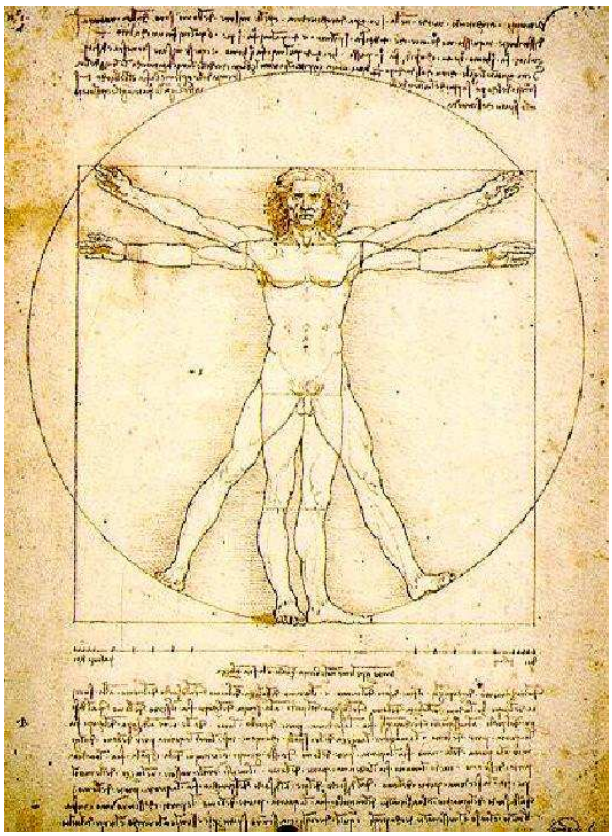
# Foundations of Work-Systems Modeling

H.A. (Erik) Proper

Version of: 30-01-06



The Art & Science of Work Systems Engineering

# Foundations of Work-Systems Modeling

H.A. (Erik) Proper

This textbook was produced by means of LaTeX.

# Contents

**The DAVINCI Series** **9**

**Course Description** **13**

**Preface** **15**

**1 Introduction** **17**
1.1 Organizations . . . 17
1.2 Information systems . . . 18
1.3 Work-systems . . . 19
1.4 A fundamental view on work-systems modeling . . . 19
1.5 Formal approach . . . 20
Questions . . . 20
Bibliography . . . 20

**2 Work Systems** **23**
2.1 Exploring systems . . . 23
2.2 Observing systems . . . 24
2.2.1 Subjectivity . . . 24
2.2.2 Observing the universe . . . 26
2.2.3 Conceptions . . . 28
2.2.4 Model . . . 35
2.2.5 System . . . 35
2.3 Studying systems . . . 37
2.3.1 Sub-systems . . . 37
2.3.2 Describing systems . . . 39
2.3.3 Classes of systems . . . 40
2.4 Dealing with evolution of conceptions . . . 42
2.5 Conclusion . . . 45
Questions . . . 45
Bibliography . . . 47

**3 Basic Object-Role Modeling** **49**
3.1 Natural language grounding of modeling . . . 49
3.2 The logbook heuristic . . . 49
3.3 Verbalizing conceptions . . . 50
3.4 Elementary facts . . . 51
3.5 From instances to types . . . 53
3.6 Subtyping . . . 57
3.7 Overlap of populations . . . 59
Questions . . . 59
Bibliography . . . 61

**4 Object-Role Calculus** **63**
4.1 Introduction . . . 63
4.2 Computational domain . . . 64
4.3 Logic layer . . . 65
4.4 Path expression layer . . . 66
4.4.1 Atomic path expressions . . . 66
4.4.2 Composing paths . . . 67
4.4.3 Evolution and path expressions . . . 68
4.4.4 Path expressions as logic . . . 69
4.5 Graphical constraints . . . 69
4.5.1 Mandatory roles . . . 69
4.5.2 Uniqueness . . . 69
4.5.3 Subsets . . . 70
4.5.4 Temporal ordering . . . 70
4.6 Information-descriptor layer . . . 73
4.6.1 Naming of types . . . 73
4.6.2 Basic information descriptors . . . 74
4.6.3 Complex information descriptors . . . 74
4.6.4 Domain rules . . . 75
Bibliography . . . 75

**5 Advanced Object-Role Modeling** **77**
5.1 Subtyping . . . 77
5.2 Overlap of populations . . . 79
5.3 Abstraction . . . 80
5.4 Set types . . . 89

5.5 Multi-set types . . . 89
5.6 Sequence types . . . 91
5.7 Schema types . . . 91
Questions . . . 91
Bibliography . . . 93

**6 The Act of Modelling** **95**
6.1 What to model? . . . 95
6.2 The modeling challenge . . . 95
6.2.1 Goal-bounded and communication-driven . . . 95
6.2.2 Aspects of a method . . . 96
6.2.3 The process of modeling . . . 97
6.3 Ambition levels for modeling . . . 98
6.4 Meeting the challenge . . . 98
6.4.1 Modeling a singular domain . . . 98

**7 Natural-Language Foundations of Information-Systems Modeling** **101**
7.1 Classes of roles . . . 101
7.2 Activity types . . . 104
Questions . . . 105
Bibliography . . . 105

**I Apendixes** **107**

**A Mathematical Notations** **109**
A.1 Sets . . . 109
A.2 Functions . . . 109
A.3 Relations . . . 110

**B Answers to questions** **111**
B.1 Questions from Chapter 1 . . . 111
B.2 Questions from Chapter 2 . . . 112
B.3 Questions from Chapter 3 . . . 119
B.4 Questions from Chapter 5 . . . 122
B.5 Questions from Chapter 7 . . . 123

**Bibliography** **125**

**List of Symbols** **133**

**Dictionary** **135**

**Author Index** **139**

**Subject Index** **141**

# The DAVINCI Series

Version: 30-08-05

The subtitle of the DAVINCI series of lecture notes is *The Art & Science of Work Systems Engineering*. On the one hand, this series of lecture notes takes a fundamental view (*craft*) on the field information systems engineering. At the same time, it does so with an open eye to practical experiences (the *art*) gained from information system engineering in industry.

The kinds of information systems we are interested in range from personal information appliances to enterprise-wide information processing. Even more, we regard an information system as a system that "handles" information, where "handling" should be interpreted in a broad fashion. The actors that do this "handling" can be computers, but can equally well be other "symbol wielding machines", but can also be humans. The mix of humans and computers/machines in information systems makes the field of information system engineering particularly challenging.

The concept of "information" itself is very much related to the concepts of *data*, *knowledge* and *communication*. Based on [FVV+98], we will (throughout the DAVINCI series) use the following definitions:

**Data –** Any representation in some language. Data is therefore simply a collection of symbols that may, or may not, have some meaning to some actor.

**Information –** The knowledge increment brought about when a human actor receives a message. In other words, it is the difference between the conceptions held by a human actor *after* interpreting a received message and the conceptions held beforehand.

**Knowledge –** A relatively stable, and usually mostly consistent, set of conceptions posessed by a single (possibly composed) actor.

In more popular terms: "an actor's picture of the world".

**Communication –** An exchange of messages, i.e. a sequence of mutual and alternating message transfers between at least two human actors, called communication partners, whereby these messages represent some knowledge and are expressed in languages understood by all communication partners, and whereby some amount of knowledge about the domain of communication and about the action context and the goal of the communication is made present in all communication partners.

When referring to an information system, we therefore really refer to systems that enable the communication/sharing of knowledge by means of the representation (by human actors), storage, processing, retrieval, and presentation (to human actors) of the underlying representations (data). This also implies that we will treat *information retrieval systems*, *knowledge-based systems*, *groupware systems*, etc., as special classes of information systems.

The lecture notes in the DAVINCI series have been organized around four key key processes in an information system's life-cycle:

**Definition process –** A process leading to a definition description.

Where **definition** is defined as:

> The requirements that should be met by a desired work system as well its system description including the descriptions of the system's definition, design as well as documentation for the operational system.
>
> These requirements will typically identify: *what* it should do, *how well* it should do this, and *why* it should do so.

**Design process –** A process leading to a design description.

Where **design** is defined as:

> The identification and motivation of *how* a work system will meet the requirements set out in its definition. The resulting design may (depending on the design goals) range from high-level designs to the detailed level of programming statements or specific worker tasks.

**Realization process –** The combination of a construction process and a deployment process.

Where **construction process** is defined as:

> A process aiming to realize and test a system that is regarded as a (possibly artificial) artifact that is not yet in operation.

Where **deployment process** is defined as:

> A process aiming to make a system operational, i.e. to implement the use of the system by its prospective users.

**Architecting –** The processes which tie definition, design and deployment and to the explicit and implicit needs, desires and requirements of the usage context. Issues such as: business/IT alignment, stakeholders, limiting design freedom, negotiation between stakeholders, enterprise architectures, stakeholder communication, and outsourcing, typify these processes.

**Domain modeling –** Modeling of the domains that are relevant to the information system being developed. The resulting models will typically correspond to *ontologies* of the domains. These domains can pertain to the information that will be processed by the information system, the processes in which the information system will play a role, the processing as it will occur inside the information system, etc. Understanding (and modeling) these domains is fundamental to the other activities in information system engineering.

For each these aspects, attention will be paid to relevant theories, methods and techniques to execute the tasks involved. When put together, these aspects can be related as depicted in figure 1. Note that we regard maintenance of systems as being functionality that should be designed "into" the system. If a system needs to be maintained, and in most cases one indeed wants to, then the maintenance should be designed into the workings of this system and/or its context.

The use of the name DAVINCI originates from earlier work [Pro98] done on architecture-driven information systems engineering. The work reported in [Pro98] was the result of a confrontation between industrial practice and a theoretical perspective on information systems and their evolution [Pro94a]. The result was a shared vision on the architecture-driven development of information systems by a Dutch IT consultancy firm. In this shared vision, a foundation was laid for an integrated view on information system engineering. At that stage, the name "DAVINCI" was also selected. Not as some artificial acronym, but rather to honor an inspiring artist, scientist, inventor and architect. To us he personifies a balance between art and engineering, between human and technology.

After the development of the first DAVINCI version, a more elaborate version [Pro04] was developed at the Radboud University Nijmegen in the form of lecture notes associated to a course on *Architecture & Alignment*. In this version, a more fundamental outlook on information system development was added to complement the practical orientation of the first version.

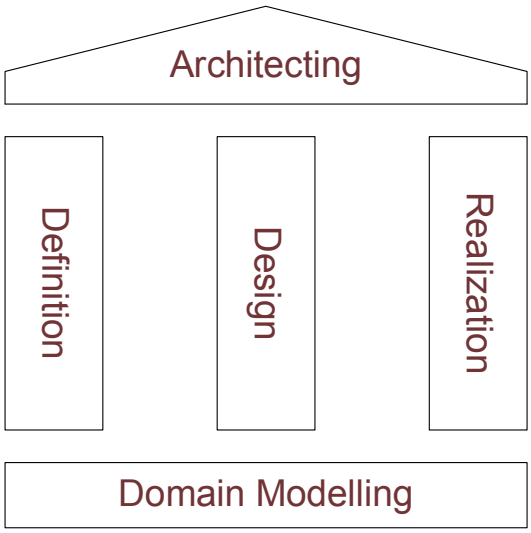



Figure 1: Aspects of Information Systems Engineering

As a third step, we have now taken on the underlying philosophy of the first two DAVINCI documents, and used this as the source of inspiration to shape an entire line of lecture notes for a number of mutually related courses on different aspects of information systems engineering. In making this step we have also been able to anchor some of the fundamental research results from the co-authors, on subjects such as information modeling [BHW91, BW92a, HW93, HPW93, PW94, BBMP95, CHP96, CP96, PW95, BFW96, HPW97, Pro97, HVH97, FW02, FW04a], information retrieval [BW90, BW91, BW92b, BB97, WBW00, SFG+00, PB99, PPY01, WBW01], (enterprise) information architecture [JLB+04] and information system engineering [Pro01, VHP04] into the core of the DAVINCI series.

# Course Description

Version:
18-06-05

## Short description

Organizations can by found anywhere. A University is an organization, a sports club is an organization, a bank is one, government departments are, etc. Organizations are everywhere. In our modern western society, most organizations use some information systems to support the activities of the organization. Large parts of these information systems are likely to be computerized.

Organizations and (computerized) information systems are examples of so-called *work-systems*. For information system engineers it is relevant to be able to model relevant aspects of the design of such systems. This may be the design of a currently existing system or the design of the future evolution/development of the current system. In this course we will discuss several examples of work-systems from organizational, information systems, biological and sociological domains.

## Learning goals

After this course, students are able to:

1. given a case-description of an work-system (such as an organization):
   (a) produce models for different aspects of this system,
   (b) understand and evaluate given models of that system,
2. argue about, and prove, properties regarding the syntax & semantics of the models,
3. reason about the link between an work-system's strategy, and the services & processes it uses to realize this strategy,
4. reason about the position of work-system modeling the context of information systems engineering.

## Topics

1. Work-systems as a generalization of organizations and information systems
2. Temporal ordering, actors, actions, actands.
3. Activity modeling.
4. Actor modeling.
5. Actand modeling.
6. Design patterns.
7. Work-system strategy.

# Preface

Version: 17-01-06

In 2006, the course "Modeling of Organizations" is taught for the third time. This third time will be the second time we will use the new lecture notes "Work Systems Modelling" from the DA VINCI series. These lecture notes, however, will be evolved further hand-in-hand with the actual process of lecturing. In the academic year 2005/2006, a second incarnation of these lecture notes will be created, where the aim is to deliver these lecture notes in three increments.

An important step that will be taken in this academic year is the integration of the ICIS Work Systems Modelling lecture notes with the NICI course on Organisational Dynamics. The first results of this integration will start to appear in the second and third trimester.

Needless to say that any feedback from either students or colleagues is more than welcome.

The priority of this initial version of this textbook is on completeness of the topics that need to be covered, rather than readability and completeness of text. Students are advised to make notes during lectures.

Special thanks go out to the students attending the "Information Intensive Organizations" course in 2004/2005. They where the guinea pigs for the new Lecture Notes, pointing out several major shortcomings. Also thanks go out to Arnoud Vermeij, for adding several questions and answers, as well as commenting on draft versions of these lecture notes.

# Chapter 1

# Introduction

Version: 17-01-06

The focus of these lecture notes is on modeling of different aspects of work systems. In doing so, we will build on top of the general modeling foundations laid in the *Domain Modeling* course.

In this chapter, we provide a brief exploration of the concept of work systems. The understanding of this concept as provided in this chapter will serve as a starting point. In the next chapters, this concept will be put in a more fundamental context. We will do so by discussing the concept of work system, from the bottom up. We will start (**??**) with a fundamental discussion of the underlying ontology of systems, modeling and work systems from which we will approach these phenomenon. We will then proceed (**??**) by discussing specific modeling techniques for the modeling of an work system's formal structures as well as their potential evolution over time. In the last part (**??**), we will consider the design of work systems from the perspective of their strategies.

Note: Not all parts of these lecture notes will be used in the "Modelleren van Organisaties" course.

## 1.1 Organizations

Organizations are an ubiquitous phenomenon in our modern day society. Most of our lives are spent in the context of organizations. We are born in *hospitals*, we receive an education from *schools* and *universities*. Later on we work for *factories*, *enterprises*, etc. In our spare-time, we visit *restaurants*, *sportclubs*, etc. These are all examples of organizations. Our modern-day lives are surrounded by a plethora of organizations.

However, the ubiquity of organizations is not something new. Also in the past organizations have always been dominantly present. In line with our working definition, organizations can be regarded as a more or less stable network of social relationships. In doing so, the concept of organizations can be said to be as old as humanity itself.

At first glance, the concept of organization comes natural to us. Organizations can be found anywhere. A University is an organization, a sports club is an organization, a bank is one, government departments are, etc. Organizations are everywhere. In general, an organization could be an enterprise, an institution, a company, a factory, etc., or it could be a functional, geographic or organizational part thereof, typically a department or a specific kind of the business. The concept of organization, however, is not limited to such "formal" and "explicit" organizations. Ad-hoc groupings of people, such as a group of friends regularly having a beer after office hours in order to relax after a day of work, may also classify as (ad-hoc) organizations. In this textbook, we will use the following definition of organization:

**Organization –** A group of actors with a purpose, who:

- interact with each other,
- form a network of roles,
- make use of (the services of) other actors.

An organization in itself is an actor as well, and may as such participate in yet another organizations.

Note the term *purposely*. An organization is formed by a group of actors. They will typically do so in order to achieve some shared/private goals.

## 1.2 Information systems

In our modern western society, most organizations use some form of information systems to support the activities of the organization. With "information system" we (informally) refer to information processing activities that may be performed by computerized as well as non-computerized actors. Without these information systems, most organizations would no longer be able to exist. Even more, some organizations are actually large information systems themselves. For example, banks, insurance companies, taxation offices, are really 'just' very large information systems comprising human, physical (money, bankcards, etc.) and computerized actors.

The concept of information system can roughly be defined as that aspect of an organization that provides, uses and distributes information. An information system *may* contain computerized sub-systems to automate certain elements. Some information system may not even be computerized at all. A filing cabinet used to store and retrieve several dossiers is, in essence, an information system. The kind of information systems we are interested in, however, are indeed presumed to have some computerized core parts.

What we may perceive to be an information system, may vary highly in terms of their scope. Some examples would be:

- Personal information appliances, such as electronic agenda's, telephone registries in mobile phones, etc.
- Specific information processing applications.
- Enterprise wide information processing.
- Value-chain wide information processing.

Some concrete examples are:

- An insurance-policy administration is an information system
- A bank is (primarily) an information system
- Clients are actors in that information system
- The taxation department is an information system
- The PDA you use as an agenda
- The phone number collection in your mobile phone

In practice, the concept of "*information system*" is used quite differently by different groups of people. It seems (see e.g. [FVV+98]) to be interpreted in at least three different ways:

- As a technical system, implemented with computer and telecommunications technology.
- As a social system, such as an organization, in connection with its information processing needs.
- As a conceptual system (i.e. an abstraction of either of the above).

## 1.3 Work-systems

In the field of information systems, this has also lead to a generalisation of the notion of organization and information systems to *work systems*, focusing on the essential common properties such as actors, resources, etc. In [Alt99, Alt02] Alter defines a work system as:

> *A work system is a system in which human participants and/or machines perform business processes using information, technologies, and other resources to produce products and/or services for internal or external customers.*

Typical business organizations contain work systems that procure materials from suppliers, produce products, deliver products to customers, find customers, create financial reports, hire employees, coordinate work across departments, and perform many other functions.

We will actually generalize the definition of work-system even further to:

**Work system –** An open active system in which actors perform processes using information, technologies, and other resources to produce products and/or services for internal or external actors.

where we have purposely generalized "human participants and/or machines" as used in Alter's definition of work systems to the notion of actors in order to abstract from the fact wether these actors are of a biological, mechanical, chemical, electronical, or whichever, means. Actors are presumed to perform *activities* (work!) in order to achieve some *purpose*. We have furthermore replaced any explicit reference to business like terminology as work-systems is intended to be a more general notion that just organisations, enterprises, etc.

As discussed in [Alt99, Alt02], the work-system concept can be used as a common denominator for many types of systems. Enterprises, value chains, organizations, operational information systems, projects, supply chains, and ecommerce web sites can all be viewed as special cases of work systems. Organisations are work systems. An information system is a work system whose work practices are devoted to processing information. A project is a work system designed to produce a product and then go out of existence. A supply chain is an interorganizational work system devoted to procuring materials and other inputs required to produce a firm's products. An ecommerce web set can be viewed as a work system in which a buyer uses a seller's web site to obtain product information and perform purchase transactions. The relationship between work systems in general and the special cases implies that the same basic concepts apply to all of the special cases, which also have their own specialized vocabulary. In turn, this implies that much of the body of knowledge for the current information systems discipline can be organized around a work system core.

Specific information systems exist to support (other) work systems. Many different degrees of overlap are possible between an information system and a work system that it supports. For example, an information system might provide information for a non-overlapping work system, as happens when a commercial marketing survey provides information to a firm's marketing managers In other cases, an information system may be an integral part of a work system, as happens in highly automated manufacturing and in ecommerce web sites. In these situations, participants in the work system are also participants in the information system, the work system cannot operate properly without the information system, and the information system has little significance outside of the work system.

## 1.4 A fundamental view on work-systems modeling

Modeling is at the very heart of the field of information systems engineering as well as organizational engineering. Any course on the modeling of work systems should therefore also provide

a fundamental understanding of modeling, in particular the modeling of work systems. As we will see in the next chapter, when two people model the same domain, they are likely to produce quite different models. Even when they use the same information (informants, documents, etc.) to produce the models, the models are still likely to differ considerably. This also means that if two people communicate about the same work system, they are likely to do so with different models of this work system in mind. *Why do these differences occur What are the origins of these different models? What happens when people produce models?* Questions that beg for a fundamental answer. These lecture notes try to provide some of the answers.

## 1.5 Formal approach

In developing our understanding of modeling organizations, we will discuss several modeling languages for different aspects of organizations. When discussing these modeling languages, we will also discuss their syntax and semantics from a formal (mathematical) perspective. In [HW92, Hof93, HP98] three major reasons for a formal approach to the syntax and semantics of modeling techniques are given.

Even though in literature it has often been emphasized that modeling languages should have a rigorous formal basis (see e.g. [Coh89, TP91, Spi88, Jon86, HL89], somehow this need for formality has not been generally acknowledged in the field of information systems engineering and organization engineering. This has contributed greatly to the appearance of the "*Methodology Jungle*", a term introduced in [Avi95]. In [Bub86] it is estimated that during the past years, hundreds if not thousands of information system development methods have been introduced. Most organizations and research groups have defined their own methods. The techniques advocated in these methods usually do not have a formal foundation. In some cases their syntax is defined, but attention is hardly ever paid to their formal semantics. The discussion of numerous examples, mostly with the use of pictures, is a popular style for the "definition" of new concepts and their behavior. This has led to *fuzzy* and *artificial* concepts in information systems development methods (see also [Bub86]).

Version: 17-01-06

## Questions

1. What is an organization? Give some examples of groupings of people that are not an organization.
2. Produce a model of the hierarchical structure of a university (faculties, departments, schools, etc). Why is the model organized this way?
3. Produce a model of the educational process of attending a course at a university. What are the contributions of the different elements in this process.
4. Describe why it is important to realize that organizations can be part of yet other organizations. Use the term 'level of abstraction' in your answer.
5. If two people were to produce a model of the same organization. Would you expect them to produce the same model? If not, why do you think these models would differ?

## Bibliography

[Alt99] S. Alter. A general, yet useful theory of information systems. *Communications of the Association for Information Systems*, 1(13), 1999.
`http://cais.isworld.org/articles/1-13/default.asp`

[Alt02] S. Alter. The work system method for understanding information systems and information system research. *Communications of the Association for Information Systems*, 9(9):90–104, 2002.
`http://cais.isworld.org/articles/default.asp?vol=9&art=6`

[Avi95] D.E. Avison. *Information Systems Development: Methodologies, Techniques and Tools*. McGraw–Hill, New York, New York, USA, 2nd edition, 1995. ISBN 0077092333

[Bub86] J.A. Bubenko. Information System Methodologies – A Research View. In T.W. Olle, H.G. Sol, and A.A. Verrijn–Stuart, editors, *Information Systems Design Methodologies: Improving the Practice, Amsterdam, The Netherlands, EU*, pages 289–318. North–Holland/IFIP WG8.1, Amsterdam, The Netherlands, EU, 1986.

[Coh89] B. Cohen. Justification of Formal Methods for System Specification. *Software Engineering Journal*, 4(1):26–35, January 1989.

[FVV+98] E.D. Falkenberg, A.A. Verrijn–Stuart, K. Voss, W. Hesse, P. Lindgreen, B.E. Nilsson, J.L.H. Oei, C. Rolland, and R.K. and Stamper, editors. *A Framework of Information Systems Concepts*. IFIP WG 8.1 Task Group FRISCO, IFIP, Laxenburg, Austria, EU, 1998. ISBN 3901882014

[HL89] I. van Horenbeek and J. Lewi. *Algebraic specifications in software engineering: an introduction*. Springer, Berlin, Germany, EU, 1989.

[Hof93] A.H.M. ter Hofstede. *Information Modelling in Data Intensive Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1993.

[HP98] A.H.M. ter Hofstede and H.A. (Erik) Proper. How to Formalize It? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 40(10):519–540, October 1998.

[HW92] A.H.M. ter Hofstede and Th.P. van der Weide. Formalisation of techniques: chopping down the methodology jungle. *Information and Software Technology*, 34(1):57–65, January 1992.

[Jon86] C.B. Jones. *Systematic Software Development using VDM*. Prentice–Hall, Englewood Cliffs, New Jersey, USA, 1986.

[Spi88] J.M. Spivey. *Understanding Z: A Specification Language and its Formal Semantics*. Cambridge University Press, Cambridge, United Kingdom, EU, 1988.

[TP91] T.H. Tse and L. Pong. An Examination of Requirements Specification Languages. *The Computer Journal*, 34(2):143–152, April 1991.

# Chapter 2

# Work Systems

Version: 30-01-06

[1]In this chapter, we discuss our fundamental view on work systems, organizations and information systems. We will provide a definition of terms, which is based on a system theoretic [Ber01] foundation.

## 2.1 Exploring systems

Even though the notion of system is, in an IT context, often equated to 'software system', the original sense of the word is much broader. The notion of system[2] is also not uniquely defined in the literature, but typically, it can be found explained as: "*A collection of interrelated parts characterized by a boundary with respect to its environment*" [Iiv83] or just as: "*A set of objects with a set of links*" [Lan71]. In general, humans refer to all sorts of things as 'systems'. The broadness of our understanding of the concept of 'system' comes, for example, to the fore in the definition as found in [Mer03]:

> A regularly interacting or interdependent group of items forming a unified whole, as:
>
> 1. a group of interacting bodies under the influence of related forces,
> 2. an assemblage of substances that is in or tends to equilibrium,
> 3. a group of body organs that together perform one or more vital functions,
> 4. the body considered as a functional unit,
> 5. a group of related natural objects or forces,
> 6. a group of devices or artificial objects or an organization forming a network especially for distributing something or serving a common purpose,
> 7. a major division of rocks usually larger than a series and including all formed during a period or era,
> 8. a form of social, economic, or political organization or practice.

