to change suitably when new decisions are made. Typically, this is a time-, cost- and intellect-intensive endeavour, the principal reason being the current software engineering technology enforces determinism on specification even when the requirements are nebulous. In addition, assumptions about the environment get hardcoded into the software implementation, thus making it brittle to change. Moreover, the requirements are arrived at based upon knowledge at that point in time which is typically incomplete and possibly uncertain as shown in Fig. 7.5a.

Current software system architectures do not provide any means to capture and reason with knowledge. As a result, when software does not behave as expected, the only recourse is to rely on human experts who need to analyse the current state, possibly acquire the necessary knowledge and adapt the system suitably. This is a time-, effort- and intellect-intensive endeavour.

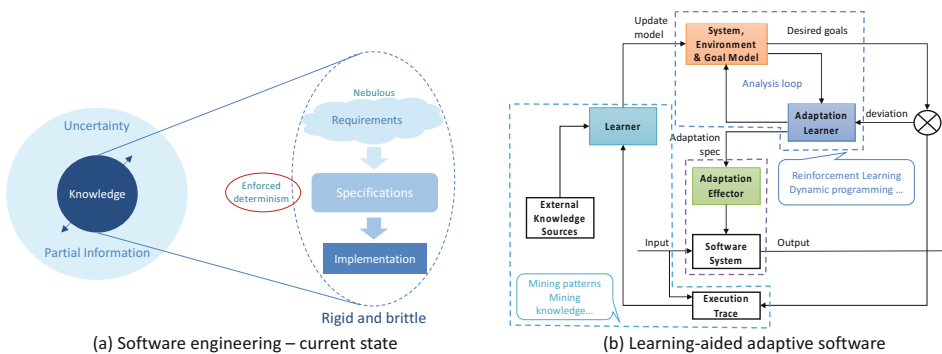


Fig. 7.5 Effecting decisions in software. (a) Software engineering—current state, (b) learning-aided adaptive software

We propose a learning-aided adaptive architecture [9] as shown in Fig. 7.5b. The key components are system, environment and goal (SEG) model, adaptation learner, adaptation effector and learner. The SEG model is in essence a digital twin of the software system and its operating environment where all alternatives are open for exploration. Adaptation learner component monitors the system output with respect to the desired goals. When the goals are not met, the adaptation loop kicks in to identify the interventions to be introduced into the system behaviour through intelligent simulation of the SEG model. Adaptation effector component introduces the identified interventions into the software. Learner component enhances the existing knowledge through the analysis of system execution traces as well as extraction of relevant knowledge from external sources [10].

Effecting the Decisions in Business Processes

With enterprises relying more and more on automated business processes, these processes need to change suitably when new decisions are made [11]. Typically, this is a time-, cost- and intellect-intensive endeavour, the principal reason being the current business process engineering technology enforces determinism on the orchestration of processes even when the requirements are nebulous. In addition, assumptions about the environment get hardcoded into the process implementation, thus making it brittle to change. Moreover, the requirements are arrived at based upon knowledge at that point in time which is typically incomplete and possibly uncertain as shown in Fig. 7.6.

Current business process implementation does not provide any means to capture and reason with knowledge [12]. As a result, when a process does not behave as expected, the only recourse is to rely on human experts who need to analyse the current state, possibly acquire the necessary knowledge and adapt the system suitably. This is a time-, effort- and intellect-intensive endeavour.

We leverage the learning-aided adaptive architecture of Fig. 7.5b. We propose to specify a business process as a set of interacting autonomous agents [13]. An agent knows how to meet its stated goals by making use of its capabilities and available resources. It may use

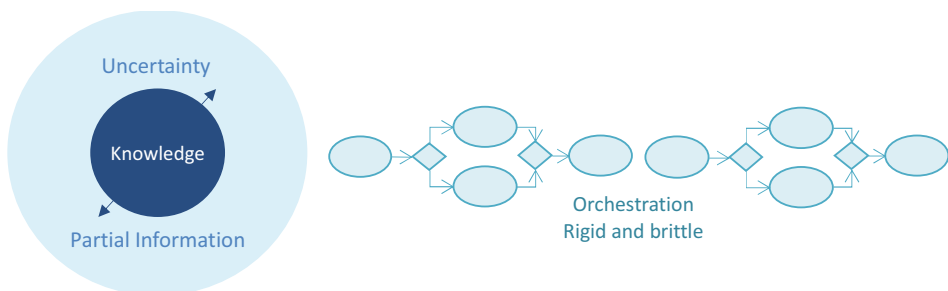


Fig. 7.6 Business process engineering – current state

other agents in this process. We propose to use Reinforcement Learning to arrive at the right purposive orchestration that's mindful of the current state of environment, goals of agents and the system goal [14]. The learner component helps keep agent specifications current through the analysis of system execution traces as well as extraction of relevant knowledge from external sources.