The IEEE Recommended Practice for Architectural Description of Software-Intensive Systems [IEE00] provides a functionality-oriented perspective on systems:

> A collection of components organized to accomplish a specific function or a set of functions.

[1]Parts of this chapter are based on work the author was doing with A.A. (Xander) Verrijn-Stuart on a revised edition on the FRISCO report [FVV+98]. As Xander passed away unexpectedly, the revised edition was never finished.

[2]The term 'System' is derived from the Greek phrase 'Syn histanai' ($\sigma\upsilon\nu\ \acute{\iota}\sigma\tau\eta\nu\alpha\iota$): 'to put together'.

In practice, most people intuitively agree on such simple definitions of systems. Apparently these definitions are broad enough to cover the meaning of usual linguistic constructs where 'system' is used. But system is a much more difficult concept. If we look at what in practice are considered systems, and if we really think about it, it becomes obvious that some very important aspects of the system concepts are missing in the traditional definitions. In [FVV+98] some examples are given of what we would, and would not, observe to be systems in our daily life:

**Example 2.1.1**

One can regard an organization or a bicycle as systems. Also a Hitchcock film recorded on a video cassette, which is inserted in a video cassette player, which again is connected to a TV-set, could easily be interpreted as a system. Nothing is unusual with such system views, and they are well covered by the definitions. But if you buy some eggs from a farmer and use two of them for breakfast, then the domain of obviously interrelated phenomena: You, the farmer, the farmers hen that laid the eggs, the frying pan you used to prepare the eggs, and the two eggs now in your stomach (and thereby in some transformed form a part of yourself) – this domain might probably not be regarded as a system, because it might be difficult to see a purpose for that. But it fits the definitions.

Or consider a single raindrop in an April shower: It consists of a vast number of water molecules, kept together by surface tension and constantly moving around among each other in a complicated manner controlled by a set of (thermo-) dynamic forces. Again according to the simple definitions above, the drop qualifies as a system. But that is strange, because when you on your way back from the farmer, happen to get soaked in the shower, you might feel it is caused by raindrops – not by systems.

On the other hand, a meteorologist studying possible weather situations that could cause rain, may see a purpose in regarding a raindrop as a system in interaction with the surrounding atmosphere, but in most other situations a raindrop is just a raindrop.

However, when looking at what is regarded as a system in practice, it becomes apparent that some very important aspects of the system concept are missing from the traditional definitions.

In [Rop99] it is argued that, strictly speaking, there exist three different interpretations of systems:

**Structural –** The structural interpretation is known best. According to this interpretation, a system includes a set of elements and a set of links between these elements. This interpretation complies with the ancient definition of the *holon* by Aristotle.

**Functional –** The functional interpretation views a system as an entity, sometimes called black box, which transforms inputs into outputs. Depending on specific internal states; the kind of transformation is called a function (in the descriptive meaning of the word).

**Hierarchical –** The structural interpretation turns into the hierarchical interpretation, if (some of) the constituting elements are regarded as subsystems. Concluding by analogy, the original system may be considered as a subsystem of a more extensive supersystem.

## 2.2 Observing systems

The aim of this section is to more precisely define the concept of system, also giving more foundation to our earlier definitions of work system, organizational system, and information system.

### 2.2.1 Subjectivity

When two people discuss a system, do they really mean the same system? One serious cause for confusion in our professional domain is, that people, usually, think about a system as something

that can be objectively determined, for example by a specification of its parts and their relationships, as the above quoted definitions may indicate. But even then, the problem remains. Are both people indeed discussing the same system?

As an example take the simple domain of a car and its driver in the traffic of a city. One person may see it as a useful transport system in action, which is able to move large objects from one location to another in a convenient way. The driver alone cannot, nor can the car, but in combination they can. However, a policeman on his job will regard the same domain differently – as a controllable system which behavior can be directed by road regulations, traffic lights, arm signals and by certain traffic rules. Again, an environmental activist would probably regard the car as a dangerous polluting system, which is a potential cause of injury or death to persons in the traffic.

Here we have three views of the same domain, but with quite different sets of properties. All three persons could in fact be the same viewer of the same system, e.g. a transport conscious public servant caring about the conditions for people in the city, who just conceives different properties by regarding the same system from different points of view.

Let us elaborate this car example a little further in order to illustrate the difficulties we face when we regard something as a system. Consider for example the question about which parts and which activities are involved in the possible system view: Are the driver and the car two interacting sub-systems – one with the property of being able to observe the traffic and to control the car, and the other with the property of being able to transform chemical energy into movement in a controlled manner. Or is the car to be regarded as a single system with the driver, motor, gear, and steering devices as sub-systems each with their own properties? Is the motor the active part and the chassis a passive component, or is it the other way around – the car as a device transporting among other things the motor. Quite another view – but still one from the same domain – could be to regard the car as a moving cage of Faraday protecting the driver from certain kinds of dangerous electrical fields. There are many possible system views, and still the domain is extremely simple compared with the organizational domains usually considered as systems.

If we regard a business enterprise, an institution or any other kind of organization as a system – an organizational system – we have a domain which is much more complicated than a car and driver. Furthermore, the number of possible views of an organization is most often enormous.

Key to understanding the system concept, and ultimately organizations is therefore to realize that a system is a subjective phenomenon. In other words, it is not an absolute or objective thing. Systems are not a priori given. As Checkland [Che81] expresses it, there must be a describer/observer who conceives or thinks about a part of the world as a system. In other words, it is important, that there is a *viewer* who can see a purpose in regarding some '*set of elements*' as a system.

Viewers may also be regarded at an aggregated level. For example, a *single* business manager, observing an organization, is indeed a viewer, but the *collective* business management can be seen as a viewer of the organization as well.

The purpose in regarding some *set of elements* as a system should be expressed in terms of at least one meaningful links between the *set of elements* and its *environment*. Such a link is called a *systemic property*. It is a property the viewer associates with the *set of elements* they experience as a system. One viewer may regard the *set of elements* as a system having one set of systemic properties, while another viewer may see other systemic properties concerning the same set of elements.

Most often, the systemic properties of a system cannot be attributed exclusively to any of its constituent components. For example, none of the constituent parts of a train has the exclusive 'train property'. A separate carriage is not a train. A locomotive on its own is (from the perspective of a passenger) also not a train. Together, however, the parts do have the 'train property'. In other words, the whole is more than just the collection of its parts. The farmer-you-frying-pan-eggs-hen situation as discussed in the above example, is a situation which may not constitute a 'whole'

with any sensible systemic property. In which case we will not consider it to be a system. In the case of the train, we have an interesting situation if the train consists of two connected train-sets. In this case, each of the individual train-sets still has the 'train property'.

In order for us to gain a fundamental understanding of systems, and ultimately the kind of systems we refer to as organizations, we first need to introduce some core concepts, most of which are based on the ones found in [FVV+98].

### 2.2.2 Observing the universe

Let us start by considering what happens if some viewer observes 'the universe'. It is our assumption, based on the work of C.S. Peirce [Pei69a, Pei69b, Pei69c, Pei69d], that *viewers* perceive a *universe*, leading to a *perception* of this universe and then produce a *conception* of that part they deem relevant. Peirce argues that both the perception and conception of a viewer are strongly influenced by their interest in the observed universe. This leads to the following (necessarily cyclic, yet irreflexive) set of definitions:

**Universe –** The 'world' under consideration.

**Viewer –** An actor perceiving and conceiving (part of) a domain.

**Perception –** That what results, in the mind of a viewer, when they observe a domain with their senses, and forms a specific pattern of visual, auditory or other sensations in their minds.

**Conception –** That what results, in the mind of a viewer, when they interpret a perception of a domain.

In general, people tend to think of the universe as consisting of elements. In [FVV+98] this approach is indeed taken. In our view, presuming that the universe consists of a set of elements constitutes a subjective choice, which essentially depends on the viewer observing the universe. Nevertheless, taking this assumption has proven to be a sensible assumption, in particular in the context of systems. However, we do *not* presume the *universe* itself to consist of elements, but *rather* the *conception* of a universe. This makes the identification of elements relative to a viewer.

The conceptions harbored by a viewer are impossible to communicate and discuss with other viewers unless they are articulated somehow. In other words, a conception needs to be *described* somehow in terms of a description:[3]

**Description –** The result of a viewer denoting a conception, using some language to express themselves.

The resulting situation is illustrated in Figure 2.1. Descriptions may be:

- Formal or informal
- Complete or incomplete
- More refined/less refined

A system description may have a formal semantics, in mathematical terms, and will also have an intensional (pragmatics!) semantics. Formal semantics is needed when approaching design. The latter is likely to be of more use when communicating with different stakeholders.

The underlying relationships between viewer, universe, conception and description can be expressed in terms of the so-called FRISCO tetrahedron [FVV+98], as depicted in Figure 2.2[4].

[3] In [FVV+98] the term *representation* is used rather than the term *description* as it is used here. We have chosen to favor the term *description*, as it is the term of choice of [IEE00].

[4] The original FRISCO tetrahedron uses 'domain' where we use 'universe'. This difference is due to the above, more refined, discussion on the subjectivity of viewing the universe as a set of elements.

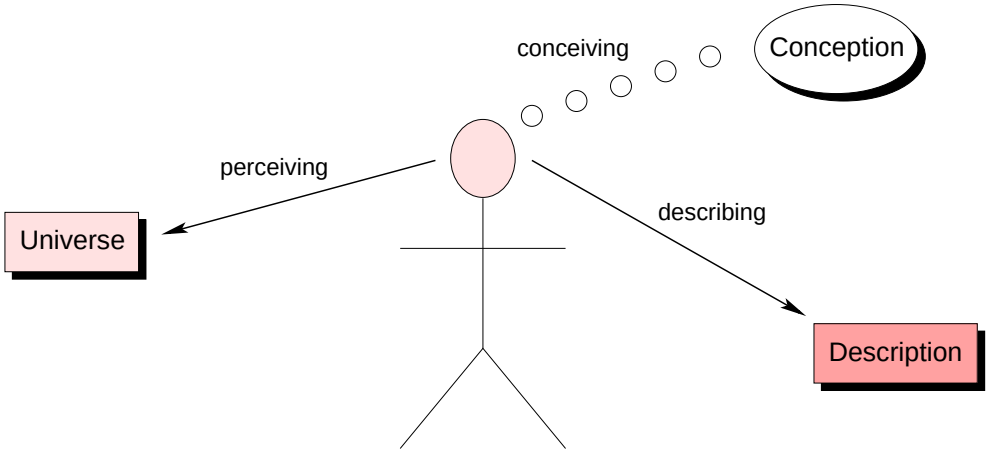



Figure 2.1: A viewer, having a conception of the universe, and describing this in terms of a description.

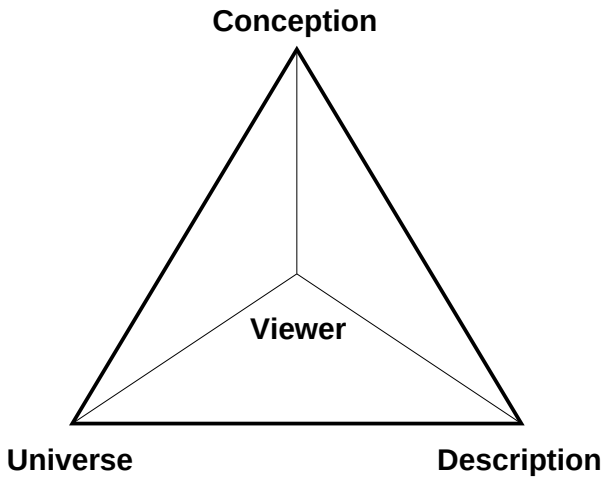



Figure 2.2: The (revised) FRISCO tetrahedron.

### 2.2.3 Conceptions

To express the above discussions more formally, let $\mathcal{UN}$[5] be the set of universes that may be observed, and let $\mathcal{VW}$[6] be the set of all possible viewers. Let furthermore, $\mathcal{EL}$ be the set of elements that may be part of conceptions. These basic sets should be disjoint:

**[S1]** $\mathcal{VW}, \mathcal{UN}$ and $\mathcal{EL}$ are mutually disjoint.

Let $\_ \models^{c} \_ \subseteq \mathcal{UN} \times \mathcal{VW} \times \wp(\mathcal{EL})$ be the links expressing which conception is held by which viewer. The fact that a viewer $v \in \mathcal{VW}$ harbors a conception $C \subseteq \wp(\mathcal{EL})$ for universe $U \in \mathcal{UN}$ can be expressed as $U \models^{c}_{v} C$. This situation is depicted more graphically in Figure 2.3. A viewer, when observing a domain, draws a picture of the observed universe (their conception). In painting this picture of the world, they will use certain 'constructs'. At the moment the only constructs we presume to exist are *elements*. Note: the fact that viewers can change their conception over time is ignored for the moment.

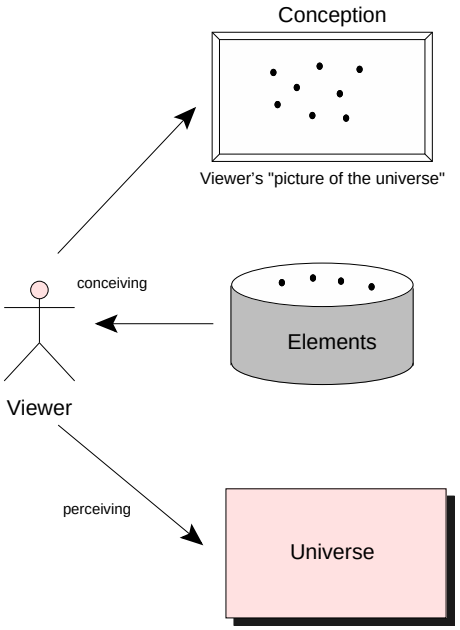



Figure 2.3: Painting a picture of the universe

#### Domains and their environments

A viewer may zoom in on a particular part of the universe they observe, or to state it more precisely, they may zoom in on a particular part of their conception of the universe.

**Domain –** Any 'part' or 'aspect' of the universe a viewer may have an interest in.

When reasoning about systems, which we will regard as a particular class of domains, it is commonplace to also identify their environments [Ber01]. Even more, the very definition of a system depends on our ability to distinguish it from its environment. This is illustrated in Figure 2.4.

[5]The reader is advised that appendix A provides an overview of the mathematical notations/conventions used in this textbook.

[6]One should actually regard this set as the set of *states* a viewer may have. A viewer may for example have, in differing states, different interests with which they conceive the universe. The elements from $\mathcal{VW}$ should really be regarded as the states of these viewers.

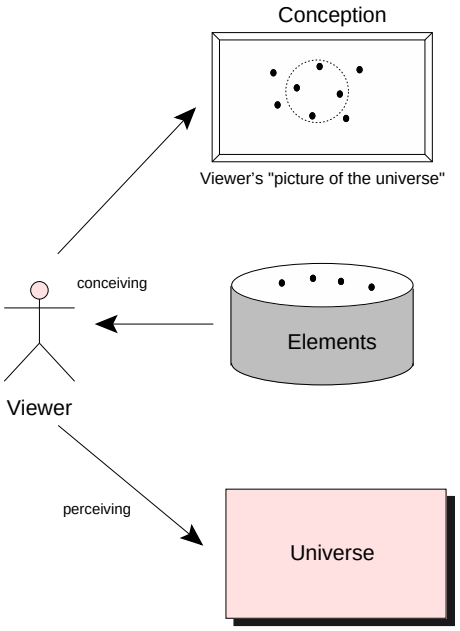



Figure 2.4: Identifying a domain and its environment

To be able to define the environment of a domain in general and a system in particular, however, we must first be able to define the direct environment of a domain. Formally, a domain $D$ can be regarded as a subset of a conception $C$, in other words a sub-conception $D \subseteq C$. If $U \models^{\mathsf{c}}_{v} C$ and $D \subset C$ is some domain within $C$, then the environment of $D$ will not generally be $C - D$ as a whole, but rather a subset $E \subseteq C - D$. For $E$ to be a sensible environment of domain $D$, the elements in $E$ must have some links to the elements in $D$. In order to more precisely define the notion of environment, we should therefore first refine our notion of elements of a conceptions. There are really two types of elements: concepts and links connecting the concepts. We will define these notions as follows:

**Element –** The elementary parts of a viewer's conception.

**Concept –** Any element from a conception that is not a link.

**Link –** Any element from a conception that relates two concepts.

The distinction between a link and an concept for the elements of a given conception, may not always be that clear, as the distinction is rather *subjective*. It all depends, to no surprise, on the viewer of a domain.

Let $\mathcal{CO} \subseteq \mathcal{EL}$ be the set of concepts and let $\mathcal{LI} \subseteq \mathcal{EL}$ be the set of links. These sets should form a partition of $\mathcal{EL}$:

**[S2]** $\mathcal{CO} \cap \mathcal{LI} = \emptyset$ and $\mathcal{EL} = \mathcal{CO} \cup \mathcal{LI}$.

In terms of Figure 2.4, our viewer can now select from two classes of elements: concepts and links. This is depicted in Figure 2.5. In the next chapter, we will provide an even more refined view on the classes of elements we identify.

If $X \subseteq \mathcal{EL}$, then we will use the following abbreviations:

$$\mathcal{CO}_X \triangleq X \cap \mathcal{CO} \text{ and } \mathcal{LI}_X \triangleq X \cap \mathcal{LI}$$

Links run between concepts. We furthermore presume functions $\mathsf{From} : \mathcal{LI} \to \mathcal{CO}$ and $\mathsf{To} : \mathcal{LI} \to \mathcal{CO}$ to exist, providing the source and destination of these links respectively. As an abbreviation we will use:

$$\mathsf{Involved}(r) \triangleq \{\mathsf{From}(r), \mathsf{To}(r)\}$$

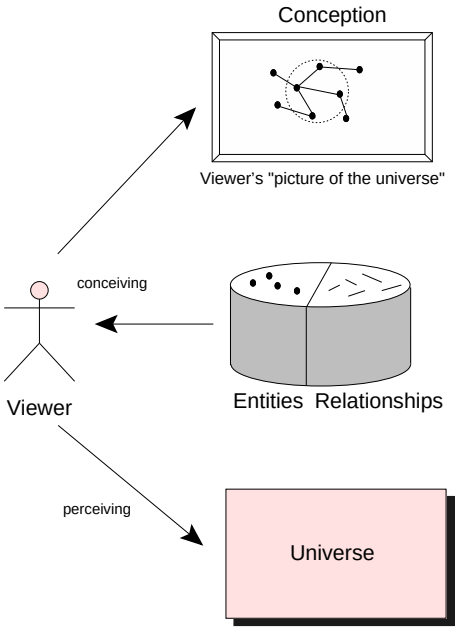



Figure 2.5: Painting a more refined picture of the observed domain

to select the concepts involved in a link, and:

$$e\ \mathsf{LinkedTo}_C\ f \triangleq \exists_{l \in \mathcal{LI}_C}\ [\mathsf{From}(l) = e \wedge \mathsf{To}(l) = f]$$

to represent the fact that a concept $e$ is connected to concept $f$.

A conception $C$ is considered to be closed iff:

$$\forall_{r \in \mathcal{LI}_C}\ [\mathsf{Involved}(r) \subseteq C]$$

Conceptions of viewers should indeed be closed:

**[S3]** If $U \models^{\mathrm{e}}_{v} C$, then $C$ is closed.

Figure 2.6 provides a model, our ontology, of the classes of elements which a conception may consist of and their mutual relationships. The notation we have used there is the ORM (Object-Role Modeling) [Hal01][7] notation, which is the same notation as used in the *Domain Modeling* course.

A conception $C$ which is closed under $\mathcal{LI}$ can essentially be regarded as a graph:

$$\langle \mathcal{CO}_C, \mathcal{LI}_C, \mathsf{From}, \mathsf{To} \rangle$$

A conception $C$ is called connected iff the associated graph is connected. A conception is required to be a connected graph:

**[S4]** If $U \models^{\mathrm{e}}_{v} C$, then $C$ is connected.

Note: if a conception would not be a connected graph, it would be a conception of multiple universes.

We generalise From and To to sets of links as follows:

$$\begin{aligned} \mathsf{From}(L) &\triangleq \{\mathsf{From}(l) \mid l \in L\} \\ \mathsf{To}(L) &\triangleq \{\mathsf{To}(l) \mid l \in L\} \end{aligned}$$

We are now in a position to properly define the environment of a domain:

[7] We have used the extension introduced in [HP95] to signify that Concept and Link are really self-defining subtypes, i.e. not requiring a dedicated subtype defining rule.

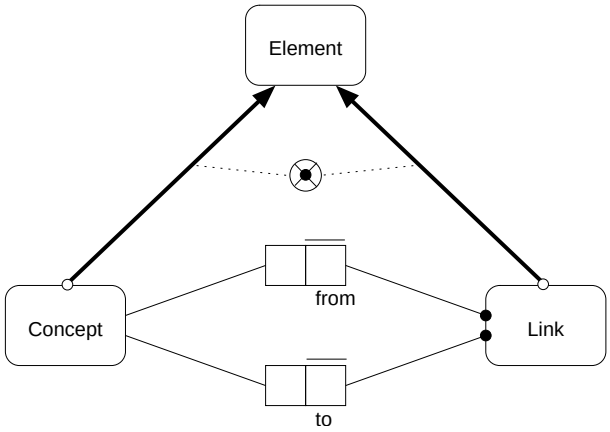



Figure 2.6: The ontology of a viewer's conception

**Environment –** The environment of a domain is that part of a viewer's conception of a universe, which has a direct link to the domain.

Formally, we view a domain $D$ and an environment $E$ as being a subset of a conception $C$, in other words a sub-conception $D, E \subseteq C$. Let $\_ \models^{\mathsf{d}}_{\_} \langle \_ : \_ : \_ \rangle \subseteq \mathcal{UN} \times \mathcal{WW} \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL})$ now be the relation expressing which domain and environment a viewer conceives. The fact that a viewer $v$ harbors a conception of domain $D$ and environment $E$ for universe $U$ can be expressed as $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$. This link should limit itself to the conceptions held by viewer $v$:

**[S5]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then $U \models^{\mathsf{c}}_v C$ and $E, D \subseteq C$.

A domain and its environment should not overlap:

**[S6]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then $E \cap D = \emptyset$.

The domain and environment should be closed:

**[S7]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then $D$ is closed.

**[S8]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then $E \cup D$ is closed.

A domain should be connected:

**[S9]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then $D$ is connected.

The combination of a domain and its environment is connected as well:

**[S10]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then $D \cup E$ is connected.

Note that an environment does not have to be connected!

Concepts in the environment are related somehow to concepts in the domain:

**Lemma 2.2.1**

Let $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$ with a non-empty environment $E$. If $P$ is a *maximum*[8] subset of $E$ such that it is connected, then:

$$\exists_{e \in \mathcal{CO}_P, r \in \mathcal{LI}_E, d \in \mathcal{CO}_D} [\{e, d\} = \mathsf{Involved}(r)]$$

**Proof:**

Left as an exercise to the reader.

The authors of [FVV+98] also define the notions of domain and environment. However, they do not take the subjectivity with regards to viewing the universe as a set of elements into consideration. As a result, they define domain and environment as being parts of the *universe* as opposed to being parts of a viewer's conception of the universe.

In the remainder of this book, we will use the phrase: *a viewer $v$ observing a domain (of interest) $D$ (with environment $E$)* as an abbreviation for: *a viewer $v$ having a conception of the universe, zooming in on domain $D$ (with environment $E$).*

[8] In other words, there is no $P'$ such that $P \subset P' \subseteq E$ while $P'$ is still connected.

**Decomposition of conceptions**

When a viewer conceives a domain, we presume there to be an concept in their conception representing the *whole* of the domain as well as one representing the *whole* of the environment. The same applies to the universe. In other words, the concepts in the domain and the environment can be regarded as decompositions of entities representing the whole of the domain and environment, while these latter concepts are decompositions of another concept representing the universe as a whole. This is illustrated in figure 2.7.

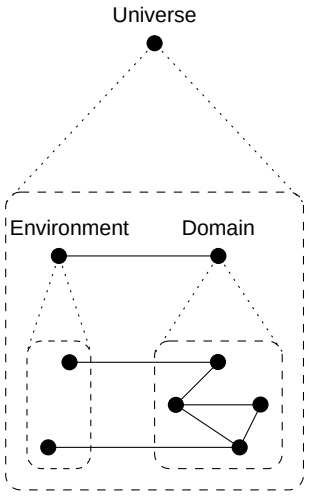



Figure 2.7: Decomposition of the universe

This 'decomposition game' can be played repeatedly. When viewing a domain a viewer may decide to zoom in further into a specific part of this domain. For example, when observing an insurance claim-handling process, involving amongst other things an evaluation of the claim, one may decide to zoom in closer into the actual evaluation process. This has been illustrated in figure 2.8.

The fact that one concept is in the 'decomposition' of another concept really means that there is a link between them in the viewer's conception. This has been illustrated in figure 2.9. This really implies we need to identify a specific class of links called decomposers. Let us therefore presume we have a set: $\mathcal{DC} \subseteq \mathcal{LI}$ of links.

To more easily reason about decompositions within a conception, we will introduce the derived relationship $\_ \rightarrow_{\_} \_ \subseteq \mathcal{CO} \times \wp(\mathcal{EL}) \times \mathcal{CO}$. If $x \rightarrow_C y$, the concept $x$ in conception $C$ is decomposed into (possibly amongst others) concept $y$. This relationship is defined (precisely) by the two following (recursive) derivation rules:

$$\begin{array}{lll} 1: & \exists_{d \in \mathcal{DC}_C} [x = \mathsf{From}(d) \wedge y = \mathsf{To}(d)] & \vdash \quad x \rightarrow_C y \\ 2: & x \rightarrow_C y \wedge y \rightarrow_C z & \vdash \quad x \rightarrow_C z \end{array}$$

The decompositions should be acyclic:

**[S11]** $x \rightarrow_C y \Rightarrow x \neq y$

As an abbreviation we introduce: $x \rightarrow\!\!\!\!=_C y \triangleq x = y \vee x \rightarrow_C y$.

To enforce the fact that a viewer's conception of a universe consists of one 'top' concept representing the universe as a whole, we require:

**[S12]** If $U \models\!\!\!\!=_v C$, then: $\exists_{u \in \mathcal{CO}_C} \forall_{x \in \mathcal{CO}_C} [u \rightarrow\!\!\!\!=_C x]$

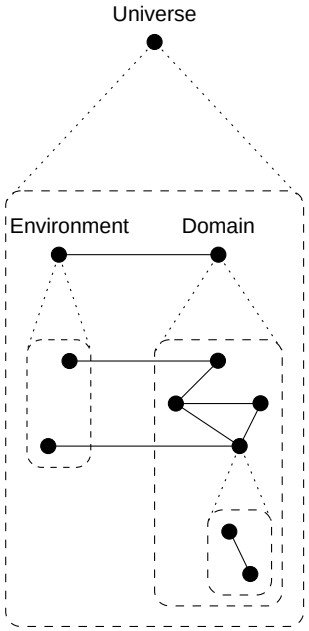



Figure 2.8: Decomposition of a part of a domain

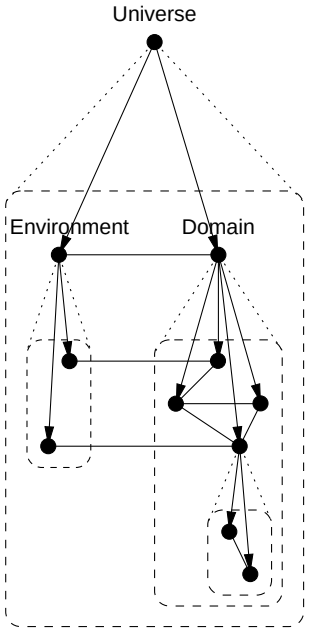



Figure 2.9: Decomposer relationships

Even more, the $u \in C$ is unique:

**Corollary 2.2.1**
If $U \models^{\mathsf{c}}_v C$, then: $\exists!_{u \in \mathcal{CC}_C} \forall_{x \in \mathcal{CC}_C} [u \rightrightarrows_C x]$

**Proof:**
Left as an exercise to the reader.

Similarly, for domains and environments we have:

**[S13]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then: $\exists_{d \in D} \forall_{x \in \mathcal{CC}_D} [d \rightrightarrows_D x]$

**[S14]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then: $\exists_{e \in E} \forall_{x \in \mathcal{CC}_E} [e \rightrightarrows_E x]$

Corollary 2.2.1 applies to each of these as well. So for $C$, $D$ and $E$ there are unique tops in the hierarchies. This unique *top* elements will be referred to as $\mathsf{Top}(C)$, $\mathsf{Top}(D)$ and $\mathsf{Top}(E)$ respectively. For these tops, we should have:

**[S15]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then: $\exists!_{d \in \mathcal{DC}_C} [\mathsf{Top}(C) = \mathsf{From}(d) \wedge \mathsf{Top}(E) = \mathsf{To}(d)]$

**[S16]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then: $\exists!_{d \in \mathcal{DC}_C} [\mathsf{Top}(C) = \mathsf{From}(d) \wedge \mathsf{Top}(D) = \mathsf{To}(d)]$

**[S17]** If $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$, then: $\exists!_{r \in \mathcal{LI}_C - \mathcal{DC}_C} [\mathsf{Top}(D) = \mathsf{From}(d) \wedge \mathsf{Top}(E) = \mathsf{To}(d)]$

These three axioms require the top part of a conception to have the structure as depicted in Figure 2.9.

By adding the set of decomposers, we have now enriched our ontology to the situation as depicted in figure 2.10. Note that the asterisk (*) attached to the is decomposed into relationship signifies this to be a derived relationship.

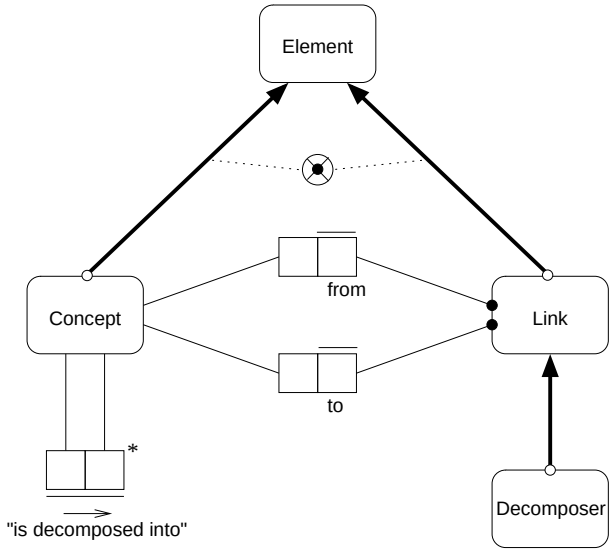



Figure 2.10: Ontology refined with decomposers

### 2.2.4 Model

In the context of organizations, we are not interested in all types of conceptions. Our interest is limited to those conceptions, that may be referred to as a *model*:

**Model –** A purposely abstracted domain (possibly in conjunction with its environment) of some 'part' or 'aspect' of the universe a viewer may have an interest in.

For practical reasons, a model will typically be consistent and unambiguous with regards to some underlying semantical domain, such as logic.

As a model is a conception, it also consists of elements, which can be specialized further into concepts and links:

**Model element –** An element from a conception which is a model.

**Model concept –** A concept from a conception which is a model.

**Model link –** A link from a conception which is a model.

We are now also in a position to define more precisely what we mean by *modeling*:

**Modeling –** The act of purposely abstracting a model from (what is conceived to be) a part of the universe.

For practical reasons, we will understand the act of *modeling* to also include the activities involved in the *description* of the model by means of some language and medium.

To represent the fact that some viewer produces a model in an environment when they observe some part of the universe, we introduce the relation:

$$\_ \models^{\mathsf{m}}_{\_} \langle \_ : \_ : \_ \rangle \subseteq \mathcal{UN} \times \mathcal{VW} \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL})$$

Formally, the fact that a viewer $v$ views the universe $U$ and in doing so has a conception $C$ including a model $M$ with environment $E$ is represented by: $U \models^{\mathsf{m}}_{v} \langle C : E : M \rangle$. Every model is a conception. We should therefore have:

**[S18]** If $U \models^{\mathsf{m}}_{v} \langle C : E : M \rangle$, then $U \models^{\mathsf{d}}_{v} \langle C : E : M \rangle$.

**Corollary 2.2.2**
If $U \models^{\mathsf{d}}_{v} \langle C : E : M \rangle$, then $U \models^{\mathsf{c}}_{v} C$.

**Proof:**
Left as an exercise to the reader.

Note that not all conceptions of a domain produce models. So the reverse of the above axiom does *not* hold! As an abbreviation we also introduce:

$$U \models^{\mathsf{m}}_{v} M \;\triangleq\; \exists_{C,E} \left[ U \models^{\mathsf{m}}_{v} \langle C : E : M \rangle \right]$$

### 2.2.5 System

Using the above general definitions, we can, in line with [FVV+98], more precisely define the way we view systems:

**System domain –** A domain that is conceived to be a system, by some viewer, by the distinction from its environment, by its coherence, and because of its systemic property.

**Systemic property –** A meaningful relationship that exists between the domain of elements considered as a whole, the system domain and its environment.

**System viewer –** A viewer of a system domain.

**System –** A special model of a system domain, whereby all the things contained in that model are transitively coherent, i.e. all of them are directly or indirectly related to each other and form a coherent whole.

A system is conceived as having assigned to it, as a whole, a specific characterisation (a non-empty set of systemic properties) which, in general, cannot be attributed exclusively to any of its components.

**System description –** The description of a system.

The elements, concepts and links concepts can be further specialized to systems:

**System element –** Any element from a system.

**System concept –** Any element from a system that is a concept.

**System link –** Any element from a system that is a link.

As identified in [FVV+98], there is a potential objection against our subjectivity-based definition of system. In daily life, it is quite sensible to talk about "designing, constructing and implementing a system" or "to interact with a system". The use of the terms 'system' gives associations to this term as denoting something that can be interacted with in a rather concrete way and not just as a conception. These associations, however, do not lead to any inconsistencies. These example phrases are simply convenient abbreviations for more elaborate expressions. For instance, "to interact with a system" really means:

> *to interact with phenomena in the system domain that is conceived as a system (because of its systemic properties).*

To "design, construct and implement" a system really means:

> *to bring together and structure phenomena in a particular part of the world (which then becomes the system domain) with the purpose of constructing them such that they together have certain systemic properties.*

To represent the fact that some viewer 'sees' a system in an environment when they observe some part of the universe, we introduce the relation:

$$\_ \models^{\mathsf{s}}_{\_} \langle \_ : \_ : \_ \rangle \subseteq \mathcal{UN} \times \mathcal{WW} \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL})$$

Formally, the fact that a viewer $v$ views the universe $U$ and in doing so has a conception $C$ including a system $S$ with environment $E$ is represented by: $U \models^{\mathsf{s}}_{v} \langle C : E : S \rangle$. Every system is a model. We should therefore have:

**[S19]** If $U \models^{\mathsf{s}}_{v} \langle C : E : M \rangle$, then $E \neq \emptyset$ and $U \models^{\mathsf{m}}_{v} \langle C : E : M \rangle$.

Note that not all conceptions of a domain produce models. So the reverse of the above axiom does *not* hold! As an abbreviation we also introduce:

$$U \models^{\mathsf{s}}_{v} S \;\triangleq\; \exists_{C,E} \left[ U \models^{\mathsf{s}}_{v} \langle C : E : S \rangle \right]$$

In our informal exploration of the concept of system, we already discussed that there are three major ways of viewing systems [Rop99]: *structural*, *functional* and *hierarchical* (as a specific class of structural). A major difference between a structural and a functional perspective is the distinction between the white-box and black-box approach when regarding systems. In other words, is one looking *inside* the system (white-box) or is one only looking at the *outside* of the system. This does seem to raise the question whether, when viewing a system as a black-box, one can still argue that the system consists of elements? The answer to this question is a resounding *yes*. When a viewer, for some reason, views a domain as a system, and does so using a black-box approach, that what they conceive of as being a system is still a conception consisting of elements. The difference between a white-box and a black-box approach when viewing a system, however, is in the concepts and links one will see. When taking a black-box approach, one will only see the external behavior of the system, while when taking the white-box approach one will see the internal structure/behavior of the system as well.

## 2.3 Studying systems

In order to really to understand the concept system it is necessary to be aware of a number of important aspects:

- that the system domain always comprises several elements,
- that all elements are related to each other such that it constitutes a transitively coherent whole,
- that the whole is conceived to have at least one systemic property,
- that it is only relevant to incorporate a thing as an element of the particular system domain if in the system view it somehow contributes to the systemic property,
- that when viewing a thing as an element of a system domain then only those aspects of the thing that directly or indirectly contributes to the systemic properties are relevant for the system view.

### 2.3.1 Sub-systems

To gain a better understanding of complex systems it has proven to be useful to identify smaller-scale systems within a larger system, leading to sub-system. A detailed discussion on dealing with complexity by systems in general, and the role played by hierarchical decomposition, may be found in e.g. [Sim62]. In this textbook, for example, information systems will be positioned as sub-systems of organizational systems.

However, when it comes to the point of being less intuitive and more explicit about the concept, there is little consensus about what really characterizes a sub-system – or rather what should characterize it, if the concept is to be a useful one. The influence from the absoluteness of the 'classical' system concept together with some apparent preference to associate the understanding of sub-system with the subset concept seem to be the main cause of the confusion.

The 'old', simple interpretation of the concept system as being just 'a set of interrelated parts', made it rather obvious to think of sub-system as: A subset of the parts together with an appropriate subset of their mutual relationships. However, with the introduction of the notion that in order for something to be a system, it must have at least one systemic property, the matters became more difficult: Should the definition of sub-system then also involve the specification of a subset of the systemic properties? Intuitively this notion could be reasonable, and it may even work in some cases, but the problem is that this is not always so. Consider, for example, a well-functioning mechanical watch. It can be conceived to have the systemic property that under certain conditions it 'shows the time'. A possible sub-system of such a watch is the energy supplying device for the clockwork consisting of the spring, the winding knob, the exchange and click mechanism for tightening the spring, and a part of the frame to support these mechanical parts. The only sensible systemic property of such a sub-system is that it serves as a storage of mechanical energy. But then we have a serious problem with the subset notion applied on the systemic property, because being an energy storage is in no way a subset of the systemic property of showing the time.

The problem of defining a sensible sub-system concept by means of subset relationships becomes even more difficult with the notion of a system as a subjective issue. Apart from the systemic properties not being absolute, but rather depending of the viewer, one element in the system domain may now also potentially be viewed as several different components in the system. Consider, for example, an organization that is viewed as an and a person from that organization: Here the person may appear as an actor of the type salesman that is the agent of various sales activities. But independent hereof, the same person may also be conceived as having the type employee relevant in connection with calculations of salaries and the planning of sales campaigns. The

person may even be regarded as being of type transportable object in the context of an activity transport by car during sales trips. This causes the following question: Should a possible subset relationship applied in attempts to define a sub-system concept then refer to the domain alone, or to the system alone, or to both? It is certainly difficult to find logical or pragmatic arguments that universally justify any of these choices. (For further aspects of the problems encountered when one is aiming at defining sub-system by means of subsets, see the more comprehensive discussion in [FVV+98].)

It is necessary to consider the sub-system concept differently – in fact, in a way that very well is in accordance with the way people intuitively apply it in practice. The 'solution' is to realize that when viewing something as a system then only one system should be considered at a time. Applied here, either one must consider that which is regarded as the system or that which is regarded as the sub-system. The advantage of this sub-system interpretation is exactly what appears to be the main positive feature of the intuitively applied concept: Depending on which level of detail as regard potential components you want to consider, you can use the concept to encapsulate unnecessary details on a chosen level of abstraction. Applied to organizations one obvious way to consider the relationship between an organization and a sub-system of it, is to conceive the sub-system equivalent with what an actor in the organization does (or a part of that). Typically a whole department (a possible system candidate in itself) may be considered a single actor in the organization, and (part of) what is done in that department in respect to other departments (i.e. possible systemic properties of the "department system") may be conceived as a single action at the organizational-level. A data-processing system may be conceived as a single (artificial) actor carrying out data-processing actions in the organization, even if we know that it, in fact, is composed of a lot of components.

A sub-system may, in line with [FVV+98], defined as:

**Sub-system –** A sub-system $S'$ of a system $S$, is a system where the set of elements in $S'$ is a subset of the elements in $S$.

Formally, this can be expressed as:

$$U \models^{s}_{v} S' \subset S \quad \triangleq \quad U \models^{s}_{v} S, \quad U \models^{s}_{v} S' \text{ and } S' \subset S$$

**Corollary 2.3.1**
If $U \models^{s}_{v} S' \subset S$, then: $\exists_{s \in S'} \forall_{x \in \mathcal{CO}_{S'}} [s \rightarrow_{S'} x]$

**Proof:**
Left as an exercise to the reader.

Two common dimensions along which to define sub-systems are: component system and aspect system.

**Component system –** A component-system $S'$ of a system $S$, is a sub-system, where the set of model concepts in $S'$ is a proper subset of the set of entities in $S$.

Formally:

$$U \models^{s}_{v} S' \subset_{c} S \quad \triangleq \quad U \models^{s}_{v} S' \subset S \text{ and } (S' \cap \mathcal{CO}) \subset S$$

**Aspect system –** an aspect-system $S'$ of a system $S$, is a sub-system, where the set of model links in $S'$ is a proper subset of the set of the links in $S$.

Formally:

$$U \models^{s}_{v} S' \subset_{a} S \quad \triangleq \quad U \models^{s}_{v} S' \subset S \text{ and } (S' \cap \mathcal{LI}) \subset S$$

Note that some authors, for example [Vel92, Bem98], use the term sub-system to refer to the above defined concept of component system. However, we prefer to use the term sub-system as defined above (following the definition in [FVV+98]), as it allows us to view it as a generalization of the concepts component system and aspect system.

Different viewers may disagree on the fact whether some sub-system is an aspect system or a component system (or a combination thereof). This can be traced back to the subjectivity involved in distinguishing between links and concepts. Whenever there is a 'clear' analogy to physical structures, it will be easier to identify the difference. Consider a freight-train as an example system. Typical component systems of such a system are: the locomotive, the engine-driver, several types of box-cars, etc. An aspect system of a freight-train would be the hydraulic braking system of the train as a whole.

A sub-system is indeed a system. As such, a sub-system $S'$ of a system $S$ will also have its own systemic properties. However, these properties are most likely no subset of the systemic properties of $S$. For example, the engine-driver's systemic properties are by no-means a clear subset of the systemic properties of a freight-train.

### 2.3.2 Describing systems

When a system developer in a system viewing or modeling process gradually realizes what (currently) 'is the system', i.e. becomes conscious of all relevant aspects of the involved elements and of each of the systemic property, it is very useful to be aware of the type of system in question and to produce a system exposition in accordance with the system type.

**System type –** A type that determines the potential kinds of systemic properties, elements of the system domain and roles of the elements in achieving the systemic properties.

**System exposition –** a description of all the elements of the system domain where each element is specified by all its relevant aspects and all the roles it plays, being of importance for the interest of the viewer. (The system viewer may conceive one and the same thing in the system domain to play more than one role in the system.)

A system type can be regarded as a viewing template to be used by a system developer, analyst or modeler in order to decide which kinds of things (and thereby which aspects of the things) to consider relevant in realizing what actually 'is the system'. A system type comprises:

- Properties determining 'the nature' of the systemic properties, for example for open active systems that the system is seen as something that changes things in the domain of the environment and that the environment is seen as changing things in the system domain. This set of properties may be called the *system characteristic*.
- Properties determining the kinds of things which are relevant to incorporate in the exposition of the system domain, and for each kind the kinds of roles they may play in respect to the potential kinds of systemic properties. Examples of such kinds of things are for dynamic systems: states, transitions and transition occurrences, and for open active systems (among other things): actions, subjects, agents, transitions in the domain of the environment caused by actions in the system, etc. This set of properties may be called the *exposition characteristic*.

A more detailed elaboration of concepts related with the system viewing process can be found in [FVV+98]. A semi-formal description of it based on an example is presented in [Lin92].

In conceiving a domain as an organization, several classes of elements may be relevant to include in a system exposition of that domain. As part of the domain it may also be relevant to incorporate a number of concepts generally relevant in an organizational context, for example public services, laws or other kinds of constraints imposed by society, or aspects of the particular professional field of the organization. However, for an organization it is generally relevant to consider the following kinds of things as candidates to (at least) be included in a system exposition:

**Actors –** human actors as well as artificial actors and all kinds of symbiotic compositions of these two kinds.

**Actions –** (together with the associated goals) such that a (not exclusive) distinction is made between those influenced by impressions from the environment and those either directly constituting expressions of the system or only contributing to (or in some cases even explicitly counteracting) the expressions. Actions that are irrelevant for the expression of the system should be ignored in the exposition.

**Co-actions –** i.e. co-ordinated actions performed by several actors together.

**Knowledge –** that is necessary for the actors to know the relevant pre-states of their actions and the respective goals. A goal may be situation dependent.

**Triggers –** involving internal and/or external dynamic criteria for the initiation of actions (temporal, impressive and actor- or action-caused transitions).

**Communication –** between actors to ensure that they have the information necessary to perform their actions.

**Representation –** of the information/knowledge relevant to the organization's activities, in order to enable the preservation or communication of it. That includes all relevant aspects of the use of data technology and/or data-technical sub-systems to accomplish the preservation or communication.

In practice, aspects of organizational culture, social norms, empation (i.e. knowledge that cannot be properly represented), resources in general (energy, skills, intellect, etc.), ecology, economy, etc., may be added to this list.

### 2.3.3 Classes of systems

A work system, a organization, as well as an information systems belong to a system type that primarily is characterized as being open and active (where the latter implies also that it is dynamic). We can define these specific types of systems as:

**Active system –** A special kind of system that is conceived of as begin able to change parts of the universe.

**Dynamic system –** A special kind of system that is conceived of as undergoing change in the cause of time.

**Open system –** A special kind of dynamic system that is conceived as reacting to external triggers, i.e. there may be changes inside the system due to external causes originating from the system's environment.

Note that a system may be active and yet be non-dynamic. For example, the mere presence of a dummy speeding camera, i.e. one that is not able to capture speeding vehicles on film, may lead drivers to drive more slowly. The dummy speeding camera may thus be seen as an active, yet non-dynamic, system.

Note that the sub-system of an open active system does not have to be an open active system. In other words, even though our main interest lies with open active system, we may quite well need to consider non-open or non-active sub-systems of these systems.

For open active systems – therefore for organizations too – it is relevant to consider the following. The behavior of an open active system is generally reflected as:

**Internal function –** Conceptions of changes in the system domain caused by processes in the domain itself.

**External function –** Here the following two kinds are distinguished:

**Impression –** Conceptions of changes in the system as caused by the environment.

**Expression –** Conceptions of changes in the environment as caused by the system.

The very fact that something is regarded as a system often serve the purpose of hiding the internal function and focus on the external function. (Like the phrase "a black-box system"). The internal function of an open active system is referred to as "the function *in* the system", while the external function is "the function *of* the system". The latter is equivalent with the systemic property of an open active system.

One can classify open active systems in several ways according to their behavior (for details see [Ack71]). Here we shall only distinguish between three kinds of open active system based on the following distinctions. A reaction of an open active system is an expression that is seen as unconditionally caused by an impression. An action of an open active system is an expression that is seen as being completely independent on any kind of impression. Thereby we can define the three additional types of open active systems:

**Reactive system –** An open active system where each expression of the system is a reaction, and where each impression immediately causes a reaction.

**Responsive system –** An open active system (possibly also a reactive system) where it holds for at least one expression that a certain impression or a temporal pattern of impressions is a necessary, but not a sufficient dynamic condition for its occurrence. The receipt of an order is a necessary impression to a "sales system", for the expression "delivery of the ordered goods", but it is not a sufficient condition.

**Autonomous system –** an open active system (possibly also a responsive system, but not a reactive system) where at least one expression is an action. A human being and most (if not all) organizations can be regarded as autonomous systems.

As mentioned before, in [Alt99, Alt02] Alter defines a work system as:

> *A work system is a system in which human participants and/or machines perform business processes using information, technologies, and other resources to produce products and/or services for internal or external customers.*

where *information systems* are to be regarded as special classes of work systems. We will therefore operate under the assumption that we have the following hierarchy of systems:

1. Systems in general.
2. Open active systems: Subclass of systems
3. Work systems: Subclass of open active systems.
4. Organizational systems: Subclass of work systems.
5. Information systems: Subclass of work systems and a sub-system of organisational systems.
6. Computerised information systems: Subclass of work systems and a sub-system of information systems.

Based on [FVV+98] and [Alt99], we can provide the following stacked set of definitions:

**Work system –** An open active system in which actors perform processes using information, technologies, and other resources to produce products and/or services for internal or external actors.

**Organizational system –** A special kind of work system, being normally active and open, and comprising the conception of how an organization is composed and how it operates (i.e. performing specific actions in pursuit of organizational goals, guided by organizational rules and informed by internal and external communication), where its systemic property are that it responds to (certain kinds of) changes caused by the system environment and, itself, causes (certain kinds of) changes in the system environment.

**Computerized information system –** A sub-system of an information system, whereby all activities within that sub-system are performed by one or several computer(s).

**Information system –** A sub-system of an organizational system, comprising the conception of how the communication and information-oriented aspects of an organization are composed and how these operate, thus leading to a description of the (explicit and/or implicit) communication-oriented and information-providing actions and arrangements existing within the organizational system.

## 2.4 Dealing with evolution of conceptions

Before concluding this chapter, there is one final issue to deal with. An organizational system is an open active system. This specifically means that it is a system which changes over time. Thus far we have taken the assumption that the conception of a viewer is a static notion. If we write $U \models^{c}_{v} C$ it really means that viewer $v$ has *at some point in time* the conception $C$ when observing (a part of) universe $U$. However, in the course of time this conception will evolve, which raises the question: *How to deal with evolution of conceptions?*.

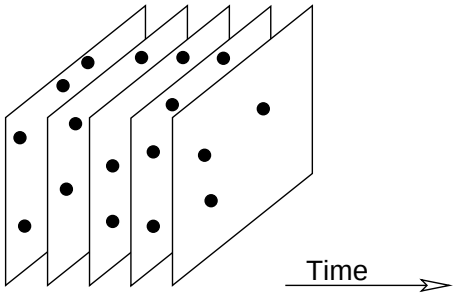



Figure 2.11: Modeling evolution by snapshots

Several strategies exist to deal with evolution [Pro94a]. One strategy to deal with this evolution is to take snapshots, like photographs, of a viewer's conceptions. This leads to the situation depicted in figure 2.11. This approach, however, does have as drawback that one cannot 'trace' the evolution of a specific element in a viewer's conception. The approach we take, therefore, is illustrated in figure 2.12.

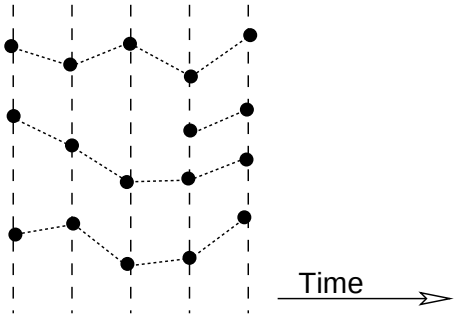



Figure 2.12: Modeling evolution by functions in time

Based on the approach taken in [Pro94a], the evolution of the elements in a viewer's conception is treated as a set of (partial[9]) functions over time. At each point in time, a specific element (a version) may be associated to such a function. This means (as also illustrated in figure 2.13), the situation depicted in figure 2.11 can still be derived. When we know the entire evolution

[9]For a discussion on the difference between total and partial functions, see appendix A.

of a nation, we can also provide a detailed descriptions of the state-of-affairs as it holds at any arbitrary point in time.