Bringing It All Together

An enterprise is typically viewed along three planes, namely, Strategy, Process and System planes [15]. The strategy plane concerns with the definition of the desired goals and devising suitable strategies for achieving these goals. The process plane concerns with the definition of roles and responsibilities and workflows to effectively operationalize the strategies. The system plane concerns with the definition of software systems for the efficient automation of the operational processes. Decisions need to be made at each plane such that they are consistent with each other and meet the overall goals. In essence, decisions arrived at strategy plane set goals to be achieved for process plane, and decisions arrived at process plane set goals for system plane. Partial information, high dynamics and uncertainty make it highly challenging. Current decision-making practice that relies heavily on human experts is turning out to be ineffective.

We propose an architecture, associated method and automation aids to support decision-making as well as effecting these decisions as shown in Fig. 7.7. We borrow proven ideas from established fields, namely, Digital Twin(s) from modelling and simulation, Reinforcement Learning from artificial intelligence and model reference adaptive control from

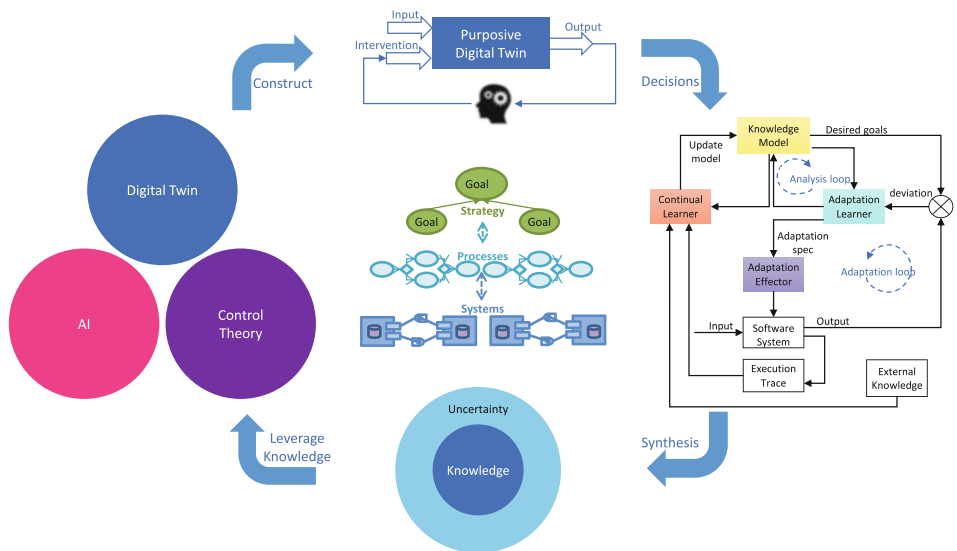


Fig. 7.7 Bringing it all together

control theory. To support decision-making at strategy plane, we construct a purposive digital twin for the enterprise as a system and its environment. We support automation-aided method that leverages domain knowledge and enterprise data to construct purposive Digital Twin(s). Thus, in essence, the digital twin captures behaviours exhibited so far as well as art of the possible. We use the purposive digital twin as a risk-free “in silico” business experimentation aid to arrive at right decisions to achieve enterprise goals. These decisions in turn set goals for process and system planes. We propose a learning-aided adaptation architecture for software systems and business processes that facilitates easy adaptation to meet these goals. The business processes and software systems are instrumented to generate execution traces. We provide AI-based machinery to analyse and learn from these traces to update the knowledge that may lead to updated Digital Twin (s), thus completing the cycle [16].

Illustrative Example

We discuss how the wellness platform enables the wellness provider to help various stakeholders of the wellness ecosystem adapt to changing needs as well as goals [17]. We illustrate this capability with a few scenarios involving two stakeholders, namely, individual and gym.