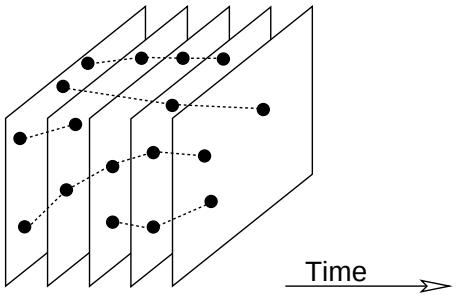



Figure 2.13: Deriving snapshots

In order to introduce the notion of evolution formally, we first need to define a time axes. We presume all viewers to agree that the time axes (at least) consists of:

- A set of points in time $\mathcal{TI}$.
- A complete and total order $< \subseteq \mathcal{TI} \times \mathcal{TI}$.

We also define: $t_1 \leq t_2 \triangleq t_1 < t_2 \vee t_1 = t_2$.

Since we have a complete order on the points in time, we can define the relation $\rhd \subseteq \mathcal{TI} \times \mathcal{TI}$ with intended intuition: if $t_1 \rhd t_2$, then $t_2$ is the next point in time immediately after $t_1$. Formally, this relation is defined as:

$$t_1 \rhd t_2 \triangleq t_1 < t_2 \wedge \neg \exists_s [t_1 < s < t_2]$$

As, $<$ provides a *complete* and *total* order, for a given $t_1$ there is always at most one $t_2$ such that $t_1 \rhd t_2$:

**Corollary 2.4.1**

$t_1 \rhd t_2 \wedge t_1 \rhd t_3 \Rightarrow t_2 = t_3$

**Proof:**

Left as an exercise to the reader.

As a result, we can actually view $\rhd$ as a function: $\rhd : \mathcal{TI} \rightarrow \mathcal{TI}$ and write $\rhd t$ as an abbreviation for: the unique $t'$ such that $t \rhd t'$.

The evolution over time of an element from a conception, can now indeed be modelled formally as a function: $h : \mathcal{TI} \rightarrowtail \mathcal{EL}$. By means of $h(t)$ we obtain the "version" of the element's evolution as described by $h$ at point of time $t$, while $h(\rhd t)$ would yield the version at the next point in time. These functions, which represent an element's evolution, will be referred to as *element evolutions*. This also requires us to think of the elements in $\mathcal{EL}$ as the possible *versions* an element evolution may take on. Whenever we need to emphasize this, we shall therefore use the term *element version*.

Element evolutions are *partial* functions, which means that they are not required to be defined for all points in time. In other words, at some point in time an element evolution may not have an element version associated, which really means that the element evolution does not exist yet at that point in time (it has not been born yet), or that it has ceased to exist (it died). In other words, element evolutions are allowed to be re-born. We could, for example, have a situation where: $h$ is an element evolution, $t_1 < t_2 < t_3$ are points in time, while: $h{\downarrow}t_t \wedge \neg\, h{\downarrow}t_2 \wedge h{\downarrow}t_3$[10].

The set of all element evolutions is defined as: $\mathcal{EE} \triangleq \mathcal{TI} \rightarrowtail \mathcal{EL}$, in other words the set of partial functions from the time axes to the possible element versions. The evolution of an entire conception can then be represented formally as a set of element evolutions. In other words: $\mathcal{CE} \triangleq \wp(\mathcal{EE})$. To formally express the fact that $H \in \mathcal{CE}$ is a conception of viewer $v$ for universe $U$, we will write $U \models^{\mathsf{c}}_{v} H$.

[10] Please refer to appendix A for an explanation of the notion used.

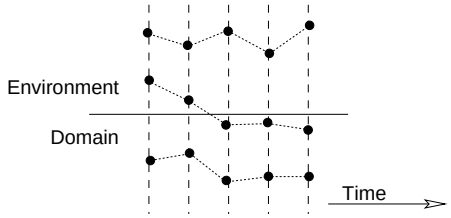



Figure 2.14: An element evolution migrating from the environment to the domain

If $H \in \mathcal{CE}$, then we will use as abbreviation: $H(t) \triangleq \{h(t) \mid h \in H\}$, yielding the entire "state" of $H$ at time $t$. If $H$ is a conception of some viewer, then each of the states should be a valid conception:

**[S20]** If $U \models^{\mathsf{c}}_v H$, then: $\forall_{t \in \mathcal{TI}} [U \models^{\mathsf{c}}_v H(t)]$.

Similarly, to $\_ \models^{\mathsf{c}}_\_ \_$ we want to extend $\_ \models^{\mathsf{d}}_\_ \langle \_ : \_ : \_ \rangle$, $\_ \models^{\mathsf{m}}_\_ \langle \_ : \_ : \_ \rangle$ and $\_ \models^{\mathsf{s}}_\_ \langle \_ : \_ : \_ \rangle$ to deal with evolution as well. We might, for example, view $U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle$ to means: viewer $v$ has a conception evolution $H_C$, environment evolution $H_E$ and domain evolution $H_D$ of universe $U$, where $H_D, H_E \subseteq H_C$. The difficulty with this is that an element evolution might start out as being in the environment, but might evolve *into* the domain, or vice versa. This is illustrated in Figure 2.14. To properly deal with such evolution, we will need to regard $H_E$ and $H_D$ as a classification of the element evolutions from $H_C$ at each point in time. In other words as functions:

$$H_E, H_D : \mathcal{TI} \to \wp(H_C)$$

yielding the set of conception evolutions from $H_C$ that are part of the environment/domain respectively at a given point in time. This means that $H_E(t), H_D(t) \subseteq H_C$ are sets of element evolutions, while $H_E(t)(t)$ and $H_D(t)(t)$ yield the actual environment and domain as it holds at $t$. For this we have:

**Corollary 2.4.2**
If $U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle$, then: $\forall_{t \in \mathcal{TI}} [H_E(t)(t) \subseteq H_C(t)]$ and $\forall_{t \in \mathcal{TI}} [H_D(t)(t) \subseteq H_C(t)]$.

**Proof:**
Left as an exercise to the reader.

Note: the same would hold for $U \models^{\mathsf{m}}_v \langle H_C : H_E : H_D \rangle$ and $U \models^{\mathsf{s}}_v \langle H_C : H_E : H_D \rangle$. In each case, the states of the evolutions should be valid conceptions/environments/domains as well:

**[S21]** If $U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle$, then $\forall_{t \in \mathcal{TI}} \left[U \models^{\mathsf{d}}_v \langle H_C(t) : H_E(t)(t) : H_D(t)(t) \rangle\right]$

**[S22]** If $U \models^{\mathsf{m}}_v \langle H_C : H_E : H_D \rangle$, then $\forall_{t \in \mathcal{TI}} [U \models^{\mathsf{m}}_v \langle H_C(t) : H_E(t)(t) : H_D(t)(t) \rangle]$

**[S23]** If $U \models^{\mathsf{s}}_v \langle H_C : H_E : H_D \rangle$, then $\forall_{t \in \mathcal{TI}} [U \models^{\mathsf{s}}_v \langle H_C(t) : H_E(t)(t) : H_D(t)(t) \rangle]$

Adding evolution of conceptions to our ontology from figure 2.10, leads to the refinement of our ontology to the situation as depicted in figure 2.15.

Mainly due to Axiom **S6** (page 31), we have:

**Corollary 2.4.3**
If $U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle$, then: $\forall_{t \in \mathcal{TI}} [H_E(t)(t) \cap H_D(t)(t) = \emptyset]$

**Proof:**
Left as an exercise to the reader.

Even more specifically, we have:

**Lemma 2.4.1**
If $U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle$, then: $\forall_{t \in \mathcal{TI}} [H_E(t) \cap H_D(t) = \emptyset]$

**Proof:**
Left as an exercise to the reader.

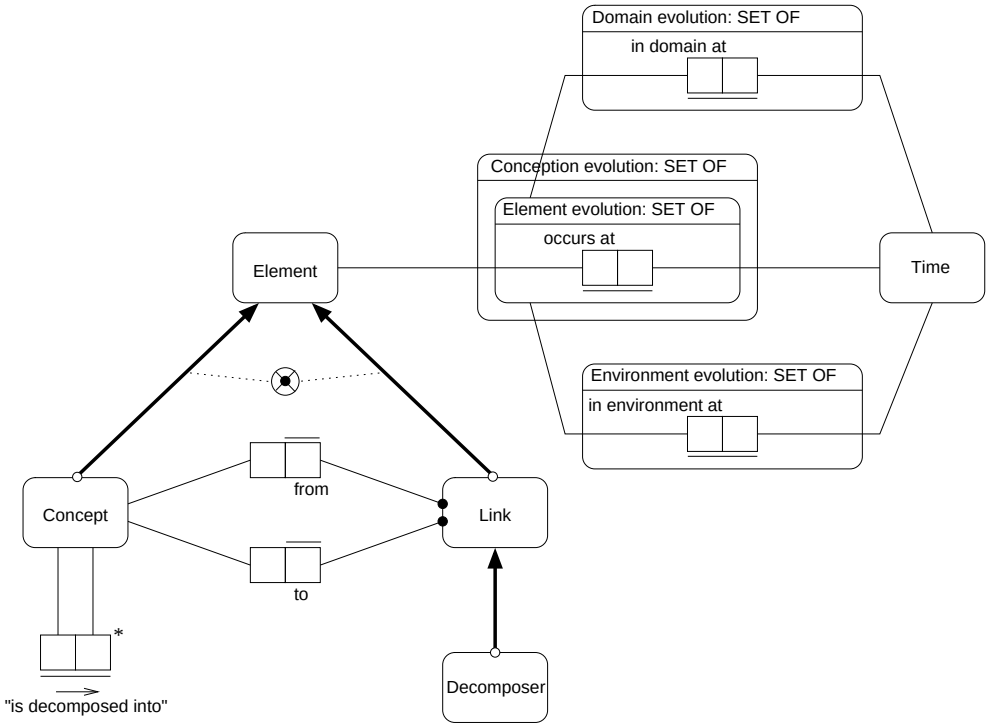



Figure 2.15: Ontology with evolution added

## 2.5 Conclusion

In this chapter we have taken a highly fundamental and formal outlook on information systems, organizations and work systems, and the way they are modeled, including their decompositions and evolutions.

## Questions

Version: 16-03-05

1. How are the terms 'organization', 'domain' and 'universe' be related to each other, given the definitions provided in this textbook?
   - Describe this relation in natural language.
   - Describe this relation in the formal language given in this chapter.
2. Proof Axiom **S8** (page 31).
3. Proof Lemma 2.2.1 (page 31).
4. Not all conceptions of a domain produce models. Why not?
5. Beschouw een Autoproducent, zoals bijvoorbeeld Seat, BMW en Toyota.
   (a) Wat zijn de belangrijkste systemische eigenschappen?
   (b) Beschrijf het primaire gedrag van deze organisatie in termen van interne en externe functies.
6. Give an example of a reactive system, of a responsive system and of an autonomous system (other examples than the ones already given, of course).
7. From a modeling point of view, organizations can be considered as systems containing a.o. concepts and links.

- Why is it important to be aware of the aspect of subjectivity when creating models?
- What view does an information system developer have when modeling organizations?
- Why would an information system developer want to start by creating a model of an organization, instead of directly focusing on modeling an information system?

8. Waarom zullen verschillende mensen wanneer ze verschillende domeinen modelleren toch verschillende modellen opleveren? Hoe kun je deze situatie verbeteren? Waarom zou je dit willen verbeteren?
9. Stel $U \models^{\text{s}}_{v} S' \subset S$, bewijs/beargumenteer dan dat:
$$|\mathcal{LI}_{S'}| + 1 \geq |\mathcal{CO}_{S'}|$$
10. Beschouw Axiom **S3** en Axiom **S4**. Daarin worden de voorwaarden voor een gesloten, connected graph gegeven. Stel dat we te maken hebben met een niet-connected graph.
    - Is het waarschijnlijk dat we het hier over hetzelfde Universe hebben? Beargumenteer je antwoord.
    - In een connected graph is de relatie tussen het aantal concepten en links $|\mathcal{LI}_C| + 1 \geq |\mathcal{CO}_c|$.
      Kun je ook een soortgelijke relatie onderkennen als een graph niet connected is? Zo ja, welke? Zo nee, waarom niet?
11. Proof Corollary 2.2.1 (page 34).
12. Suppose you are requested by a large organization (a holding company holding some daughter companies) to create more insight into their own activities by creating some models of their organization. The focus of this models must, according to the board of directors, be on their internal information flows, since the organization has the impression that a lot of business efficiency is lost due to an incompetent set of information systems. Keeping in mind what is explained in the two previous chapters, give an impression of:
    (a) Where would you start modeling?
    (b) What would you model?
    (c) Why model that?
13. Proof Corollary 2.3.1 (page 38).
14. Consider a home cinema set.
    (a) Describe the systems elements.
    (b) Distinguish proper sub-systems.
    (c) Can you derive typical aspect systems and component systems?

    Explain your answers.
15. Consider a travel agency.
    (a) Describe the most important system characteristics and exposition characteristics.
    (b) Describe its behavior in terms of internal and external functions.
16. Describe why *information* systems contain *data*bases. Use the descriptions of the terms data, information and knowledge as described in Chapter 2 in your description.
17. Give some examples of:
    (a) Work systems that are not organizational systems.
    (b) Organizational systems.
    (c) Information systems.
18. Describe, in your own words, the differences between knowledge, information and data.
19. Proof Corollary 2.4.1 (page 43).
20. Proof Corollary 2.4.2 (page 44).
21. Proof Corollary 2.4.3 (page 44).
22. Proof Lemma 2.4.1 (page 44).

# Bibliography

[Ack71] R.L. Ackoff. Towards a System of System Concepts. *Management Science*, 17, July 1971.

[Alt99] S. Alter. A general, yet useful theory of information systems. *Communications of the Association for Information Systems*, 1(13), 1999.
`http://cais.isworld.org/articles/1-13/default.asp`

[Alt02] S. Alter. The work system method for understanding information systems and information system research. *Communications of the Association for Information Systems*, 9(9):90–104, 2002.
`http://cais.isworld.org/articles/default.asp?vol=9&art=6`

[Bem98] T.M.A. Bemelmans. *Bestuurlijke Informatiesystemen en Automatisering*. Kluwer, Deventer, The Netherlands, EU, 7th edition, 1998. In Dutch. ISBN 9026727984

[Ber01] L. von Bertalanffy. *General Systems Theory – Foundations, Development, Applications*. George Braziller, New York, New York, USA, revised edition, 2001. ISBN 0807604534

[Che81] P. Checkland. *Systems thinking, systems practice*. John Wiley & Sons, New York, New York, USA, 1981. ISBN 0471279110

[FVV+98] E.D. Falkenberg, A.A. Verrijn–Stuart, K. Voss, W. Hesse, P. Lindgreen, B.E. Nilsson, J.L.H. Oei, C. Rolland, and R.K. and Stamper, editors. *A Framework of Information Systems Concepts*. IFIP WG 8.1 Task Group FRISCO, IFIP, Laxenburg, Austria, EU, 1998. ISBN 3901882014

[Hal01] T.A. Halpin. *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Mateo, California, USA, 2001. ISBN 1558606726

[HP95] T.A. Halpin and H.A. (Erik) Proper. Subtyping and Polymorphism in Object–Role Modelling. *Data & Knowledge Engineering*, 15:251–281, 1995.

[IEE00] Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471–2000, The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE, Piscataway, New Jersey, USA, September 2000. ISBN 0738125180
`http://www.ieee.org`

[Iiv83] J. Iivari. Contributions to the theoretical foundations of systemeering research and the PIOCO model. Technical Report 150, University of Oulu, Oulu, Finland, EU, 1983. ISBN 9514215435

[Lan71] B. Langefors. *Editorial notes to: Computer Aided Information Systems Analysis and Design*. Studentlitteratur, Lund, Sweden, EU, 1971.

[Lin92] P. Lindgreen. A General Framework for Understanding Semantic Structures. In E.D. Falkenberg, C. Rolland, and E.N. El Sayed, editors, *Information System Concepts: Improving the understanding – Proceedings of the second IFIP WG8.1 working conference (ISCO–2), Alexandria, Egypt*, Amsterdam, The Netherlands, EU, April 1992. North–Holland/IFIP WG8.1. ISBN 0444895078

[Mer03] Meriam–Webster Online, Collegiate Dictionary, 2003.
`http://www.webster.com`

[Pei69a] C.S. Peirce. *Volumes I and II – Principles of Philosophy and Elements of Logic*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969. ISBN 0674138007

[Pei69b] C.S. Peirce. *Volumes III and IV – Exact Logic and The Simplest Mathematics*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969. ISBN 0674138005

[Pei69c] C.S. Peirce. *Volumes V and VI – Pragmatism and Pragmaticism and Scientific Metaphysics*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969. ISBN 0674138023

[Pei69d] C.S. Peirce. *Volumes VII and VIII – Science and Philosophy and Reviews, Correspondence and Bibliography*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969. ISBN 0674138031

[Pro94] H.A. (Erik) Proper. *A Theory for Conceptual Modelling of Evolving Application Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1994. ISBN 909006849X

[Rop99] G. Ropohl. Philosophy of Socio–Technical Systems. *In Society for Philosophy and Technology*, 4(3), 1999.

[Sim62] H.A. Simon. The architecture of complexity. In *Proceedings of the American Philosophical Society*, volume 106, pages 467–482, 1962.

[Vel92] J. in 't Veld. *Analyse van organisatieproblemen – Een toepassing van denken in systemen en processen*. Stenfert Kroese, Leiden, The Netherlands, EU, 1992. In Dutch. ISBN 9020722816

# Chapter 3

# Basic Object-Role Modeling

Version: 30-01-06

The previous chapter did (see Figure 2.1) refer to the fact that viewers are able to provide a description of the conception. However, we did not really follow up on this. This chapter, however, will indeed take these descriptions as a starting point. In this chapter, we will essentially provide a brief summary of the modeling approach from *Domain Modeling*. In Chapter 5, we will enrich this modeling approach with constructs that allow us to model work systems.

## 3.1 Natural language grounding of modeling

It is not an uncommon approach to base modeling on natural language analysis:

- ORM [Hal01],
- NIAM [Win90, NH89],
- UML use cases [BRJ99],
- DEMO [RMD99],
- KISS [Kri94] and
- OOSA [EKW92].

When people work together, they are bound to use some language. The language skills of the human race evolved hand-in-hand with the levels of organization of our activities. From organization of hunting parties by our pre-historic ancestors, to the organization of factories and businesses in the present. Without the use of language, it would not have worked. As a result, most (if not all) organizations we see around us are social constructs that are the result of communication between actors, mostly *human* actors.

This makes it all the more natural to base our modeling endeavors on the language we use most to talk about organizations, i.e. natural language.

## 3.2 The logbook heuristic

Natural language based modeling approaches such as ORM employ different variations of the so-called telephone heuristic. This heuristic presumes some viewer to observe a domain (including its *evolution*), and use a 'telephone' to convey their observations to some other person (or computer). This is depicted in Figure 3.1. The left hand viewer tells the right hand viewer 'what they see'.

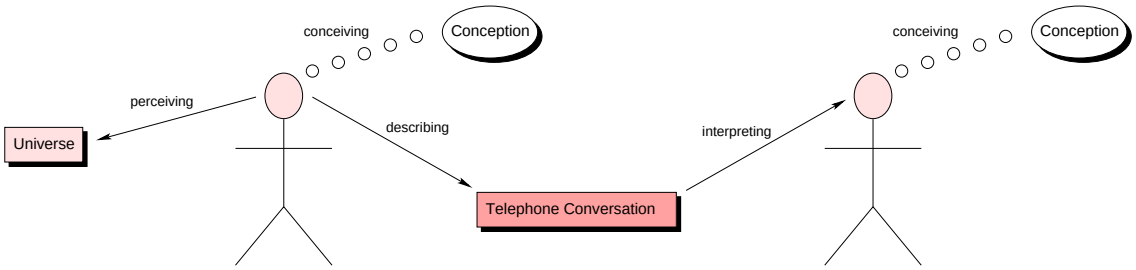



Figure 3.1: The telephone heuristic

In this chapter we are interested in having a "transcript" of the telephone conversation from Figure 3.1. More specifically, we want to maintain a *logbook* of this telephone conversation, leading to the situation as depicted in Figure 3.2.

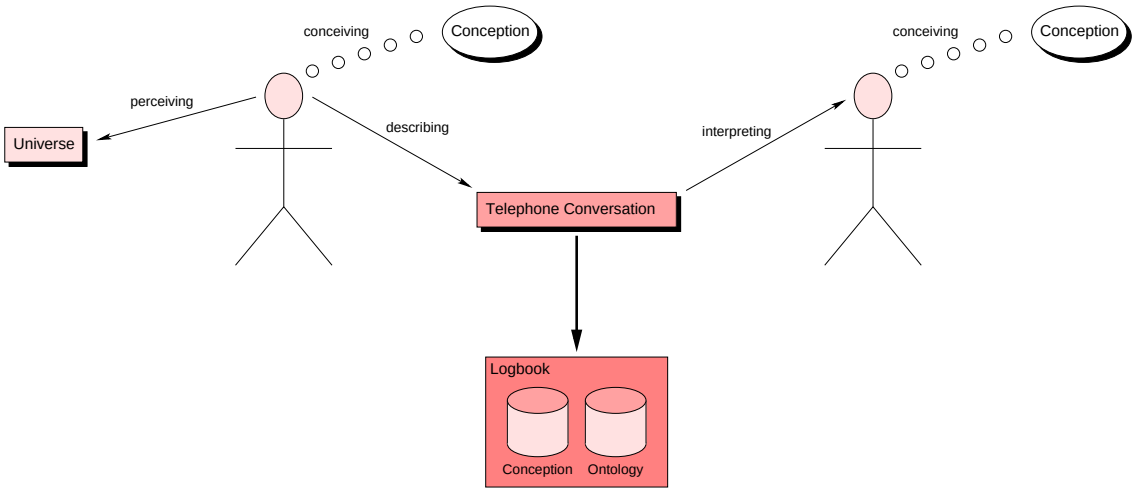



Figure 3.2: Logging the telephone conversation

Even more, we could actually replace the second person from Figure 3.1, leaving only the original viewer and the logbook to maintain the transcript. This leads to the *logbook heuristic* as depicted in Figure 3.3.

Note that the logbook is regarded as having a *conception* based on an *ontology* as well. As a starting point, we will presume this ontology to consist at least of the situation as depicted in Figure 2.15 (page 45).

Let $\lambda$ be a logbook, logging the transcript as produced by a viewer leads to a situation where the logbook has its own conception of the observed universe (by way of the viewer). In the case of a viewer observing a system, we will denote this as: $U \models^{s}_{\lambda} \langle H_C : H_E : H_S \rangle$.

All of the **S** axioms apply to the conception held by logbook $\lambda$.

In the remainder of this text book, we will further refine the ontology as presented in Figure 2.15 (page 45)

## 3.3 Verbalizing conceptions

We presume the transcriptions that are entered into the logbook to refer to events 'in the life' of specific element evolutions. These events refer to changes in the state of the elements, and are

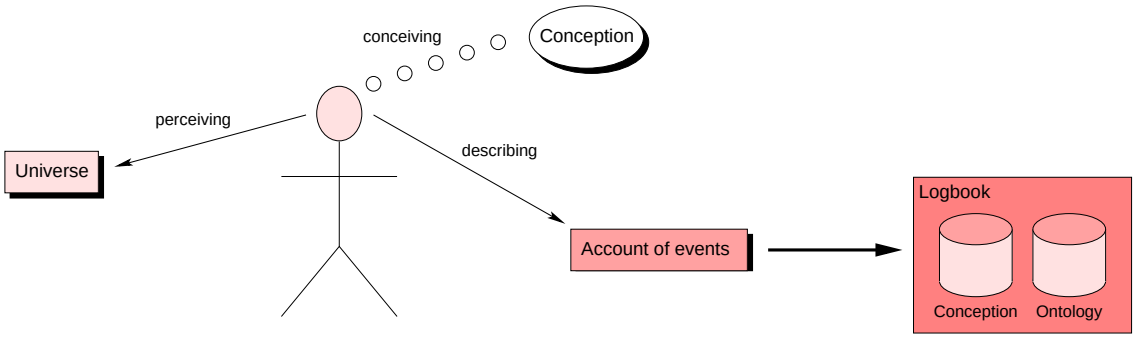



Figure 3.3: The logbook heuristic

presumed to be reported in terms of *facts* about the elements. An example would be:

| | |
|---|---|
| Person #001 was born | 22-05-1967 |
| Person #001 received name Erik Proper | 23-05-1967 |
| Person #001 lives at address: Koperwiekstraat 6, Rheden, The Netherlands, EU | 23-05-1967 |
| Person #001 lives at address: 3/26 Rylatt Street, Brisbane, Australia | 28-06-1994 |
| Person #002 lives at address: Koperwiekstraat 6, Rheden, The Netherlands, EU | 29-06-1994 |
| Person #001 works for employer: University of Queensland | 28-06-1994 |
| Person #003 works for employer: University of Queensland | 22-04-1995 |

When considering this transcript, it is easy to spot that it really deals with more than one element evolution. The following element evolutions *might* be discerned:

persons: #001; #002; #003
name Erik Proper
addresses: Koperwiekstraat 6, Rheden, The Netherlands, EU; 3/26 Rylatt Street, Brisbane, Australia
employer: University of Queensland
ownership of the name Erik Proper by person #001
living of person #001 at some address
living of person #002 at some address
habitation of address Koperwiekstraat 6, Rheden, The Netherlands by some person
habitation of address 3/26 Rylatt Street, Brisbane, Australia by some person
coworkership of person #001 for some employer
coworkership of person #002 for some employer
employment offered by University of Queensland to a group of people

In the above example, an important trade-off already comes to the surface. What should be selected as element evolutions:

coworkership of person #001 for some employer

and/or

employment offered by University of Queensland to a group of people

What is it that evolves? Either? Both? Ultimately, this is a subjective matter. To be able to better understand the underlying trade-off, we will now first focus on the transcription of a specific snapshot of a conception.

## 3.4 Elementary facts

Similarly to *Domain Modeling*, we require the facts in the transcripts to be *elementary*, in other words, no logical connectors like *and* and *or*, and most likely no *not*s either.

Consider, the following domain:

A person with name Erik is writing a letter to his loved one, at the desk in a romantically lit room, on a mid-summer's day, using a pencil, while the cat is watching.

We can rephrase this as the set of elementary facts:

A person is writing a letter
This person has the name Erik
This letter has a romantic nature
This letter has intended recipient Erik's loved one
The writing of this letter by Erik, occurs on a mid-summer's day
The writing of this letter by Erik, is done using a pencil
The writing of this letter by Erik, is done while the cat is watching
The writing of this letter by Erik, is taking place at a desk
This desk is located in a room
This room is romantically lit

Within these elementary facts, several *players* can be discerned, such as: person Erik, letter, etc. Such players can be any 'object', be it human, physical, social, fictive, etc. The players are regarded as playing a *role* in the facts. In the above example, we can isolate the players and facts as follows:

[A person] is writing [a letter]
[This person] has [the name Erik]
[This letter] has a [romantic nature]
[This letter] has intended recipient [Erik's loved one]
[The writing of this letter by Erik], occurs on a [mid-summer's day]
[The writing of this letter by Erik], is done using [a pencil]
[The writing of this letter by Erik], is done while [the cat] is watching
[The writing of this letter by Erik], is taking place at [a desk]
[This desk] is located in [a room]
[This room] is lit in [a romantic] way

The roles played by the players can be made more explicit as follows:

[A person (writer)] is writing [a letter (written)]
[This desk (positioned object)] is located in [a room (location)]

The roles involved in a fact are linked to the *players* of these roles and to the *facts* in which these roles take part by means of the relationships:

$$\mathsf{Player}, \mathsf{Fact} \subseteq \mathsf{LinkedTo}$$

respectively, with intuition:

$$\begin{array}{lcl} r\ \mathsf{Player}\ x & = & \text{role } r \text{ is played by } x \\ r\ \mathsf{Fact}\ x & = & \text{role } r \text{ is involved in fact } x \end{array}$$

The set of roles is defined as:

$$\mathcal{RO} \triangleq \left\{ r \;\middle|\; \exists_x \left[ r\ \mathsf{Player}\ x \vee r\ \mathsf{Fact}\ x \right] \right\}$$

The Player and Fact relationships are exclusive:

**[S24]** $\mathsf{Player} \cap \mathsf{Fact} = \emptyset$

Even more, they behave as total functions from roles to concepts:

**[S25]** $\mathsf{Player}, \mathsf{Fact} \in \mathcal{RO} \to \mathcal{CO}$

This allows us to write $\mathsf{Player}(r)$ and $\mathsf{Fact}(r)$. We generalize these functions to sets of roles as follows:

$$\begin{array}{rcl} \mathsf{Player}(R) & \triangleq & \left\{ \mathsf{Player}(r) \;\middle|\; r \in R \right\} \\ \mathsf{Fact}(R) & \triangleq & \left\{ \mathsf{Fact}(r) \;\middle|\; r \in R \right\} \end{array}$$

With this we can define the set of facts and players as:

$$\begin{aligned} \mathcal{FC} &\triangleq \mathsf{Fact}(\mathcal{RO}) \\ \mathcal{PL} &\triangleq \mathsf{Player}(\mathcal{RO}) \end{aligned}$$

The set of objects is defined as:

$$\mathcal{OB} \triangleq \mathcal{FC} \cup \mathcal{PL}$$

Objects and roles are disjunct classes of concepts:

**[S26]** $\mathcal{RO} \cap \mathcal{OB} = \emptyset$

The set of roles involved in a fact is defined as:

$$\mathsf{RolesOf}(f) \triangleq \{ r \mid \mathsf{Fact}(r) = f \}$$

Facts are to be regarded as complex objects. In other words, the roles and players involved in a fact are part of its decomposition:

**[S27]** $\forall_{r \in \mathsf{RolesOf}(f)} [r \ \mathsf{DecompOf} \ f]$

**[S28]** $r \in \mathcal{RO} \wedge r \ \mathsf{DecompOf} \ f \Rightarrow \mathsf{Player}(r) \ \mathsf{DecompOf} \ f$

At this moment, we have actually refined our ontology from Figure 2.15 (page 45) to the situation as depicted in Figure 3.4 (where we have omitted the aspects pertaining to the evolution of conceptions for reasons of compactness).

Our formal considerations take place in the context of some logbook $\lambda$. In other words, we have: $U \overset{s}{\models}_{\lambda} \langle H_C : H_E : H_S \rangle$. Given a $H_C$, the set of elements of the conception at some point in time $t$ is given by: $H_C(t)$. For any subset $X \subseteq \mathcal{EL}$ of elements we introduce the abbreviation:

$$X_t \triangleq X \cap H_C(t)$$

A conception version should be closed with regards to the roles included in it:

**[S29]** $r \in \mathcal{RO}_t \Leftrightarrow \mathsf{Fact}(r) \in \mathcal{FC}_t$ and $r \in \mathcal{RO}_t \Leftrightarrow \mathsf{Player}(r) \in \mathcal{PL}_t$

As a direct consequence we have:

**Corollary 3.4.1**
$\mathcal{FC}_t = \mathsf{Fact}(\mathcal{RO}_t)$ and $\mathcal{PL}_t = \mathsf{Player}(\mathcal{RO}_t)$.

**Proof:**
Left as an exercise to the reader.

## 3.5 From instances to types

Consider the elementary sentences:

Person "Erik" is examined by Doctor "Jones"
Person "Wil" is examined by Doctor "Smith"
Person "Marc" is examined by Doctor "Jones"

As we have learned in *Domain Modeling*, we can generalize these sentences to the "type" level:

A Person is examined by a Doctor

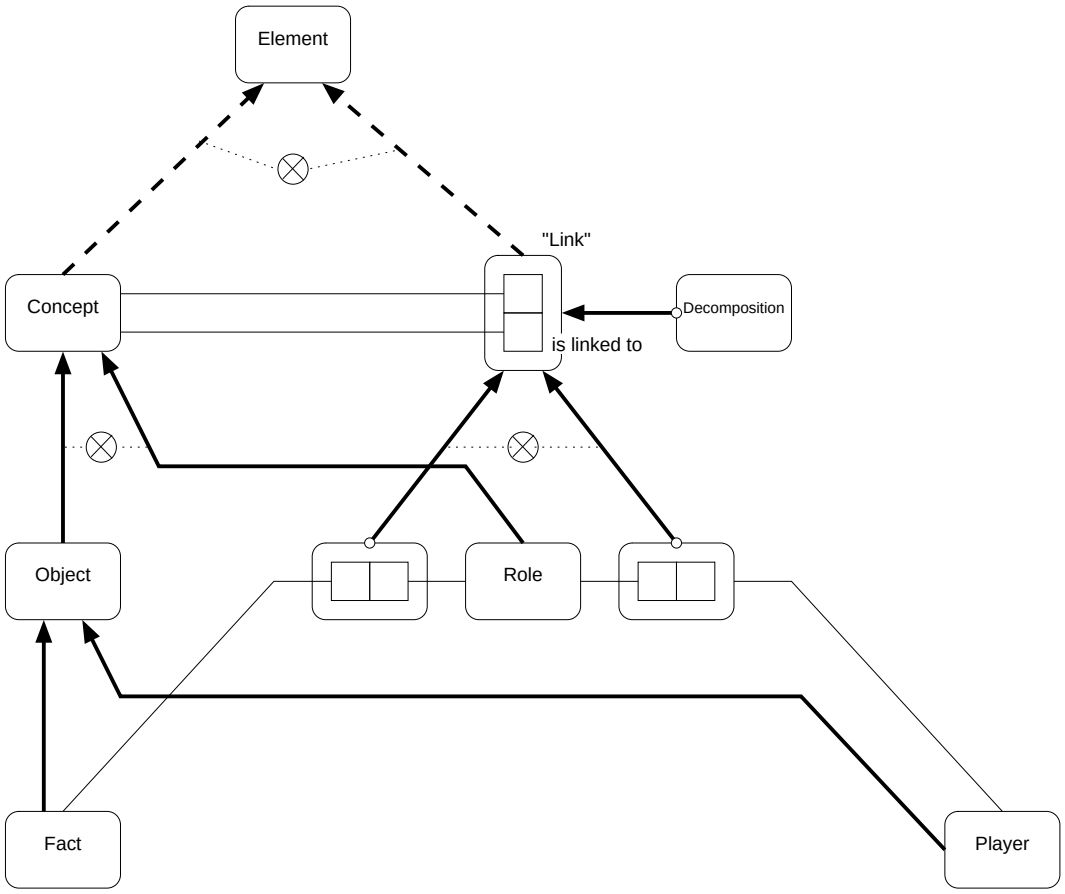



Figure 3.4: Our ontology refined with facts and participants

with sample population:

| Person | Doctor |
|---|---|
| Erik | Jones |
| Wil | Smith |
| Marc | Jones |

Formally, we introduce typing as a special kind of decomposition $\mathsf{HasType} \subseteq \mathsf{DecompOf}$. The set of instances ($\mathcal{IN}$) and the set of types ($\mathcal{TP}$) can be defined as:

$$\begin{aligned} \mathcal{TP} &\triangleq \{y \mid \exists_x [x\ \mathsf{HasType}\ y]\} \\ \mathcal{IN} &\triangleq \{x \mid \exists_y [x\ \mathsf{HasType}\ y]\} \end{aligned}$$

Types and instances form a partition of the set of concepts:

**[S30]** $\mathcal{TP} \cap \mathcal{IN} = \emptyset$ and $\mathcal{TP} \cup \mathcal{IN} = \mathcal{CO}$.

All instances have some type:

**Corollary 3.5.1**
$\forall_{x \in \mathcal{IN}} \exists_{y \in \mathcal{TP}} [x\ \mathsf{HasType}\ y]$

**Proof:**
Left as an exercise to the reader.

Note that the reverse does not hold. In other words, we do not generally have:

$$\forall_{y \in \mathcal{TP}} \exists_{x \in \mathcal{IN}} [x\ \mathsf{HasType}\ y]$$

Some types may have an empty population.

We actually require Corollary 3.5.1 to hold at each point in time:

**[S31]** Let $t \in \mathcal{TI}$, then: $\forall_{x \in \mathcal{IN}} \exists_{y \in \mathcal{TP}} [x\ \mathsf{HasType}_t\ y]$

Sets such as $\mathcal{FC}$, $\mathcal{PL}$, etc, contain both types and instances. To be able to refer to the types and instances respectively, we introduce:

$$\hat{X} \triangleq X \cap \mathcal{TP} \quad \text{and} \quad \check{X} \triangleq X \cap \mathcal{IN}$$

for any set $X \subseteq \mathcal{EL}$ of elements. As a direct consequence we have:

**Corollary 3.5.2**

$$\begin{aligned} \hat{\mathcal{PL}} &\triangleq \mathsf{Player}(\hat{\mathcal{RO}}) & \check{\mathcal{PL}} &\triangleq \mathsf{Player}(\check{\mathcal{RO}}) \\ \hat{\mathcal{FC}} &\triangleq \mathsf{Fact}(\hat{\mathcal{RO}}) & \check{\mathcal{FC}} &\triangleq \mathsf{Fact}(\check{\mathcal{RO}}) \end{aligned}$$

**Proof:**
Left as an exercise to the reader.

As abbreviations we will also use:

$$\begin{aligned} \mathsf{Types}_t(x) &\triangleq \{y \mid x\ \mathsf{HasType}_t\ y\} \\ \mathsf{Types}_t(X) &\triangleq \bigcup_{x \in X} \mathsf{Types}_t(x) \\ \mathsf{Pop}_t(y) &\triangleq \{x \mid x\ \mathsf{HasType}_t\ y\} \\ \mathsf{Pop}_t(Y) &\triangleq \bigcup_{y \in Y} \mathsf{Pop}_t(y) \end{aligned}$$

The *extra temporal* versions are defined as:

$$\begin{aligned} x\,\mathsf{HasType}\,y &\triangleq \exists_{t\in\mathcal{TI}}\,[x\,\mathsf{HasType}_t\,y] \\ \mathsf{Types}(x) &\triangleq \bigcup_{t\in\mathcal{TI}} \mathsf{Types}_t(x) \\ \mathsf{Types}(X) &\triangleq \bigcup_{t\in\mathcal{TI}} \mathsf{Types}_t(X) \\ \mathsf{Pop}(y) &\triangleq \bigcup_{t\in\mathcal{TI}} \mathsf{Pop}_t(y) \\ \mathsf{Pop}(Y) &\triangleq \bigcup_{t\in\mathcal{TI}} \mathsf{Pop}_t(Y) \end{aligned}$$

Typing should adhere to the classification in our ontology. In other words:

**[S32]** For all $X \in \{\mathcal{FC}, \mathcal{RO}, \mathcal{PL}\}$ we have:

$$x\,\mathsf{HasType}\,y \Rightarrow (x \in X \Leftrightarrow y \in X)$$

The Fact and Player functions should never cross the type/instance level:

**[S33]** $\forall_{r\in\mathcal{RO}}\,[r \in \mathcal{TP} \Leftrightarrow \mathsf{Fact}(r) \in \mathcal{TP}]$

**[S34]** $\forall_{r\in\mathcal{RO}}\,[r \in \mathcal{TP} \Leftrightarrow \mathsf{Player}(r) \in \mathcal{TP}]$

All non-typing forms of decomposition should also not cross the type/instance level:

**[S35]** If $x\,\mathsf{DecompOf}\,y$, then:

$$x\,\mathsf{HasType}\,y \vee x, y \in \mathcal{TP} \vee x, y \in \mathcal{IN}$$

Players involved in role instances should behave as stipulated at the type level:

**[S36]** If $r \in \hat{\mathcal{RO}}$, then: $\mathsf{Player}(\mathsf{Pop}(r)) \subseteq \mathsf{Pop}(\mathsf{Player}(r))$

The same applies to facts:

**[S37]** If $r \in \hat{\mathcal{RO}}$, then: $\mathsf{Fact}(\mathsf{Pop}(r)) \subseteq \mathsf{Pop}(\mathsf{Fact}(r))$

Even more, as all role types of a fact type should be populated, the reverse should hold as well:

**[S38]** If $r \in \hat{\mathcal{RO}}$, then: $\mathsf{Fact}(\mathsf{Pop}(r)) \supseteq \mathsf{Pop}(\mathsf{Fact}(r))$

As an immediate result we have:

**Corollary 3.5.3**
If $f \in \hat{\mathcal{FC}}$, then:

$$\mathsf{Pop}(f) = \mathsf{Fact}(\mathsf{Pop}(\mathsf{RolesOf}(f)))$$

**Proof:**
Left as an exercise to the reader.

We can express this at the type level as:

**Corollary 3.5.4**
Let $f \in \check{\mathcal{FC}}$, then:

$$\mathsf{Types}(\mathsf{RolesOf}(f)) = \mathsf{RolesOf}(\mathsf{Types}(f))$$

Axiom **S38** does not have a pendent for players. In other words, we do not generally have:

$$\mathsf{Player}(\mathsf{Pop}(r)) \supseteq \mathsf{Pop}(\mathsf{Player}(r))$$

as this would require all instances of a player type to be involved in *all* roles in which the type is involved. However, we do have a weaker version as we will see in Section 3.6.

Facts should behave as a function from role types to instances:

**[S39]** Let $r \in \hat{\mathcal{RO}}$ and $s_1, s_2 \in \mathsf{Pop}(r)$, then:

$$\mathsf{Fact}(s_1) = \mathsf{Fact}(s_2) \Rightarrow s_1 = s_2$$

This axiom allows us, for any fact instance $f \in \check{\mathcal{FC}}$, to define the *partial* function $\vec{f} : \hat{\mathcal{RO}} \rightarrowtail \check{\mathcal{OB}}$ as:

$$\vec{f} \triangleq \{\langle r, \mathsf{Player}(s)\rangle \mid s \in \mathsf{Pop}(r) \wedge \mathsf{Fact}(s) = f\}$$

If $f$ is some fact and $R \subseteq \mathsf{RolesOf}(\mathsf{Types}(f))$, then we define the *total* function $\vec{f}[R] : R \rightarrow \check{\mathcal{OB}}$ as:

$$\vec{f}[R] \triangleq \{\langle r, \vec{f}(r)\rangle \mid r \in R\}$$

The ontology resulting after this refinement is depicted in Figure 3.5.

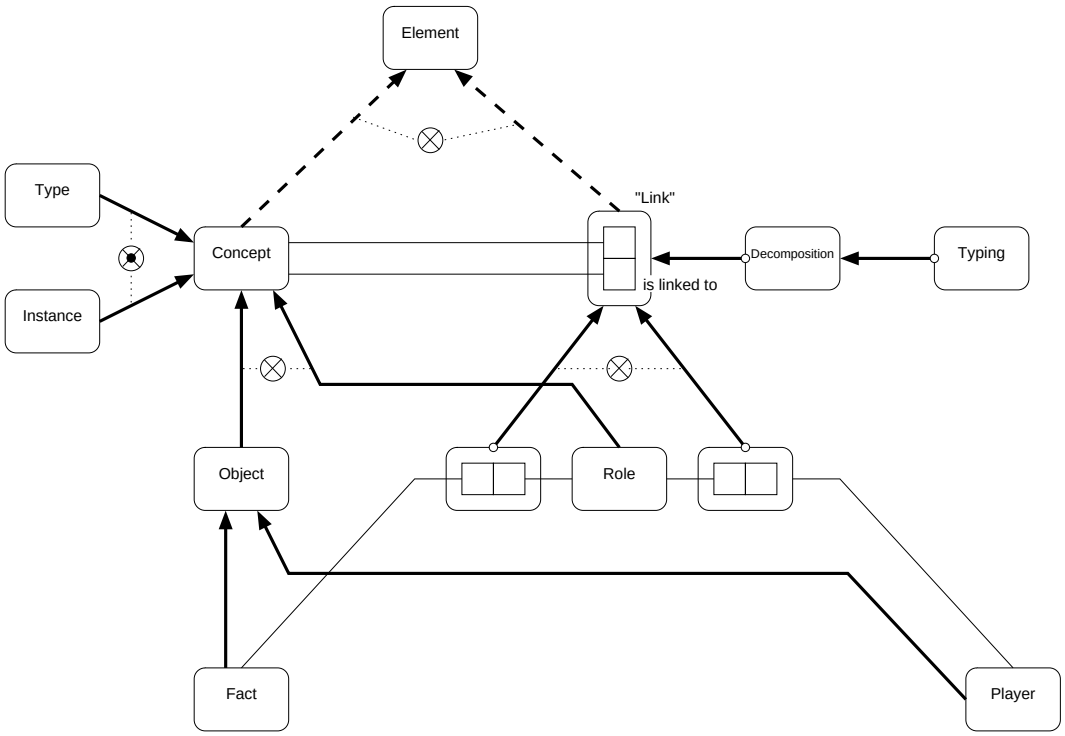



Figure 3.5: Our ontology refined with typing

## 3.6 Subtyping

Sub-typing is an important feature of object-role modeling. Figure 3.6 shows an example of sub-typing in terms of a specialization hierarchy.

In general, sub-typing involves the identification of a sub-set of the population of some super-type. For example, in the situation depicted in Figure 3.6 *flesh eater* is a specific sub-set of *animals*. In different versions of ORM, different rules apply to sub-typing [HW93, HP95, Hal01]. In this textbook we present a rather generic interpretation.

Formally, sub-typing is captured as a relationship $\mathsf{Sub} \subseteq \mathsf{LinkedTo}$ with intuition: if $x\,\mathsf{Sub}\,y$, then type $x$ is considered to be a subtype of $y$. We will also refer to $x$ as the *sub-type* and $y$ as the *super-type*. Sub-typing is a relationship over object types:

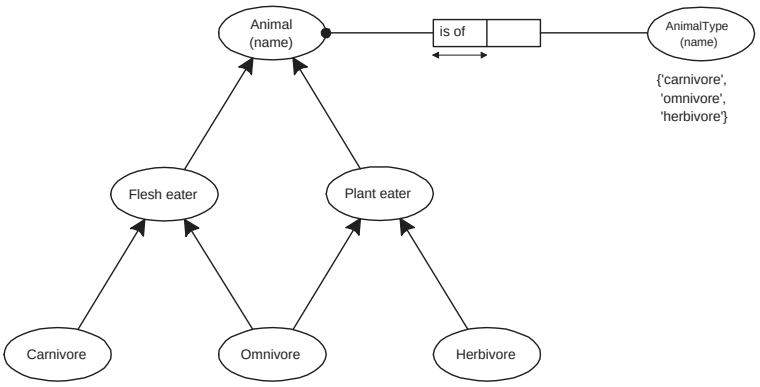



Figure 3.6: Example of a specialization hierarchy

**[S40]** $\mathsf{Sub} \subseteq \hat{\mathcal{OB}} \times \hat{\mathcal{OB}}$

The sub-typing relationship should be transitive and acyclic:

**[S41]** $x \,\mathsf{Sub}\, y \,\mathsf{Sub}\, z \Rightarrow x \,\mathsf{Sub}\, z$

**[S42]** $\neg x \,\mathsf{Sub}\, x$

The semantics of sub-typing in terms of populations is that the population of a specialized type should be a subset of the population of the supertype:

**[S43]** $x \,\mathsf{Sub}\, y \Rightarrow \mathsf{Pop}(x) \subseteq \mathsf{Pop}(y)$

We will also use as an abbreviation: $x \,\underline{\mathsf{Sub}}\, y \;\triangleq\; x \,\mathsf{Sub}\, y \vee x = y$.

The population of a subtype can be restrained further. In ORM, this is commonly done using a so-called sub-type defining rule. Using the language introduced in the next chapter, Section 5.1 will return to the issue of precisely defining the population of a subtype.

As promised, we would introduce a weaker pendant of Axiom **S38** for players. First, the set of roles types which are played by an object type as:

$$\mathsf{Plays}(p) \;\triangleq\; \{r \mid p \,\underline{\mathsf{Sub}}\, \mathsf{Player}(r)\}$$

This is actually the set of role-types that may be played by the instances of $x$, which we enforce by:

**[S44]** If $p \in \hat{\mathcal{PL}}$, then:

$$\mathsf{Pop}(p) \subseteq \mathsf{Player}(\mathsf{Pop}(\mathsf{Plays}(p)))$$

In other words, instances of a player type *must* be active in one of the associated roles. With Axiom **S36** we have:

**Corollary 3.6.1**
If $p \in \hat{\mathcal{PL}}$, then:

$$\mathsf{Pop}(p) = \mathsf{Player}(\mathsf{Pop}(\mathsf{Plays}(p)))$$

**Proof:**
Left as an exercise to the reader.

## 3.7 Overlap of populations

When considering the ORM schema as depicted in Figure 3.7, one would expect the populations of Vehicle and Distance to be disjoint, while the populations of Vehicle and Product are expected to overlap. Thus far, we have not introduced any formal mechanism to enforce this type of behavior other that the inclusion of populations for sub-types. To properly formalize this, we first define the family (as it is 'alive' at some point in time) of an object type based on the sub-typing hierarchy as follows:

$$\mathsf{Family}_t(x) \triangleq \left\{ y \in \hat{\mathcal{OB}}_t \mid y \,\mathsf{Sub}\, x \vee y = x \right\}$$

Two object types are deemed *type related* iff their families overlap:

$$x \sim_t y \triangleq \mathsf{Family}(x) \cap \mathsf{Family}(y) \neq \emptyset$$

If the population of object types overlaps, then they mus be type related:

**[S45]** $\mathsf{Pop}_t(x) \cap \mathsf{Pop}_t(y) \neq \emptyset \Rightarrow x \sim_t y$

Note that the converse does not necessarily hold.

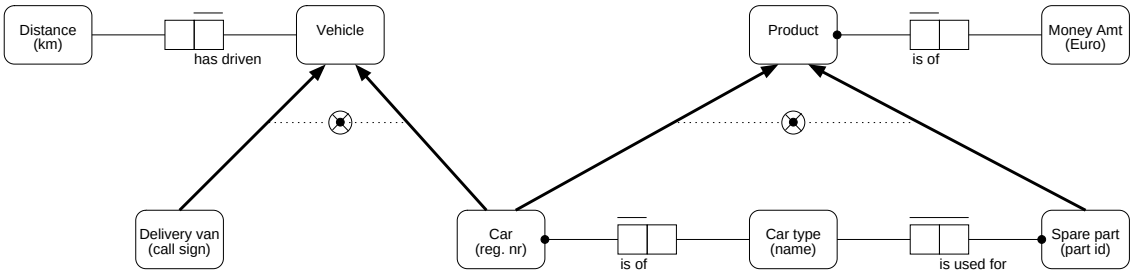



Figure 3.7: Example of a multi-rooted specialization hierarchy

# Questions

1. Given the situation:

   > A person with name Erik is writing a letter to his loved one, at the desk in a romantically lit room, on a mid-summer's day, using a pencil, while the cat is watching.

   Produce a graph consisting of concepts and links depicting this domain.
2. Stel je maakt een ontwerp voor een geldautomaat. Wat zijn voor dat domein de belangrijkste concepten en hun onderlinge links? Hoe werken ze samen?
3. Proof Corollary 3.4.1 (page 53).
4. Proof Corollary 3.5.1 (page 55).
5. Proof Corollary 3.5.2 (page 55).
6. Proof Corollary 3.5.3 (page 56).
7. Proof Corollary 3.5.4 (page 56).
8. Proof Corollary 3.6.1 (page 58).
9. Consider the following case:

Een onderneming produceert en verkoopt een tiental soorten gevulde chocolade-artikelen. De verkoop geschiedt aan grossiers tegen prijzen die voor lange tijd vast zijn. In verband met achteruitgang in kwaliteit wordt op de verpakking een uiterste verkoopdatum vermeld. Alle afleveringen geschieden met eigen auto's.

Voor de produktie van chocolade importeert de inkoopafdeling van de onderneming verschillende soorten cacaobonen uit tropische landen. Daartoe worden inkoopcontracten afgesloten die de behoefte voor ca. een half jaar dekken. De cacaobonenprijs is aan sterke schommelingen onderhevig. De ingekochte partijen hebben belangrijk uiteenlopende vetgehaltes, hetgeen mede in de inkoopprijs tot uitdrukking komt.

De cacaobonen ondergaan afzonderlijk per partij in de voorbewerkingsafdeling enkele machinale bewerkingen, zoals zuiveren, schillen, breken, branden, malen en walsen.