The gym classifies potential clientele into a set of buckets. For illustration sake, we consider two buckets, namely, *senior citizen* and *young professional*. For each bucket, the gym assumes a certain representative profile in terms of characteristics such as age, level of activity, overall health and propensity towards different kinds of exercises such as cardio, weight training, yoga, etc. Based on this information, the gym arrives at the fitness aspiration for each bucket and a fitness plan that’ll help a prototypical member of the bucket achieve the aspired fitness level. Based on these parameters and projected volume of the clientele, the gym arrives at the numbers and kinds of exercise equipment that’s sufficient to support the fitness plans of the two buckets. The gym would like to maximize return on investment as well as customer satisfaction. In case these goals are not met, the gym would like to revisit the fitness plans being offered as well as the fitness equipment portfolio.

While an individual may broadly fit into one of the two buckets, his fitness aspirations as well as propensity towards different exercises may vary significantly from the prototypical profile of the bucket. Therefore, the individual would like the bucket-level fitness plan to be suitably adapted to his fitness aspiration and propensity.

The wellness provider offers:

- A digital twin-based design service to the gym for coming up with bucket-level plans wherein Digital Twin(s) of prototypical individual of each bucket and the gym are created and fitness plan for each bucket is derived through the simulation of prototypical individuals. Coupled with the projected volume of customers for each bucket, the gym can then arrive at the required fitness equipment portfolio.

- A digital twin-based adaptation service to the gym for coming up with customized fitness plans for individuals wherein digital twin for each individual is created to simulate the bucket-level fitness plan and individual's response to the plan based on which the plan is suitably altered to maximize fitness level attainable while honouring individual's propensity towards different kinds of exercises. Coupled with volume of clientele, the gym can then check if the existing fitness equipment portfolio is sufficient and if not what's the required augmentation.
- A digital twin-based adaptation service to an individual for coming up with customized fitness plan.
- Propensities of individual will change over time, thus necessitating adaptation to fitness plan. This adaptation may necessitate adaptation at gym end in the form of fitness equipment portfolio.

For instance, consider the two buckets senior citizen and young professional having the characteristics shown in Table 7.1.

Based on the bucket characteristics, the gym arrives at characteristics of prototypical member of the bucket. Using these characteristics, the gym arrives at exercise preference ratio for the prototypical member. The wellness provider then helps the gym to come up with a fitness plan for the bucket consisting of weekly schedule, namely, < Day, Exercise, Duration >. Calorie burn can then be computed from this fitness plan, thus defining the targeted fitness level. Here, the fitness level is principally a function of calories burnt.

Let us consider two individuals from young professional bucket, namely, YP_1 and YP_2, having characteristics as shown in Table 7.1. If they followed bucket-specific fitness plan, their calorie burn would have come to 1260 kcal/week and 1470 kcal/week, respectively. However, since their exercise preferences deviate from those of the bucket, the wellness provider helps the gym to arrive at customized fitness plans for YP_1 and YP_2 honouring their preferences. With the customized fitness plans, YP_1 is able to burn

Table 7.1 Profiles of buckets and individuals

Characteristic	Senior Citizen	Young Professional	YP_1	YP_2
Age	60 – 75 years	25 – 40 years	27	35
Lifestyle	Sedentary	Active	Active	Sedentary
Height	5'3" – 5'10"	5'6" – 6'2"	5'9"	5'6"
Weight	60 – 80 Kg	50 – 75 Kg	60 Kg	70 Kg
Comorbidities	Yes	No	No	Yes
Food habit	Healthy	Unhealthy	Healthy	Unhealthy
Calory burn target	15 * Weight Kcal per week	21 * Weight Kcal per week	Bucket specific plan – 1260 Customized plan – 1810	Bucket specific plan – 1470 Customized plan – 1130
Capacity to workout	Low – Medium	Medium – High	High	Medium
Exercise preference	Cardio : Upper body : Lower body :: 50 : 20 : 30	Cardio : Upper body : Lower body :: 30 : 35: 35	Cardio : Upper body : Lower body :: 25 : 35: 40	Cardio : Upper body : Lower body :: 60 : 20: 20

1810 kcal/week which is greater than the bucket fitness level. However, YP_2 is only able to burn 1130 kcal/week which falls below the bucket fitness level, thus necessitating further tweaking of fitness plan. This would mean striking a trade-off between sticking to exercise preference and burning more calories.