Aan het onstane halffabrikaat worden door de afwerkingsafdeling suiker, smaakstoffen en – in verhouding tot het vetgehalte – cacaoboter toegevoegd. Het aldus verkregen halffabrikaat is cacaomassa van een bepaalde standaardkwaliteit, dat in speciaal daartoe geconditioneerde opslagtanks wordt bewaard. De verschillende benodigde vulsels worden ingekocht bij derden. Naar rato van de ontwikkeling van de verkoop en de gewenste voorraadvorming worden de eindprodukten gemaakt. Dit geschiedt in één arbeidsgang met behulp van automatische vorm-, vul- en droogmachines.

In de pakafdeling worden de goedkopere soorten gevulde chocolade automatisch en de duurdere soorten met de hand in sierdozen verpakt, waarna opslag in een magazijn volgt. Bij alle bewerkingen ontstaan gewichtsverliezen.

In verband met de kwaliteitsachteruitgang kunnen de grossiers de niet tijdig door hen verkochte artikelen retourneren, mits dit gebeurt binnen 10 dagen na de uiterste verkoopdatum; meestal geschiedt deze teruglevering via de chauffeurs. De teruggenomen artikelen worden vernietigd. Creditering vindt plaats voor 20 van de door hen betaalde prijs. Verrekening hiervan geschiedt slechts bij gelijktijdige nieuwe afname.

Elk van de artikelen is voorzien van een of twee cadeaubonnen, afgedrukt op de verpakking. De waarde van deze bonnen is €0,10 per stuk. Op de artikelen met een prijs tot €5,- komt één, op de overige artikelen (tussen €5,- en €11,-) komen twee bonnen voor. Op deze bonnen kunnen cadeau-artikelen (hand- en theedoeken e.d.) zonder bijbetaling worden verkregen.

Voorts kunnen op deze bonnen meer duurzame gebruiksgoederen tegen verlaagde prijs worden verkregen. Hiervoor wordt elk halfjaar een folder uitgegeven, waarin per artikel is aangegeven hoeveel bonnen moeten worden ingeleverd en hoeveel daarnaast moet worden bijbetaald. In het algemeen is het door de afnemers bij te betalen bedrag iets lager dan de inkoopprijs voor de fabriek. Veelal dient de halfjaarlijkse behoefte door de fabrikant in één keer te worden besteld; latere aanvulling is in het algemeen niet mogelijk.

Op de duurzame gebruiksgoederen wordt veelal garantie of service verleend. Hiervoor is met een gespecialiseerd bedrijf een contract afgesloten waarbij tegen een eenmalig vast bedrag per apparaat de garantie- en serviceverplichtingen worden overgedragen

Answer the following questions:

(a) Produce elementary facts for this domain.

(b) Produce an ORM model for this domain.

# Bibliography

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modelling Language User Guide*. Addison Wesley, Reading, Massachusetts, USA, 1999. ISBN 0201571684

[EKW92] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object–Oriented Systems Analysis – A model–driven approach*. Yourdon Press, New York, New York, USA, 1992. ASIN 0136299733

[Hal01] T.A. Halpin. *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Mateo, California, USA, 2001. ISBN 1558606726

[HP95] T.A. Halpin and H.A. (Erik) Proper. Subtyping and Polymorphism in Object–Role Modelling. *Data & Knowledge Engineering*, 15:251–281, 1995.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[Kri94] G. Kristen. *Object Orientation – The KISS Method, From Information Architecture to Information System*. Addison Wesley, Reading, Massachusetts, USA, 1994. ISBN 0201422999

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice–Hall, Englewood Cliffs, New Jersey, USA, 1989. ASIN 0131672630

[RMD99] V.E. van Reijswoud, J.B.F Mulder, and J.L.G. Dietz. Commucation Action Based Business Process and Information Modelling with DEMO. *The Information Systems Journal*, 9(2):117–138, 1999.

[Win90] J.J.V.R. Wintraecken. *The NIAM Information Analysis Method: Theory and Practice*. Kluwer, Deventer, The Netherlands, EU, 1990.

# Chapter 4

# Object-Role Calculus

Version:
10-10-05

## 4.1 Introduction

This chapter is concerned with a language to express rules over domains that are modeled by means of an ORM model. An ORM model is typically produced in a natural-language driven process. As a side-product, the fact types in an ORM model receive verbalizations that closely match natural language. In a rule language we would like to be able to re-use these verbalizations to arrive at rules that also closely resemble natural language. This resemblance will allow different stakeholders to understand (and validate) the rules.

In this chapter, we therefore propose a fact-based approach to the formulation of rules, and reasoning with these rules, which is based entirely on concepts that are familiar to a domain expert, as modeled through ORM. In doing so, we will built on conceptual rule languages such as Lisa-D [HPW93], RIDL [Mee82] and ConQuer [Pro94b, BH96]. These languages allow for the specification of rules in a semi-natural language format that can be more easily understood by domain experts than languages such as predicate calculus, Z [Spi88] or OCL [WK03].

The language we introduce in this chapter will be referred to as the *object role calculus* (ORC), a name which clearly signifies its relation to *object role modeling* and the fact that it is a language in which one cannot only express rules but also reason/calculate with these rules analogously to *predicate calculus*.

Note: In the current version of this chapter, we do not yet include reasoning.

The ORC is built up in four layers:

**Computational layer –** Instances from the population of an ORM model will occur in the result of a rule with a certain frequency. This frequency may be binary: *Does a specific instance occur: yes/no?*, but it may also be natural number: *How many times does a specific instance occur?* Even more, one may associate uncertainties to these frequencies.

**Logic layer –** The ORC is built on top of a multi-valued logic (using the underlying computational layer), where the computational domain is used as (one of) the underlying domains. Theoretically, this could be any kind of logic, but for practical purposes we will use traditional predicate calculus with some minor temporal extensions. This could be replaced by more advanced modal logics.

**Path-expression layer –** In its most basic form, rules formulated in ORC correspond to paths of fact-types strung together using connectors. At this layer we are concerned with the construction of these paths. Path expressions form the bones of ORC rules, while the logic layer provides the atoms from the these bones are constructed.

**Information-descriptor layer –** Path expressions still not utilize the rich verbalizations that are typical for ORM models. In the last layer we finally add skin to the bare bones in terms of natural looking expressions that can be understood by a wide(r) audience.

## 4.2 Computational domain

Let $\mathcal{FR}$ be a set of quantities. In a concrete case this could be boolean, natural numbers, probabilities, etc. These frequencies are best thought of as stating the frequency at which an element may occur in a set of results.

We need six operations to combine quantities:

$$\vee, \wedge, \oplus, \otimes, \ominus : \mathcal{FR} \times \mathcal{FR} \rightarrow \mathcal{FR}$$

and two constants:

$$1, 0 \in \mathcal{FR}$$

These operations have the following intended meaning:

- $a \vee b$ is the resulting frequency when counting elements occuring $a$ times *or* $b$ times.
- $a \wedge b$ is the resulting frequency when counting elements occuring $a$ times *and* $b$ times.
- $a \oplus b$ is the resulting frequency of when adding an element occuring $a$ times to an element occuring $b$ times.
- $a \otimes b$ is the resulting frequency of a cartesian combination of an element occuring $a$ times and $b$ times.
- $a \ominus b$ is the resulting frequency when doing a set exclusion on an element occuring $a$ times and $b$ times.

The $\vee$, $\wedge$, $\oplus$ and $\otimes$ operations should be cummutative and associative. Let $\Theta$ be one of these operations, then:

$$\begin{aligned} a \,\Theta\, b &= b \,\Theta\, a \\ a \,\Theta (b \,\Theta\, c) &= (a \,\Theta\, b) \,\Theta\, c \end{aligned}$$

The $1$ and $0$ constants should behave as neutral elements:

$$\begin{gathered} a \otimes 0 = 0 \\ a \wedge 0 = 0 \\ a \vee 0 = a \\ a \oplus 0 = a \\ a \otimes 1 = a \\ a \wedge 1 = a \end{gathered}$$

If $X$ is some set, then $\|X\| = \bigoplus_{x \in X} 1$.

This leaves us with the task of defining these operations and constants for specific computational domains.

For normal logic we would have:

$$\begin{aligned} \mathcal{FR}_{\mathbb{B}} &\triangleq \mathbb{B} \\ a \vee b &\triangleq \mathsf{Max}(a, b) \\ a \wedge b &\triangleq \mathsf{Min}(a, b) \\ a \oplus b &\triangleq \mathsf{Max}(a, b) \\ a \otimes b &\triangleq \mathsf{Min}(a, b) \\ a \ominus b &\triangleq \mathsf{Max}(a - b, 0) \\ 1 &\triangleq 1 \\ 0 &\triangleq 0 \end{aligned}$$

Note that in this case, $\oplus$ and $\otimes$ have the samen semantics as $\vee$ and $\wedge$ respectively.

For multi-sets we have:

$$\begin{array}{rcl} \mathcal{FR}_{\mathbb{N}} & \triangleq & \mathbb{N} \\ a \vee b & \triangleq & \mathsf{Max}(a,b) \\ a \wedge b & \triangleq & \mathsf{Min}(a,b) \\ a \oplus b & \triangleq & a+b \\ a \otimes b & \triangleq & a \times b \\ a \ominus b & \triangleq & \mathsf{Max}(a-b,0) \\ 1 & \triangleq & 1 \\ 0 & \triangleq & 0 \end{array}$$

We might define generalized frequency sets as:

$$\begin{array}{rcl} \mathcal{FR}_{\mathbb{Z}} & \triangleq & \mathbb{Z} \\ a \vee b & \triangleq & \mathsf{Max}(a,b) \\ a \wedge b & \triangleq & \mathsf{Min}(a,b) \\ a \oplus b & \triangleq & a+b \\ a \otimes b & \triangleq & a \times b \\ a \ominus b & \triangleq & a-b \\ 1 & \triangleq & 1 \\ 0 & \triangleq & 0 \end{array}$$

Let $\mathcal{FR}_F^d$ be one of the three above computational domains for frequencies, then we can define a probabilistic distribution version over $\mathcal{FR}$ as follows:

$$\begin{array}{rcl} \mathcal{FR}_F^d & \triangleq & \mathcal{FR}_F \rightarrow [0..1] \\ a \,\Theta\, b & \triangleq & \lambda n.1 - (\Pi_{i,j \in \mathbb{N}: i \,\Theta\, j = n}(1 - a(i) \times b(j))) \end{array}$$

where $\Theta$ is any of $\vee$, $\wedge$, $\oplus$, $\otimes$ and $\ominus$.

## 4.3 Logic layer

If $\phi$ is a well-formed formula, we will use $\mathbb{V}[\![\ \phi\ ]\!](\mathsf{Pop}, t)$ to express the valuation of $\phi$, where we will also write:

$$\mathsf{Pop} \models_t^v \phi \ \text{ iff } \ v = \mathbb{V}[\![\ \phi\ ]\!](\mathsf{Pop}, t)$$

Given an ORM model with role type $r \in \hat{\mathcal{RO}}$, object types $o \in \hat{\mathcal{OB}}$, instances $i, j \in \check{\mathcal{OB}}$ and population Pop, we have the following atomic constructions:

$$\begin{array}{rcl} \mathbb{V}[\![\ o(i)\ ]\!](\mathsf{Pop}, t) & \triangleq & \left\| \left\{ x \in \mathsf{Pop}_t(o) \ \middle|\ x = i \right\} \right\| \\ \mathbb{V}[\![\ r(i,j)\ ]\!](\mathsf{Pop}, t) & \triangleq & \left\| \left\{ x \in \mathsf{Pop}_t(r) \ \middle|\ i = \mathsf{Player}(x) \wedge j = \mathsf{Fact}(x) \right\} \right\| \end{array}$$

If $\Theta$ is one of the operators $\vee$, $\wedge$, $\oplus$, $\otimes$ and $\ominus$, then we have:

$$\begin{array}{rcl} \mathbb{V}[\![\ \phi \,\Theta\, \psi\ ]\!](\mathsf{Pop}, t) & \triangleq & \mathbb{V}[\![\ \phi\ ]\!](\mathsf{Pop}, t) \,\Theta\, \mathbb{V}[\![\ \psi\ ]\!](\mathsf{Pop}, t) \\ \mathbb{V}[\![\ \neg\phi\ ]\!](\mathsf{Pop}, t) & \triangleq & 1 \ominus \mathbb{V}[\![\ \phi\ ]\!](\mathsf{Pop}, t) \\ \mathbb{V}[\![\ \forall_x [\phi]\ ]\!](\mathsf{Pop}, t) & \triangleq & \bigwedge_c \mathbb{V}[\![\ \phi\,\langle x/c\rangle\ ]\!](\mathsf{Pop}, t) \\ \mathbb{V}[\![\ \exists_x [\phi]\ ]\!](\mathsf{Pop}, t) & \triangleq & \bigvee_c \mathbb{V}[\![\ \phi\,\langle x/v\rangle\ ]\!](\mathsf{Pop}, t) \end{array}$$

Using the traditional logic operators, path expressions would only be able to deal with specific points in time only. Since we take the evolution of a domain (conception) into account we need rules that refer to time as well. We therefore also introduce the following constructions into the logical layer:

$$\begin{aligned} \mathbb{V}[\![\ \mathsf{always}\,\phi\ ]\!](\mathsf{Pop}, t) &\triangleq \bigwedge_{s \in \mathcal{TI}} \mathbb{V}[\![\ \phi\ ]\!](\mathsf{Pop}, s) \\ \mathbb{V}[\![\ \mathsf{next}\,\phi\ ]\!](\mathsf{Pop}, t) &\triangleq \mathbb{V}[\![\ \phi\ ]\!](\mathsf{Pop}, \rhd t) \end{aligned}$$

In addition we introduce the following abreviations:

$$\begin{aligned} \mathsf{sometime}\, X &\triangleq \neg\, \mathsf{always}\, \neg X \\ X\ \mathsf{precedes}\, Y &\triangleq \mathsf{always}((X \wedge \mathsf{next}\, Y) \Rightarrow (\mathsf{next}\, \neg X \wedge \neg Y)) \end{aligned}$$

## 4.4 Path expression layer

Path expressions are essentially paths through an ORM model. As a path has a head and a tail, path expressions can be regarded as binary predicates in a logic, where the head and tail are the first two parameters. However, we will not only maintain heads and tails, but also the frequencies in which a path from head to tail is available. This leads to the view that a path expression is a ternary predicate. Even more, to enable us to express such rules as:

> *People who work for a department must be involved in a project conducted by that department, where the project is managed by another person.*

we need to be able to introduce additional variables. So, in general, path expressions need to have an arity of $n + 3$.

The semantics of path expressions is defined by means of a set of re-write rules, in the style of denotational semantics [Sto77]. If $P$ is a path expression, each rule will aim to rewrite $P$ in terms of logic expressions, breaking down $P$ into its constituent elements. Formally we will write $x[\![\ P\ ]\!]y$ at the left hand side of each of these rewrite rules. The $x$ and $y$ are variables representing the head and tail of a path expression, while $t$ is a sequence representing the actual path of instances from $x$ to $y$ combination. In addition to $x$ and $y$, the path expression $P$ can contain other variables. These will be drawn from a special set $\omega$. Let, in the remainder of this chapter, $x, y, z$ be variables that are not in $\omega$.

### 4.4.1 Atomic path expressions

At this point we can describe the meaning of elementary path expressions as follows. Let $o$ be an object type and $r$ a role type, then $o$ and $r$ are (atomic) path expressions with semantics:

$$\begin{aligned} x[\![\ o\ ]\!]y &\triangleq o(x) \wedge x = y \\ x[\![\ r\ ]\!]y &\triangleq r(x, y) \end{aligned}$$

If $c$ is a constant and $v \in \omega$ is some variable, then we also have:

$$\begin{aligned} x[\![\ c\ ]\!]y &\triangleq c = x \wedge x = y \\ x[\![\ v\ ]\!]y &\triangleq v = x \wedge x = y \end{aligned}$$

As special atomic path expressions we introduce:

$$\begin{array}{rcl} x[\![\ 1\ ]\!]y & \triangleq & x = y \\ x[\![\ 0\ ]\!]y & \triangleq & 0 \\ x[\![\ \infty\ ]\!]y & \triangleq & true \end{array}$$

As an example, consider the ORM model depicted in Figure 4.1.

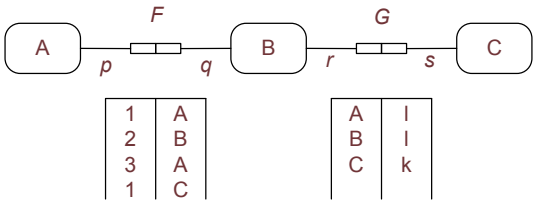



Figure 4.1: Basic ORM model

In this case we would have for $x[\![\ p\ ]\!]y$, $x[\![\ q\ ]\!]x$ and $x[\![\ B\ ]\!]y$:

| $x$ | $y$ | frequency |
|---|---|---|
| 1 | $\langle 1, A\rangle$ | 1 |
| 2 | $\langle 2, B\rangle$ | 1 |
| 3 | $\langle 3, A\rangle$ | 1 |
| 1 | $\langle 1, C\rangle$ | 1 |

| $x$ | $y$ | frequency |
|---|---|---|
| $\langle 1, A\rangle$ | 1 | 1 |
| $\langle 2, B\rangle$ | 2 | 1 |
| $\langle 3, A\rangle$ | 3 | 1 |
| $\langle 1, C\rangle$ | 1 | 1 |

| $x$ | $y$ | frequency |
|---|---|---|
| $A$ | $A$ | 1 |
| $B$ | $B$ | 1 |
| $C$ | $C$ | 1 |

### 4.4.2 Composing paths

If $P$ and $Q$ are path expressions, then we can concatenate path expressions as follows:

$$x[\![\ P \circ Q\ ]\!]y \quad \triangleq \quad \exists_z\,[x[\![\ P\ ]\!]z \otimes z[\![\ P\ ]\!]y]$$

We immediately have:

**Corollary 4.4.1**
$P \circ 1 = P$ and $P \circ 0 = 0$.

For the manipulation of heads and tails we introduce:

$$\begin{array}{rcl} x[\![\ P^{\leftarrow}\ ]\!]y & \triangleq & y[\![\ P\ ]\!]x \\ x[\![\ \mathsf{hd}\, P\ ]\!]y & \triangleq & x = y \wedge \exists_z\,[x[\![\ P\ ]\!]z] \\ x[\![\ \mathsf{tl}\, P\ ]\!]y & \triangleq & x = y \wedge \exists_z\,[z[\![\ P\ ]\!]y] \end{array}$$

where we use as an abbreviation: $x[\![\ P\ ]\!]y \ \triangleq\ \exists_a\,[x[\![\ P\ ]\!]y]$.

As an example, in the case of Figure 4.1 we would have for $x[\![\ p \circ q^{\leftarrow}\ ]\!]y$, $x[\![\ p \circ q^{\leftarrow} \circ r \circ s^{\leftarrow}\ ]\!]y$ and $x[\![\ p \circ q^{\leftarrow} \circ v \circ r \circ s^{\leftarrow}\ ]\!]y$ (where $v$ is a variable):

| $x$ | $y$ | frequency |
|---|---|---|
| 1 | $A$ | 1 |
| 2 | $B$ | 1 |
| 3 | $A$ | 1 |
| 1 | $C$ | 1 |

| $x$ | $y$ | frequency |
|---|---|---|
| 1 | $l$ | 2 |
| 2 | $k$ | 1 |
| 3 | $l$ | 1 |
| 1 | $k$ | 1 |

A special construction operator for path expression is the *confluence* operator. This operator is essential in formalizing graphical uniqueness constraints. Let $P_1, \ldots P_n$ be path expressions, then:

$$x[\![\ \langle P_1, \ldots, P_n \rangle\ ]\!]y \quad \triangleq \quad \exists_{x_1,\ldots,x_n}\,[x_1[\![\ P_1\ ]\!]y \wedge \ldots \wedge x_n[\![\ P_n\ ]\!]y \wedge x = \langle x_1, \ldots, x_n \rangle]$$

By combining this operator with $\infty$, we can also construct cartesian products:

$$P_1 \times \ldots \times P_n \quad \triangleq \quad \langle P_1 \circ \infty, \ldots, P_n \circ \infty \rangle$$

In addition to path composition operators, we also have the following ways of combining path expressions. If $\Theta$ is one of the operators $\vee$, $\wedge$, $\oplus$, $\otimes$ and $\ominus$, then we have:

$$\begin{array}{rcl} x[\![\ P \,\Theta\, Q\ ]\!]y & \triangleq & x[\![\ P\ ]\!]y \,\Theta\, x[\![\ P\ ]\!]y \\ x[\![\ \neg P\ ]\!]y & \triangleq & \neg(x[\![\ P\ ]\!]y) \end{array}$$

Note that these logical connectives 'reset' the paths between heads and tails of instances. Implication and equality of path expressions can be defined as:

$$\begin{array}{rcl} P \Rightarrow Q & \triangleq & \neg P \vee Q \\ P \Leftrightarrow Q & \triangleq & (P \Rightarrow Q) \wedge (Q \Rightarrow P) \end{array}$$

For practical purposes, however, we would like to use logical connectives that allow us to combine the heads of paths. For example, in an expressions such as:

*Person working for department 'IRIS', who also owns a car.*

In this case we would like to require:

$$\exists_y\,[x[\![\ \text{Person working for Department 'IRIS'}\ ]\!]y] \wedge \exists_y\,[x[\![\ \text{Person owning a Car}\ ]\!]y]$$

To be able to expres rules like these, we also introduce the following abbreviations:

$$\begin{array}{rcl} P \overset{\mathsf{h}}{\wedge} Q & \triangleq & \mathsf{hd}\, P \wedge \mathsf{hd}\, Q \\ P \overset{\mathsf{h}}{\vee} Q & \triangleq & \mathsf{hd}\, P \vee \mathsf{hd}\, Q \\ P \overset{\mathsf{h}}{\oplus} Q & \triangleq & \mathsf{hd}\, P \oplus \mathsf{hd}\, Q \\ P \overset{\mathsf{h}}{\ominus} Q & \triangleq & \mathsf{hd}\, P \ominus \mathsf{hd}\, Q \\ P \overset{\mathsf{h}}{\Longrightarrow} Q & \triangleq & \mathsf{hd}\, P \Rightarrow \mathsf{hd}\, Q \\ P \overset{\mathsf{h}}{\Longleftrightarrow} Q & \triangleq & \mathsf{hd}\, P \Leftrightarrow \mathsf{hd}\, Q \end{array}$$

### 4.4.3 Evolution and path expressions

To allow for path expressions to refer to evolution of the population, we introduce:

$$\begin{array}{rcl} x[\![\ \mathsf{sometime}\, P\ ]\!]y & \triangleq & \mathsf{sometime}(x[\![\ P\ ]\!]y) \\ x[\![\ \mathsf{always}\, P\ ]\!]y & \triangleq & \mathsf{always}(x[\![\ P\ ]\!]y) \\ x[\![\ P\ \mathsf{precedes}\, Q\ ]\!]y & \triangleq & (x[\![\ P\ ]\!]y)\ \mathsf{precedes}(x[\![\ Q\ ]\!]y) \end{array}$$

### 4.4.4 Path expressions as logic

If $P$ is a path expression, then we have the following logic expressions:

$$\begin{aligned} \exists[\![P]\!] &\triangleq \exists_{x,y}\,[x[\![\ P\ ]\!]y] \\ \forall[\![P]\!] &\triangleq \forall_{x,y}\,[x[\![\ P\ ]\!]y] \end{aligned}$$

## 4.5 Graphical constraints

In this section we briefly visit some of the graphical constraints that are used in ORM models.

### 4.5.1 Mandatory roles

If $o$ is an object type and $r_1, \ldots, r_n$ are roles types such that $\forall_{1\leq i\leq n}\,[o\,\underline{\mathsf{Sub}}\,\mathsf{Player}(r_i)]$, then:

$$\mathsf{Total}(o : \{r_1, \ldots, r_n\}) \quad \triangleq \quad \mathsf{always}\,\forall[\![o \overset{\mathsf{h}}{\Longrightarrow} (r_1 \overset{\mathsf{h}}{\Diamond} \ldots \overset{\mathsf{h}}{\Diamond} r_n)]\!]$$

We may even further generalize this to mandatory combination over cartesian products of object types. Let $o_1, \ldots, o_m$ be object types and $r_{1,1}, \ldots, r_{n,1}, \ldots, \ldots, r_{1,m}, \ldots, r_{n,m}$ are role types such that: $\forall_{1\leq i,j\leq m}\,\forall_{1\leq k\leq n}\,[o_i\,\underline{\mathsf{Sub}}\,\mathsf{Player}(r_{k,i}) \wedge \mathsf{Fact}(r_{k,i}) = \mathsf{Fact})r_{k,j}]$, then:

$$\begin{aligned} &\mathsf{Total}(\langle o_1, \ldots, o_m\rangle : \{\langle r_{1,1}, \ldots, r_{1,m}\rangle, \ldots, \langle r_{n,1}, \ldots, r_{n,m}\rangle\}) \ \triangleq \\ &\quad \mathsf{always}\,\forall[\![(o_1 \times \ldots \times o_m) \overset{\mathsf{h}}{\Longrightarrow} (\langle r_{1,1}, \ldots, r_{1,m}\rangle \overset{\mathsf{h}}{\Diamond} \ldots \overset{\mathsf{h}}{\Diamond} \langle r_{n,1}, \ldots, r_{n,m}\rangle)]\!] \end{aligned}$$

### 4.5.2 Uniqueness

Given a set of role types, we may be able to automatically determine a derived fact type that joins all fact types involved in these role types. The algorithm to compute the actual derivation is referred to as the *uniquest* algorithm [WHB92]. It was designed, initially, to provide the semantics of graphical uniquess constraints as depicted in Figure 4.2. In the case of roles $\mathsf{r}$ and $\mathsf{s}$, the result would be: $\langle \mathsf{r}, \mathsf{s}\rangle$, while in the case of roles $\mathsf{q}$ and $\mathsf{s}$ the result would be: $\langle \mathsf{q} \circ \mathsf{r}, \mathsf{s}\rangle$. Note that not all combinations of role types lead to join paths. In [WHB92] this is discussed in more detail.

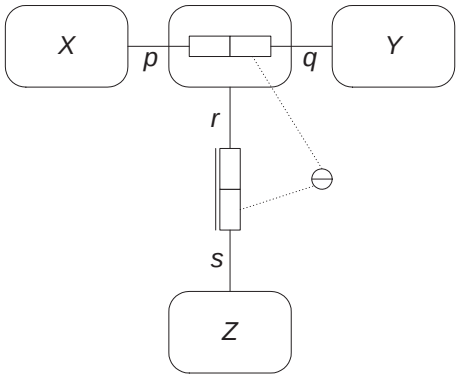



Figure 4.2: Example uniqueness constraints

Let $\mathsf{JoinPath}(R)$ be the function determining a join path for some sequence of roles types $R$, while $o$ is the central object type, using the uniques algorithm. If $\mathsf{JoinPath}(R)$ is defined for some sequence $R$ of role types, then the uniqueness of the roles types in $R$ is defined as:

$$\mathsf{Unique}(R) \quad \triangleq \quad \mathsf{always}\,\forall[\![\mathsf{JoinPath}(R) \circ \mathsf{JoinPath}(R)^{\leftarrow} \Rightarrow 1]\!]$$

### 4.5.3 Subsets

The JoinPath function cannot only be used to define uniqueness constraints, but also to define subset constraints. If $R_1$ and $R_2$ are two sequences of role types with the same length (i.e. $|R_1| = |R_2|$) such that the players are type related: $\forall_{1 \leq i \leq |R_1|}\,[R_1[i] \sim R_2[i]]$, then a subset constraint from $R_1$ to $R_2$ is defined as:

$$\mathsf{SubSet}(R_1, R_2) \quad \triangleq \quad \mathsf{always}\,\forall[\![\mathsf{JoinPath}(R_1) \overset{\mathsf{h}}{\Longrightarrow} \mathsf{JoinPath}(R_2)]\!]$$

As an example, consider the model depicted in Figure 4.3. In this case we would have:

$$\mathsf{always}\,\forall[\![\langle r, s\rangle \overset{\mathsf{h}}{\Longrightarrow} \langle p, q\rangle]\!]$$

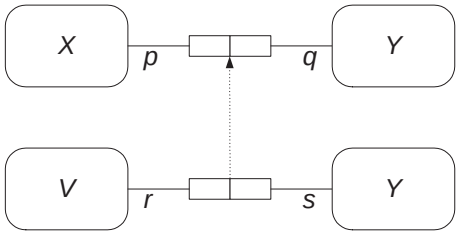



Figure 4.3: Example subset constraint

### 4.5.4 Temporal ordering

Now consider the following verbalizations (at the type level):

A Person fills in a Form
A Person is examined by a Doctor
A Doctor produces a Diagnose
A Doctor writes a Prescription

This leads to the situation as depicted in Figure 4.4.

Thus far we have not discussed properties pertaining to temporal ordering. Suppose now that in this domain:

Before a Person can be examined by a Doctor, they should have filled in a Form.
Before a Doctor produces a Diagnose, a Person should have been Examined.
Before a Doctor writes a Prescription, a Person should have been Diagnosed.

This is, however, is still an incomplete picture. The production of a diagnose and the writing of a prescription should all pertain to the same person. Even more, as a person may visit a doctor twice for two different reasons, the diagnose and prescription really pertain to one specific doctor visit. This leads to the situation as depicted in Figure 4.5.

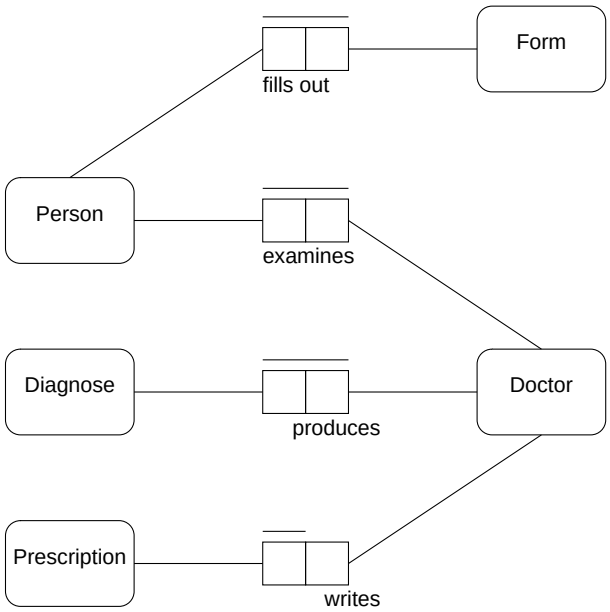



Figure 4.4: Basic model of a visit to a Doctor

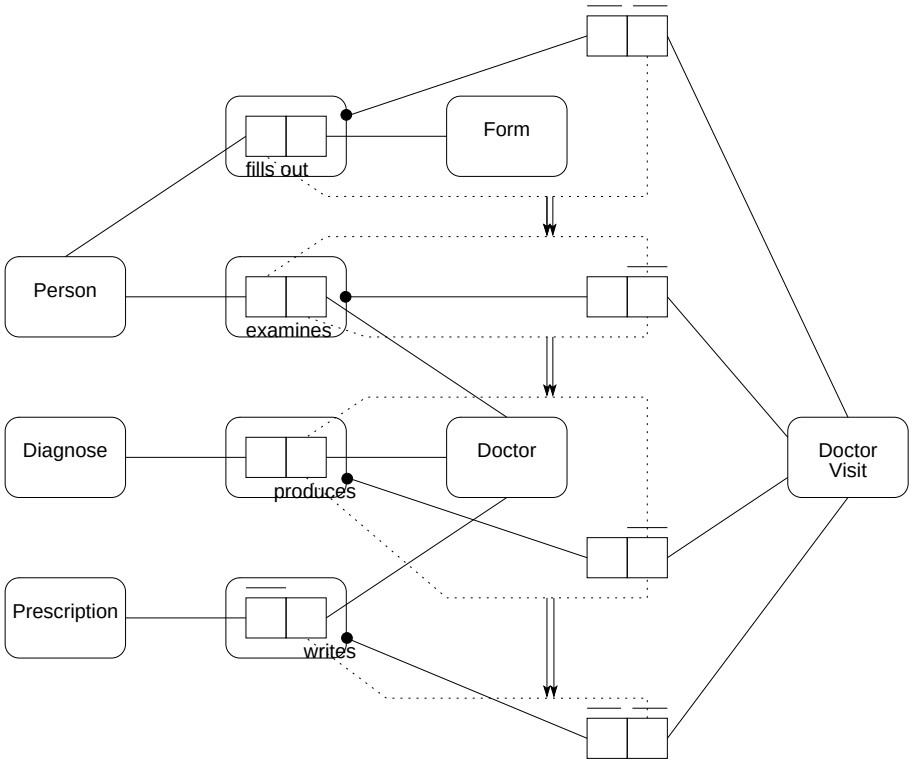



Figure 4.5: Model of a visit to a Doctor with explicit entity

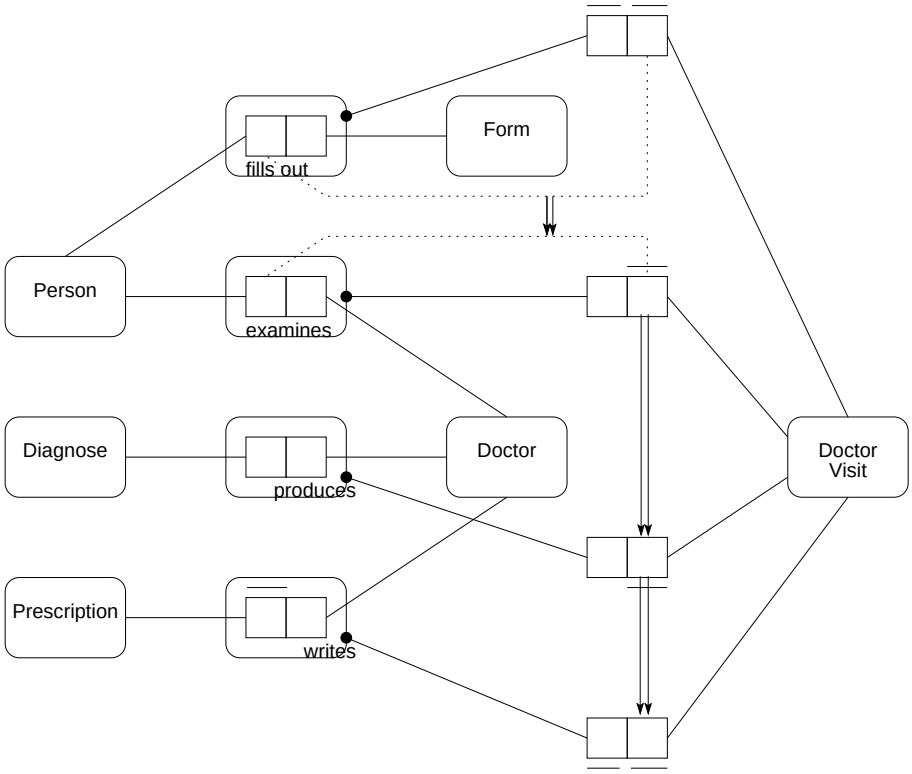



Figure 4.6: Model of a visit to a Doctor with alternative semantics

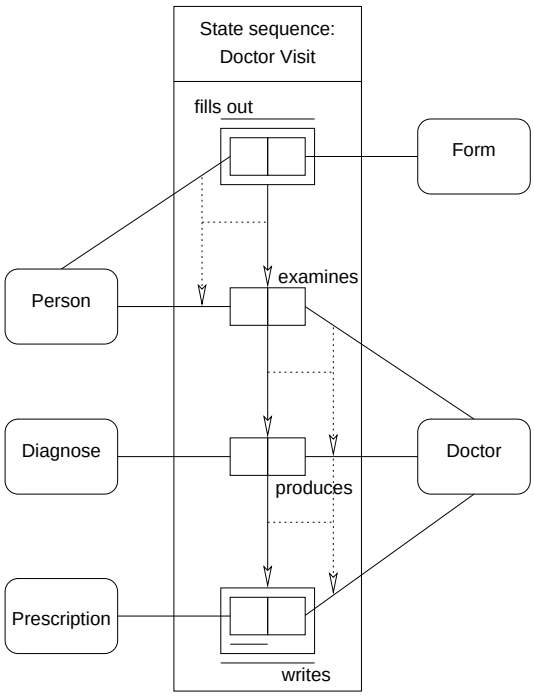



Figure 4.7: Compact model of a visit to a Doctor

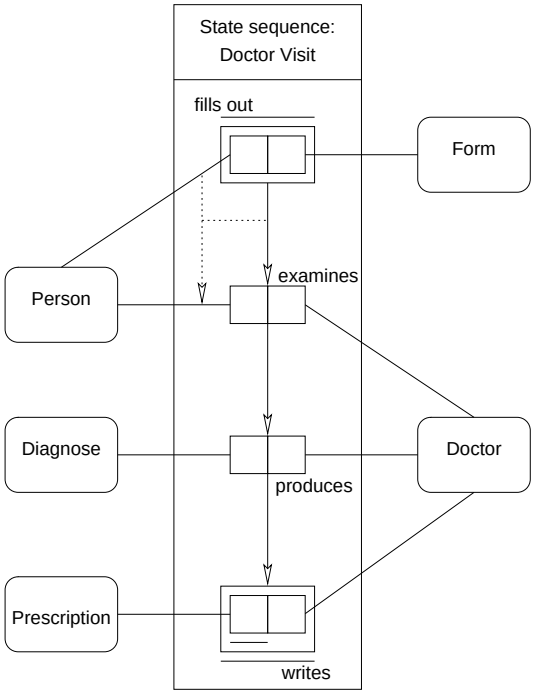



Figure 4.8: Compact model of a visit to a Doctor with alternative semantics

But also consider the situation as depicted in Figure 4.6. What is the semantic difference?

As a graphical abbreviation, we will use the notation as depicted in Figure 4.7 and Figure 4.8 respectively.

Formally, temporal dependency constraints can be defined as follows:

$$\mathsf{precedes}(R_1, R_2) \triangleq \mathsf{always}\,\forall[\![\mathsf{JoinPath}(R_1)\,\mathsf{precedes}\,\mathsf{JoinPath}(R_2)]\!]$$

## 4.6 Information-descriptor layer

### 4.6.1 Naming of types

To each object type, a name is associated: $\mathsf{ONm} : \hat{\mathcal{OB}} \rightarrow \mathcal{NM}$. To each role type, we can actually associated two names, a normal role name, and a so-called reverse role name: $\mathsf{RNm} : \hat{\mathcal{RO}} \rightarrow \mathcal{NM}$ and $\mathsf{RRNm} : \hat{\mathcal{RO}} \rightarrow \mathcal{NM}$.

In terms of Figure 4.1, we might have:

| | | |
|---|---|---|
| $\mathsf{ONm}(A) = \mathsf{Person}$ | $\mathsf{RNm}(p) = \mathsf{is\ a\ coworker\ in}$ | $\mathsf{RRNm}(p) = \mathsf{has\ coworker}$ |
| $\mathsf{ONm}(B) = \mathsf{Department}$ | $\mathsf{RNm}(q) = \mathsf{is\ employer\ in}$ | $\mathsf{RRNm}(q) = \mathsf{has\ employer}$ |
| $\mathsf{ONm}(F) = \mathsf{Department\ coworkership}$ | | |

When verbalizing facts, one will typically use verbalizations such as:

Person working for Department

The working for phrase is not the name of a role type, but really referring to the combination of two role types. For example, in the case of Figure 4.1, this might be $p \circ q^{\leftarrow}$. Using the function: $\mathsf{CNm} : \hat{\mathcal{RO}} \times \hat{\mathcal{RO}} \rightarrow \mathcal{NM}$ we associate a name to such role pairs. For example: $\mathsf{CNm}(p, q) = \mathsf{working\ for}$.

### 4.6.2 Basic information descriptors

Information descriptors can now be defined by providing their mapping to path expressions. Let $o$ be an object type with $\mathsf{ONm}(o) = n$, then we have the following atomic information descriptors:

$$\mathbb{D}[\![\, n \,]\!] \quad \triangleq \quad o$$

Also the role names lead to atomic information descriptors. Let $p$ a role type with $\mathsf{RNm}(p) = n$ and $\mathsf{RRNm}(p) = m$, then:

$$\begin{aligned} \mathbb{D}[\![\, n \,]\!] &\triangleq p \\ \mathbb{D}[\![\, m \,]\!] &\triangleq p^{\leftarrow} \end{aligned}$$

The names for role pairs lead to the following information descriptors. Let $p$ and $q$ be role types such that $\mathsf{CNm}(p, q) = n$, then:

$$\mathbb{D}[\![\, n \,]\!] \quad \triangleq \quad p \circ q^{\leftarrow}$$

If $v$ is a variable and $c$ a constant, we also have:

$$\begin{aligned} \mathbb{D}[\![\, v \,]\!] &\triangleq v \\ \mathbb{D}[\![\, c \,]\!] &\triangleq c \end{aligned}$$

### 4.6.3 Complex information descriptors

If $X$ and $Y$ are information descriptors, then so is the combination:

$$\mathbb{D}[\![\, X \;\; Y \,]\!] \quad \triangleq \quad \mathbb{D}[\![\, X \,]\!] \circ \mathbb{D}[\![\, Y \,]\!]$$

With this concatenation operator, we for example have:

$$\begin{aligned} &\mathbb{D}[\![\, \mathsf{Person\ working\_for\ Department} \,]\!] = \\ &\mathbb{D}[\![\, \mathsf{Person} \,]\!] \circ \mathbb{D}[\![\, \mathsf{working\ for} \,]\!] \circ \mathbb{D}[\![\, \mathsf{Department} \,]\!] = \\ &A \circ p \circ q^{\leftarrow} \circ B \end{aligned}$$

If $X_1, \ldots, X_n$ are information descriptors, then the confluence operator is introduced at the information descriptor level as follows:

$$\mathbb{D}[\![\, \mathsf{THE\ COMBINATION\ OF}(X_1, \ldots, X_n) \,]\!] \quad \triangleq \quad \langle \mathbb{D}[\![\, X_1 \,]\!], \ldots, \mathbb{D}[\![\, X_n \,]\!] \rangle$$

The head oriented logical connectives $\overset{\mathsf{h}}{\wedge}$, $\overset{\mathsf{h}}{\vee}$, $\overset{\mathsf{h}}{\Longrightarrow}$, $\overset{\mathsf{h}}{\Longleftrightarrow}$ and $\overset{\mathsf{h}}{\ominus}$ lead to the following information descriptors:

$$\begin{aligned} \mathbb{D}[\![\, X \,\mathsf{AND\ ALSO}\, Y \,]\!] &\triangleq \mathbb{D}[\![\, X \,]\!] \overset{\mathsf{h}}{\wedge} \mathbb{D}[\![\, Y \,]\!] \\ \mathbb{D}[\![\, X \,\mathsf{MUST\ ALSO\ BE}\, Y \,]\!] &\triangleq \mathbb{D}[\![\, X \,]\!] \overset{\mathsf{h}}{\Longrightarrow} \mathbb{D}[\![\, Y \,]\!] \\ \mathbb{D}[\![\, \mathsf{IF}\, X \,\mathsf{THEN\ ALSO}\, Y \,]\!] &\triangleq \mathbb{D}[\![\, X \,]\!] \overset{\mathsf{h}}{\Longrightarrow} \mathbb{D}[\![\, Y \,]\!] \\ \mathbb{D}[\![\, X \,\mathsf{IF\ AND\ ONLYIF}\, Y \,]\!] &\triangleq \mathbb{D}[\![\, X \,]\!] \overset{\mathsf{h}}{\Longleftrightarrow} \mathbb{D}[\![\, Y \,]\!] \\ \mathbb{D}[\![\, X \,\mathsf{OR\ IS}\, Y \,]\!] &\triangleq \mathbb{D}[\![\, X \,]\!] \overset{\mathsf{h}}{\vee} \mathbb{D}[\![\, Y \,]\!] \\ \mathbb{D}[\![\, X \,\mathsf{COMBINED\ WITH}\, Y \,]\!] &\triangleq \mathbb{D}[\![\, X \,]\!] \overset{\mathsf{h}}{\oplus} \mathbb{D}[\![\, Y \,]\!] \\ \mathbb{D}[\![\, X \,\mathsf{BUT\ NOT}\, Y \,]\!] &\triangleq \mathbb{D}[\![\, X \,]\!] \overset{\mathsf{h}}{\ominus} \mathbb{D}[\![\, Y \,]\!] \end{aligned}$$

For the temporal operators we have:

$$\begin{aligned}
\mathbb{D}[\![\ \mathsf{ALWAYS}\ X\ ]\!] &\triangleq \mathsf{always}\ \mathbb{D}[\![\ X\ ]\!] \\
\mathbb{D}[\![\ \mathsf{SOMETIME}\ X\ ]\!] &\triangleq \mathsf{sometime}\ \mathbb{D}[\![\ X\ ]\!] \\
\mathbb{D}[\![\ X\ \mathsf{PRECEDES}\ Y\ ]\!] &\triangleq \mathbb{D}[\![\ X\ ]\!]\ \mathsf{precedes}\ \mathbb{D}[\![\ Y\ ]\!]
\end{aligned}$$

### 4.6.4 Domain rules

Using information descriptors we can also define rules/predicates pertaining to the domain modeled by an ORM model. If $X$ is an information descriptor, then we have the following atomic rules:

$$\begin{aligned}
\mathbb{R}[\![\ \mathsf{ANY}\ X\ ]\!] &\triangleq \exists[\![\mathbb{D}[\![\ X\ ]\!]] \\
\mathbb{R}[\![\ \mathsf{SOME}\ X\ ]\!] &\triangleq \exists[\![\mathbb{D}[\![\ X\ ]\!]] \\
\mathbb{R}[\![\ \mathsf{ALL}\ X\ ]\!] &\triangleq \forall[\![\mathbb{D}[\![\ X\ ]\!]]
\end{aligned}$$

Logical connectives are introduced as:

$$\begin{aligned}
\mathbb{R}[\![\ R\ \mathsf{AND}\ S\ ]\!] &\triangleq \mathbb{R}[\![\ R\ ]\!] \wedge \mathbb{R}[\![\ S\ ]\!] \\
\mathbb{R}[\![\ R\ \mathsf{OR}\ S\ ]\!] &\triangleq \mathbb{R}[\![\ R\ ]\!] \vee \mathbb{R}[\![\ S\ ]\!] \\
\mathbb{R}[\![\ R\ \mathsf{IMPLIES}\ S\ ]\!] &\triangleq \mathbb{R}[\![\ R\ ]\!] \Rightarrow \mathbb{R}[\![\ S\ ]\!] \\
\mathbb{R}[\![\ R\ \mathsf{IFF}\ S\ ]\!] &\triangleq \mathbb{R}[\![\ R\ ]\!] \Leftrightarrow \mathbb{R}[\![\ S\ ]\!] \\
\mathbb{R}[\![\ \mathsf{NOT}\ R\ ]\!] &\triangleq \neg\, \mathbb{R}[\![\ R\ ]\!]
\end{aligned}$$

As an abbreviation we also use:

$$\mathsf{NO}\ X \triangleq \mathsf{NOT\ SOME}\ X$$

For the temporal operators we have:

$$\begin{aligned}
\mathbb{R}[\![\ \mathsf{ALWAYS}\ R\ ]\!] &\triangleq \mathsf{always}\ \mathbb{R}[\![\ R\ ]\!] \\
\mathbb{R}[\![\ \mathsf{SOMETIME}\ R\ ]\!] &\triangleq \mathsf{sometime}\ \mathbb{R}[\![\ R\ ]\!] \\
\mathbb{R}[\![\ R\ \mathsf{PRECEDES}\ S\ ]\!] &\triangleq \mathbb{R}[\![\ R\ ]\!]\ \mathsf{precedes}\ \mathbb{R}[\![\ S\ ]\!]
\end{aligned}$$

## Bibliography

[BH96] A.C. Bloesch and T.A. Halpin. ConQuer: A Conceptual Query Language. In B. Thalheim, editor, *Proceedings of the 15th International Conference on Conceptual Modeling (ER'96), Cottbus, Germany, EU*, volume 1157 of *Lecture Notes in Computer Science*, pages 121–133, Berlin, Germany, EU, October 1996. Springer.

[HPW93] A.H.M. ter Hofstede, H.A. (Erik) Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[Mee82] R. Meersman. The RIDL Conceptual Language. Technical report, International Centre for Information Analysis Services, Control Data Belgium, Inc., Brussels, Belgium, EU, 1982.

[Pro94] H.A. (Erik) Proper. ConQuer–92 – The revised report on the conceptual query language LISA–D. Technical report, Asymetrix Research Laboratory, University of Queensland, Brisbane, Queensland, Australia, 1994.

[Spi88] J.M. Spivey. *Understanding Z: A Specification Language and its Formal Semantics*. Cambridge University Press, Cambridge, United Kingdom, EU, 1988.

[Sto77] J.E. Stoy. *Denotational Semantics: The Scott–Strachey Approach to Programming Language Semantics*. MIT Press, Cambridge, Massachusetts, USA, 1977.

[WHB92] Th.P. van der Weide, A.H.M. ter Hofstede, and P. van Bommel. Uniquest: Determining the Semantics of Complex Uniqueness Constraints. *The Computer Journal*, 35(2):148–156, April 1992.

[WK03] J. Warmer and A. Kleppe. *The Object Constraint Language: Getting Your Models Ready for MDA*. Addison Wesley, Reading, Massachusetts, USA, 2nd edition, 2003. ISBN 0321179366

# Chapter 5

# Advanced Object-Role Modeling

Version: 13-05-05

This chapter is mainly based on the work reported in [HW93, BBMP95, HP95, CP96].

## 5.1 Subtyping

Sub-typing is an important feature of fact-based modeling. Figure 3.6 (page 58) showed an example of subtyping in terms of a specialization hierarchy.

The population of a subtype can be restrained further. Rules can be specified that specify the 'maximum' and 'minimum' population of a sub-type. Normally, if $x$ Sub $y$ then the population of $x$ is bounded by

$$\emptyset \subseteq \mathsf{Pop}(x) \subseteq \mathsf{Pop}(y)$$

Graphically, this leads to the situation i) as depicted in Figure 5.1.

The population of a subtyping can be restricted further by requiring it to be a sub/superset of some set of instances specified by a (subtype constraining) rule. Situations ii), iii) and iv) of Section 3.6 depict this graphically. In situation ii), the population of X should at least consist of those match rule $R$. In general, this can be defined as follows:

$$\mathsf{SubSet}(x, R) \triangleq \mathsf{always} \forall_{\mathbb{D}[\![ R ]\!] \overset{\mathsf{h}}{\Longrightarrow} x}$$

Note: for $R$ to be a sensible rule, we should at least have for any $y$ such that $x$ Sub $y$:

$$\mathsf{Pop} \models_t \mathsf{always} \forall_{\mathbb{D}[\![ R ]\!] \overset{\mathsf{h}}{\Longrightarrow} y}$$

otherwise $\mathsf{SubSet}(x, R)$ would lead to an inconsistent set of constraints on the population.