On similar lines, the wellness provider can offer design and adaptation services to individual and hospital, individual and insurance company and individual and leisure provider.

Similar value integrators and associated platforms can be imagined for a variety of other domains such as travel tourism and hospitality, building and construction, banking, etc. For example, we can imagine the builder acting as a value integrator for property buyers, property users and property financiers wherein the builder offers pre-leased property, thus making it quite attractive to buyers and financiers.

Essentially, the approach and associated platform can benefit any domain characterized by multiple inter-dependent stakeholders, partial information, dynamism and changing environment as well as objectives, thus requiring a solution that keeps the ecosystem viable in the least and moving towards optimality.

References

1. Hoogervorst, J. A. P. (2009). *Enterprise governance and enterprise engineering*. Springer Science & Business Media.
2. Peltoniemi, M., & Vuori, E. (2004). Business ecosystem as the new approach to complex adaptive business environments. *Proceedings of eBusiness Research Forum*, 2(22), 267–281.
3. Visnjic, I., Neely, A., Cennamo, C., & Visnjic, N. (2016). Governing the city: Unleashing value from the business ecosystem. *California Management Review*, 59(1), 109–140.
4. Briscoe, G. (2010). Complex adaptive digital ecosystems. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems* (pp. 39–46).
5. Barat, S., Kulkarni, V., & Barn, B. (2018). Towards improved organisational decision-making – a method and tool-chain. *Enterprise Modelling and Information Systems Architectures–International Journal of Conceptual Modeling*, 13(2018), 6.
6. Kulkarni, V., Barat, S., & Clark, T. (2019). Towards adaptive enterprises using digital twins. In *2019 winter simulation conference (WSC)* (pp. 60–74). IEEE.
7. Barat, S., Kulkarni, V., Clark, T., & Barn, B. (2022). Digital twin as risk-free experimentation aid for techno-socio-economic systems. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems* (pp. 66–75).
8. Barat, S., Khadilkar, H., Meisheri, H., Kulkarni, V., Baniwal, V., Kumar, P., & Gajrani, M. (2019). Actor based simulation for closed loop control of supply chain using reinforcement learning. In *Proceedings of the 18th international conference on autonomous agents and multiagent systems* (pp. 1802–1804).
9. Kholkar, D., Roychoudhury, S., Kulkarni, V., & Reddy, S. (2022). Learning to adapt–software engineering for uncertainty. In *15th Innovations in Software Engineering Conference* (pp. 1–5).
10. Clark, T., Barn, B., Kulkarni, V., & Barat, S.. (2017). *Querying histories of organisation simulations*.
11. Mehandjiev, N., & Grefen, P. (Eds.). (2010). *Dynamic business process formation for instant virtual enterprises* (Vol. 39). Springer.

12. Chang, J. F. (2016). *Business process management systems: Strategy and implementation*. Auerbach Publications.
13. Jennings, N. R., Faratin, P., Johnson, M. J., Norman, M. J., O'Brien, P., & Wiegand, M. E. (1996). Agent-based business process management. *International Journal of Cooperative Information Systems*, 5(02n03), 105–130.
14. Metzger, A., Kley, T., & Palm, A. (2020). Triggering proactive business process adaptations via online reinforcement learning. In *International Conference on Business Process Management* (pp. 273–290). Springer.
15. Kulkarni, V. (2022). Toward AI-native enterprise. In *Enterprise Engineering Working Conference* (pp. 10–17). Springer.
16. Clark, T., Kulkarni, V., Barn, B., & Barat, S. (2017). The construction and interrogation of actor based simulation histories. *CEUR Workshop Proceedings*, 1979, 334–347.
17. Roychoudhury, S., Selukar, M., Kholkar, D., Suraj, Choudhary, N., Kulkarni, V., & Reddy, S. (2022). *Learning-aided adaptation – A case study from wellness ecosystem*. The Enterprise Computing Conference, EDOC Forum, Bozen-Bolzano, Italy, 2022. <https://edocconference.org/2022/pre-proceedings/edoc-forum-roychoudhury.pdf>