While the situation from ii) provides a minimum population for X, the situation depicted in iii) does the reverse by demanding a maximimum population. The population of X is limited to those instances of X's *supertypes* that match rule $S$.

$$\mathsf{SuperSet}(x, S) \triangleq \mathsf{always} \forall_{x \overset{\mathsf{h}}{\Longrightarrow} \mathbb{D}[\![ S ]\!]}$$

The situation provided in iv) combines the maximum and minimum population. Situation v) represents a very special case. In this case, the rule $R$ actually fully determines the population of the subtype, since:

$$\mathbb{D}[\![ R ]\!] \overset{\mathsf{h}}{\Longrightarrow} x \overset{\mathsf{h}}{\Longrightarrow} \mathbb{D}[\![ R ]\!]$$

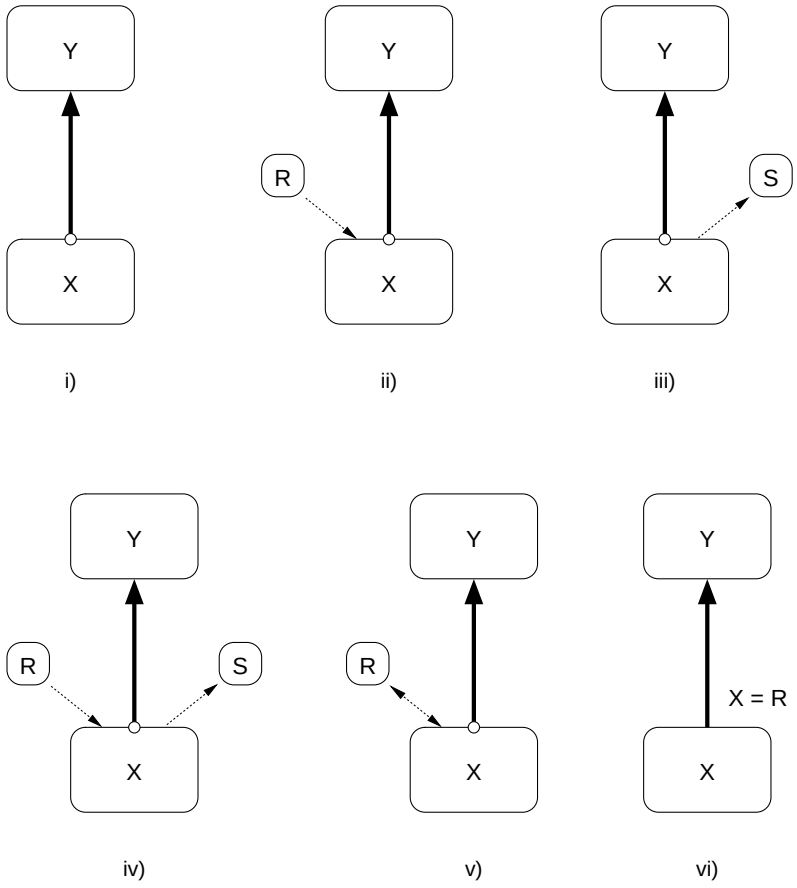



Figure 5.1: Restricting subtyping

in other words:

$$\mathbb{D}[\![\ R\ ]\!] \overset{\mathsf{h}}{\Longleftrightarrow} x$$

As a graphical abbreviation we will use the notation provided in situation vi), which corresponds to the traditional notion of *specialization* from ORM [Hal01].

A further interesting case of specialization is depicted in Figure 3.7 (page 59). It illustrates how specialization hierarchies can have multiple roots. This example also introduces two important classes of constraints: exclusiveness and totality of specializations. If $x_1, \ldots, x_n$ are types, then we can define:

$$\mathsf{Exclusive}(x_1, \ldots, x_n) \triangleq \mathsf{always}\, \forall_{(x_1 \wedge (x_2 \vee \ldots \vee x_n) \Rightarrow 0) \wedge \ldots \wedge (x_n \wedge (x_1 \vee \ldots \vee x_{n-1}) \Rightarrow 0)}$$

If $x_1, \ldots, x_n$ are types with a common supertype $s$ (such that $x_1\ \mathsf{Sub}\ s \wedge \ldots \wedge x_n\ \mathsf{Sub}\ s$) then we can define:

$$\mathsf{Total}(s : x_1, \ldots x_n) \triangleq s \overset{\mathsf{h}}{\Longleftrightarrow} x_1 \vee \ldots x_n$$

To express the semantics of totality in general we first need to identify all common supertypes of a set of types:

$$\mathsf{CommonSuper}(T) \triangleq \left\{ y \;\middle|\; \forall_{x \in T}\, [x\ \mathsf{Sub}\ y] \right\}$$

If $T$ is a set of types such that $\mathsf{CommonSuper}(T) \triangleq \{y_1, \ldots, y_m\}$, with $n > 0$, then we can define:

$$\mathsf{Total}(T) \triangleq \mathsf{Total}(y_1 : T) \wedge \ldots \wedge \mathsf{Total}(y_n : T)$$

Based on [HW93] a graphical abbreviation can be used for this kind of sub-typing. This is depicted in Figure 5.2. The right-hand side provides an abbreviation for the situation depicted in the left-hand side.

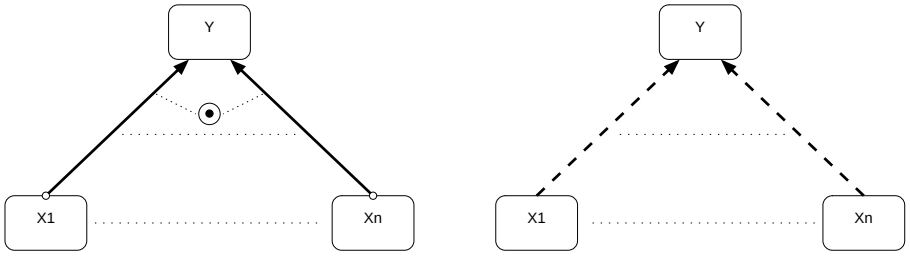



Figure 5.2: Generalization

## 5.2 Overlap of populations

When considering the ORM schema as depicted in Figure 3.7, one would expect the populations of Vehicle and Distance to be disjoint, while the populations of Vehicle and Product are expected to overlap. Thus far, we have not introduced any formal mechanism to enforce this type of behavior other that the inclusion of populations for sub-types. To properly formalize this, we first define the family (as it is 'alive' at some point in time) of an object type based on the sub-typing hierarchy as follows:

$$\mathsf{Family}_t(x) \triangleq \left\{ y \in \hat{\mathcal{OB}}_t \;\middle|\; y\ \mathsf{Sub}\ x \vee y = x \right\}$$

Two object types are deemed *type related* iff their families overlap:

$$x \sim_t y \triangleq \mathsf{Family}(x) \cap \mathsf{Family}(y) = \emptyset$$

If the population of object types overlaps, then they must be type related:

**[S46]** $\mathsf{Pop}_t(x) \cap \mathsf{Pop}_t(y) \neq \emptyset \Rightarrow x \sim_t y$

Note that the converse does not necessarily hold.

## 5.3 Abstraction

To introduce abstraction (*schema decomposition*), we start with an example domain taken from [CP96].

For our example domain, we consider a bank. Figure 5.3 shows the top level abstraction of the banking domain. This schema displays five types: Bank, Client, Service, enjoy, of. The Bank type is an abstracted type and forms the top abstraction of the entire banking application. This is also the reason why the enjoy and of relationship types, together with the remaining object types playing a role in these relationship types, are drawn inside the Bank type. Both Client and Service types are abstractions themselves, although their underlying structure is not shown at the moment. When stepping down to a lower level of abstraction, the *void* in these types will be filled with more detail.

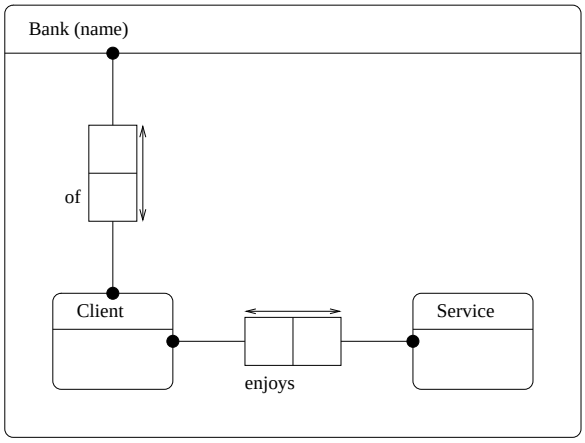



Figure 5.3: The top diagram of the Bank domain

The Client and Service type are involved in a relationship type called enjoys. This is a many to many relationship where each client must at least enjoy one service and each service offering must be enjoyed by some person. The two black dots indicate that a client of the bank must indeed enjoy some service, and conversely each service must be used by some client. The arrow tipped bar spanning the two roles of the enjoys relationship type indicates that it is a many to many relationship. Similarly, the of relationship type models the fact that a bank has many clients, and clients can be client of many banks. The (name) suffix to Bank indicates that a bank is identified by a name. Basically, the use of the (name) suffix is a graphical abbreviation of the schema fragment depicted in Figure 5.4. The broken ellipse of BankName type indicate that it is a *value type*; i.e. its instances are directly denotable (strings, numbers, audio, video, html).

As a first refinement step we can now take a closer look at what a client is. The details of the Client type are shown in Figure 5.5. There we can see that each client is identified by a Client Nr, as indicated by the (nr) suffix to Client. Each client provides the bank with a unique address as indicated by the arrow tipped bar spanning the role of the lives at relationship type that is attached to Client. This address is mandatory for each client. This "mandatoryness" is indicated by the black dot. Address is a normal object type without any other types clustered to it. Therefore, it is drawn in the traditional ORM way using a solid ellipse. The (description) suffix to Address within the solid ellipse indicates that an address is identified by a description. This corresponds to the same underlying graphical abbreviation.

Clients must all provide at least one name, but they may have aliases. This leads to the arrow tipped bar spanning both roles of the has fact type, and the black dot on the client side. For authorization of transactions ordered by telephone or fax, the bank and the client agree upon a unique password. The combination of a password and address must uniquely identify a client (indi-

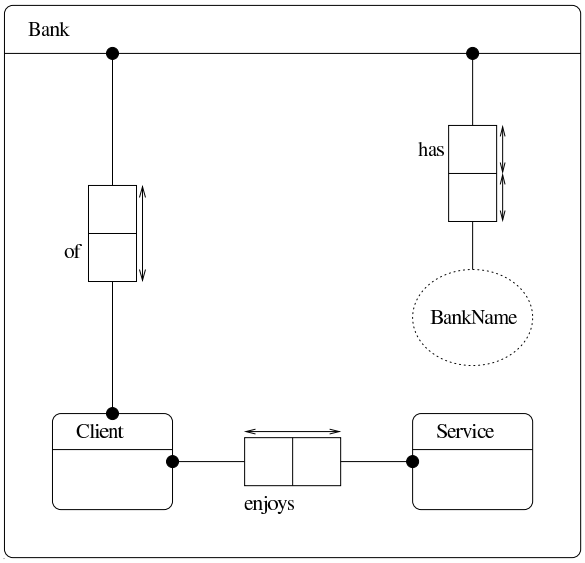



Figure 5.4: Fully detailed top diagram

cated in the diagram by the encircled U). Finally, clients may have a number of phone numbers at which they can be reached.

With respect to the abstractions, we can now say that the relationship types has identifying, lives at, reachable at, has (together with the types playing a role in these relationship types) are clustered to Client. For each abstracted type, like Client, such a clustering of types (from a lower level of abstraction) is provided. This could be an emptyset.

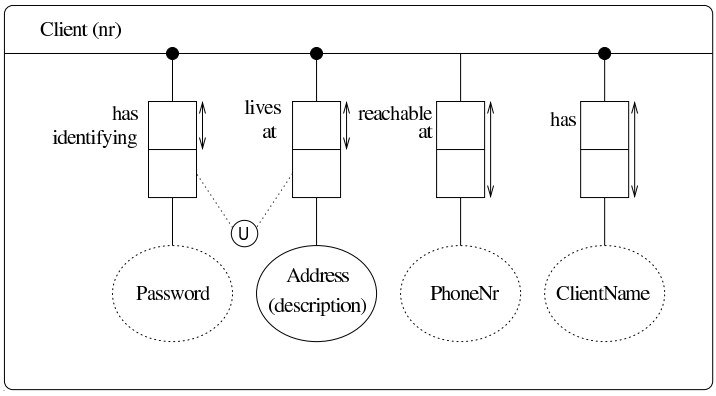



Figure 5.5: Refinement of the client type

In this example we refer to relationship types used in the bank example by means of the text associated with these relationship types, such as has identifying. This text is a so-called *mix fix predicate* verbalization. These mix fix predicate verbalizations do not have to be unique. The verbalization has typically occurs numerous times in an average conceptual schema. For example: Client has Client Name and Client has Password. To uniquely identify relationship types (and types in general), each type receives a unique name. For instance Client Naming and Issued Passwords for the two earlier given examples.

The next refinement of the bank domain provides us with more details about the service types available from the bank. This is depicted in Figure 5.6. The Service type is a generalization of three

basic types: Credit Card Account, Access Account, and Term Deposit Account. The Access Accounts and Credit Card Accounts are first combined into a so-called Statement Account. It should be noted that during a top-down modeling process, a type like Credit Card Account will start out as a 'normal' entity type like Address. However, as soon as other types are clustered to such an entity type, they become abstracted types.

The double lining around the Access Account type indicates that this type occurs in multiple clusterings. A CASE Tool supporting this kind of graphical representation, could have a feature in which clicking on such a double lining results in a list of (abstracted) types in whose clustering this type occurs.

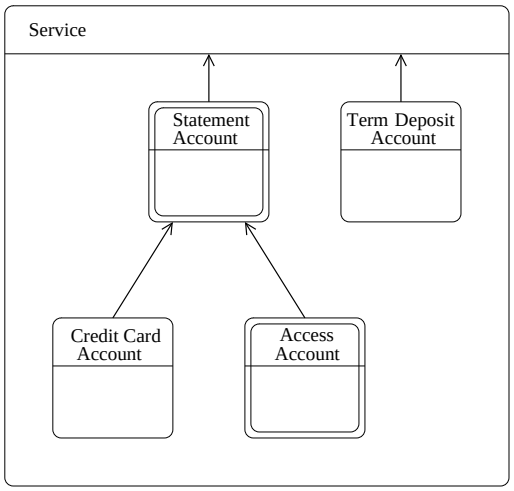



Figure 5.6: Refinement of the service type

As stated before, a statement account is a generalization of an access account and a credit card account. The intuition behind a statement account is that for such an account regular statements are sent to the clients and that a transaction record is kept. These details of the statement account are shown in Figure 5.7. For each statement account, a number of statements can be issued. A statement lists a number of transactions This is captured by the lists fact type. This fact type is, however, derivable from the (to be introduced) issue date of a statement and the dates at which the transactions took place. This derivability is indicated by the asterisk.

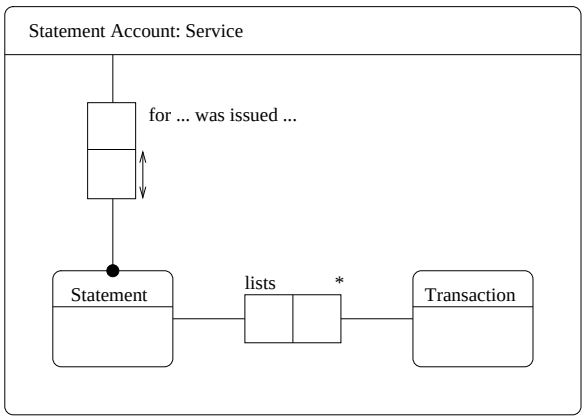



Figure 5.7: Refinement of statement account

One of the key features of the fact based modeling is inheritance of properties between types in

specializations. Instances (populations) are inherited in the direction of the arrows. For example, each credit card account is a statement account. Other properties, like clustered types, are inherited downwards. Typically, properties at the type level are inherited downward, while properties on the instance level are inherited upwards. The types clustered to Statement Account are therefore formally also part of the clusterings of Credit Card Account and Access Account. Nevertheless, to avoid cluttered diagrams, we have chosen not to show this inheritance explicitly in the diagrams. Therefore, the details of the Credit Card type do not show the details of Statement Account. The details of the Credit Card Type are provided in Figure 5.8. For each credit card the bank stores its kind, the spending limit, as well as the access account to which the credit card is linked. The suffix ": Statement Account" to "Credit Card Account (nr)" hints at the inheritance of the clustered types to Statement Account. In a CASE Tool supporting our technique, one could implement the facility that clicking on the Statement Account suffix leads to the inclusion of the clustered types introduced by Statement Account. Note that both Access Account and Money Amount have double lining, indicating that they occur in multiple clusters.

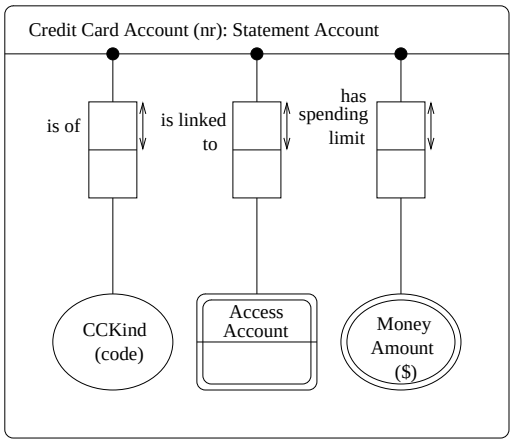



Figure 5.8: Refinement of the credit card type

For Access Account, the details are shown in Figure 5.9. All extra information actually shown there is the identification of an access account; an Access Account Nr as indicated by the (nr) suffix. Similar to the Credit Card Account, all types clustered to Statement Account are also clustered to Access Account, but we do not display this graphically.

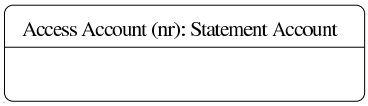



Figure 5.9: Refinement of an access account

Figure 5.10 shows the details of a statement. Each Statement is issued on a unique date. This date, together with the Statement Account for which the Statement was issued, identifies each Statement. Note that we decided to draw some contextual information of the Statement type to show how this type is identified. The for ... was issued ... and Statement Account types are not part of the clustering of Statement. The balance as listed on a Statement is, for obvious reasons, derivable from the Transactions that have taken place on this account.

The refined view on a transaction is shown in Figure 5.11. A Transaction is identified by the combination of the account it is for and a unique (for that account) transaction number. Note that contrary to a Statement, all components needed for the identification of Transactions are part of the clustering. Each Transaction involves a certain money amount, occurs on a date, and is either a

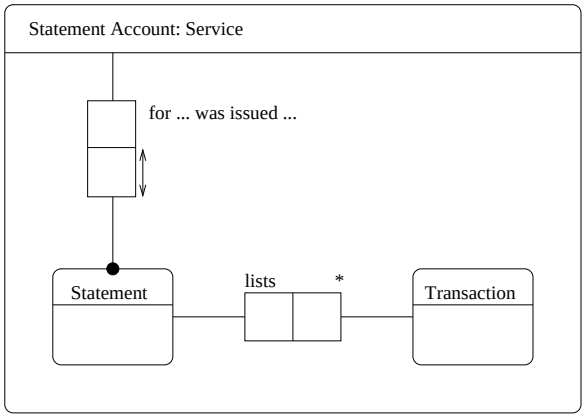



Figure 5.10: Refinement of a statement

debit or credit transaction (depicted by TR Kind). Furthermore, for each Transaction, some (unique) description may be provided. This example also shows that we must allow for *mutually recursive* abstractions, as the Transaction and Statement Account refinements refer to each other.

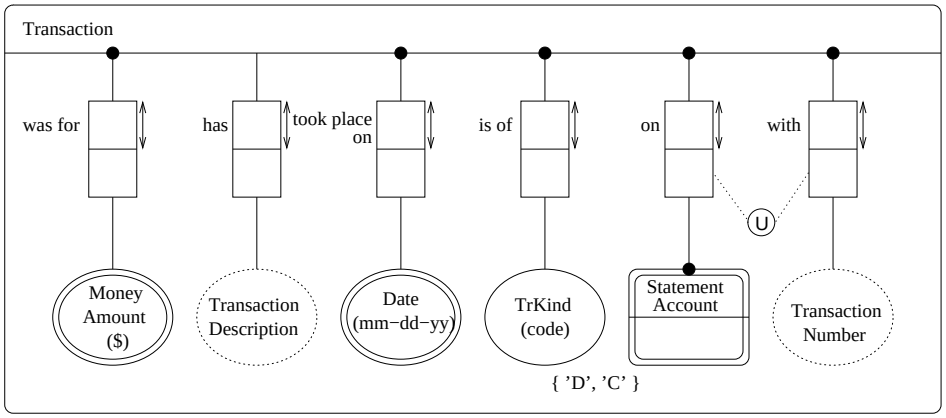



Figure 5.11: Refinement of a transaction

Term deposits form a world on their own. This is elaborated in Figure 5.12. On each Term Deposit Account, a client can have a series of term deposits. Each time a Term Deposit matures, this term deposit can be rolled-over leading to a new Term Deposit on the current Term Deposit Account. A special kind of Term Deposit is the Long Term Deposit, which is a subtype of Term Deposit. As each subtype inherits all properties from its supertype, the Long Term Deposit type is an abstracted type as well. For these Long Term Deposits we store whether the deposit is to be automatically rolled-over into a new deposit (the short Term Deposits are of this kind by default). In the refinement of a Long Term Deposit, we shall also see what the so-called subtype defining rule for these Long Term Deposits is. Upon maturation, the invested amount including the interest accrued is transferred to a pre-nominated Access Account. Finally, the interest rate given on the deposit is derived from a table listing the Periods for which amounts can be invested. The details of the Period type are given below.

A Term Deposit itself is a clustering of the start and ending dates of the deposits and the money amount invested. This is depicted in Figure 5.13. A Long Term Deposit is a term deposit with a duration of more than 60 days. In Figure 5.14 the details of a long term deposit are shown, including the subtype defining rule. The Long Term Deposit type inherits all clustered types from

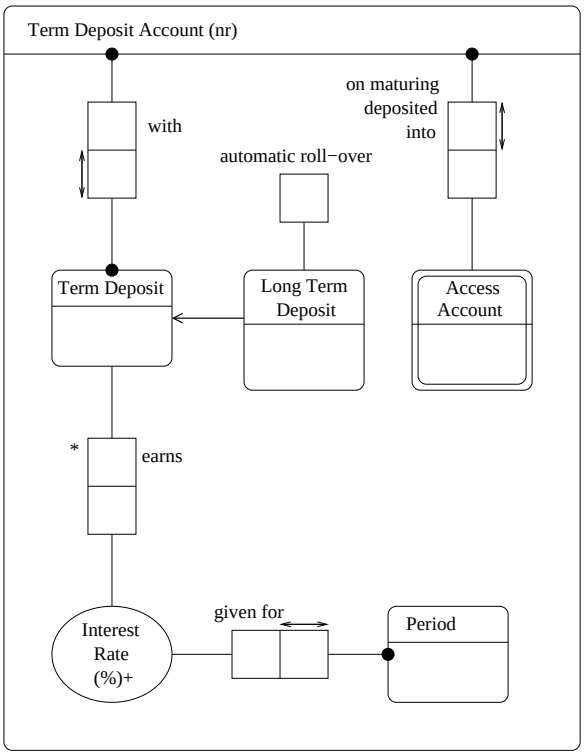



Figure 5.12: Refinement of a term deposit account

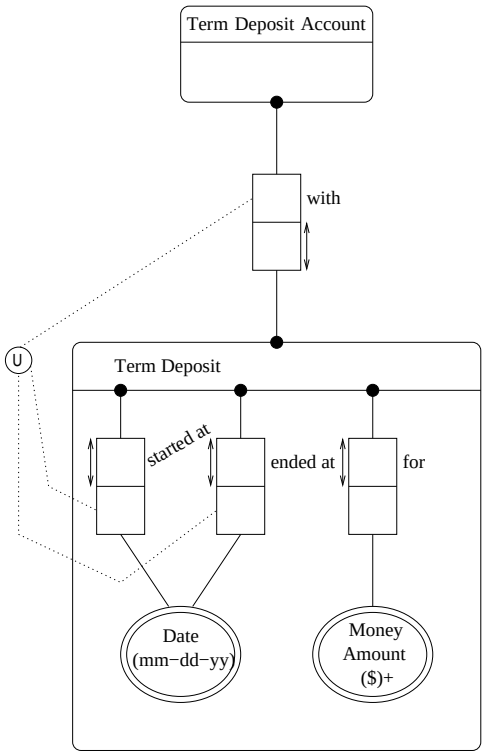
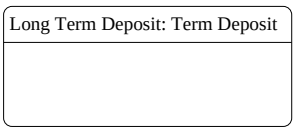



Figure 5.13: Refinement of a term deposit

Term Deposit, while not adding anything to this. Finally, the complete definition of the interest periods are given in Figure 5.15.




Long Term Deposit IS A
Term Deposit wich started at Date + 60 days > Date
at which ended THAT Long Term Deposit

Figure 5.14: Refinement of a long term deposit

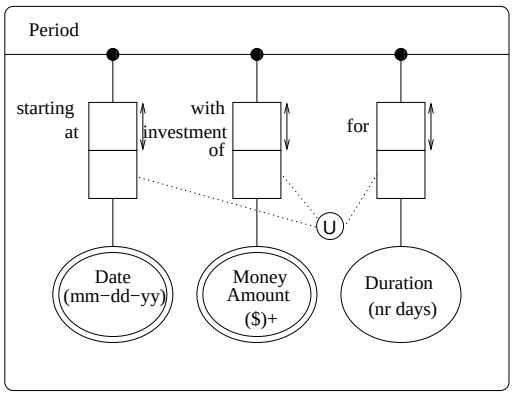



Figure 5.15: Refinement of periods

This completes the schema of the example domain. When modeling a domain like this, the modeler has the choice of using as many layers of abstraction as the modeler sees fit. We only provide a mechanism to introduce these abstractions and are (initially) not so much concerned with the 'sensibility' of abstraction steps. One may, for example, argue that the example given in this section has been split up into too many abstraction levels.

Sometimes, an analyst may want to see the entire schema. This is quite easy to do by uniting all clusterings into one large schema. From the above discussed schema fragments, one can derive the complete ORM schema as depicted in Figure 5.16 by uniting all clusters. This is, however, still not the 'lowest' level at which an ORM diagram can be displayed, since we have used the standard abbreviations for simple identifications and the short notation for objectifications. Objectification is a concept we have not yet discussed in this paper.

Also when looking at a design procedure for ORM schemas as presented in [Hal95] the decision to model a Transaction, say, as an objectification or a flat entity type is based on considerations of abstraction. When, for the modeling of the relationship types was for, has, took place on, and is of it is preferred to regard a transaction as an abstraction from its underlying relationships to a statement account and transaction number, then the objectified view is preferred to the flat entity view. This directly corresponds to the decision whether these underlying relationships should be clustered to the Transaction object type or not. Later we shall see that *set types*, *sequence types* and *schema typing* can be treated in a similar way. In [HW94, HW97] it is shown that *set types*, *sequence types* and some other composed types are not fundamental when introducing a special class of constraints which correspond to the set theoretic notion of *axiom of extensionality*. This then allows us to regard set typing, sequence typing and schema typing as forms of abstraction.

The schema depicted in Figure 5.16 has the same *formal semantics* as the combination of all previous schema fragments. However, the *conceptual semantics* is different as the abstraction levels

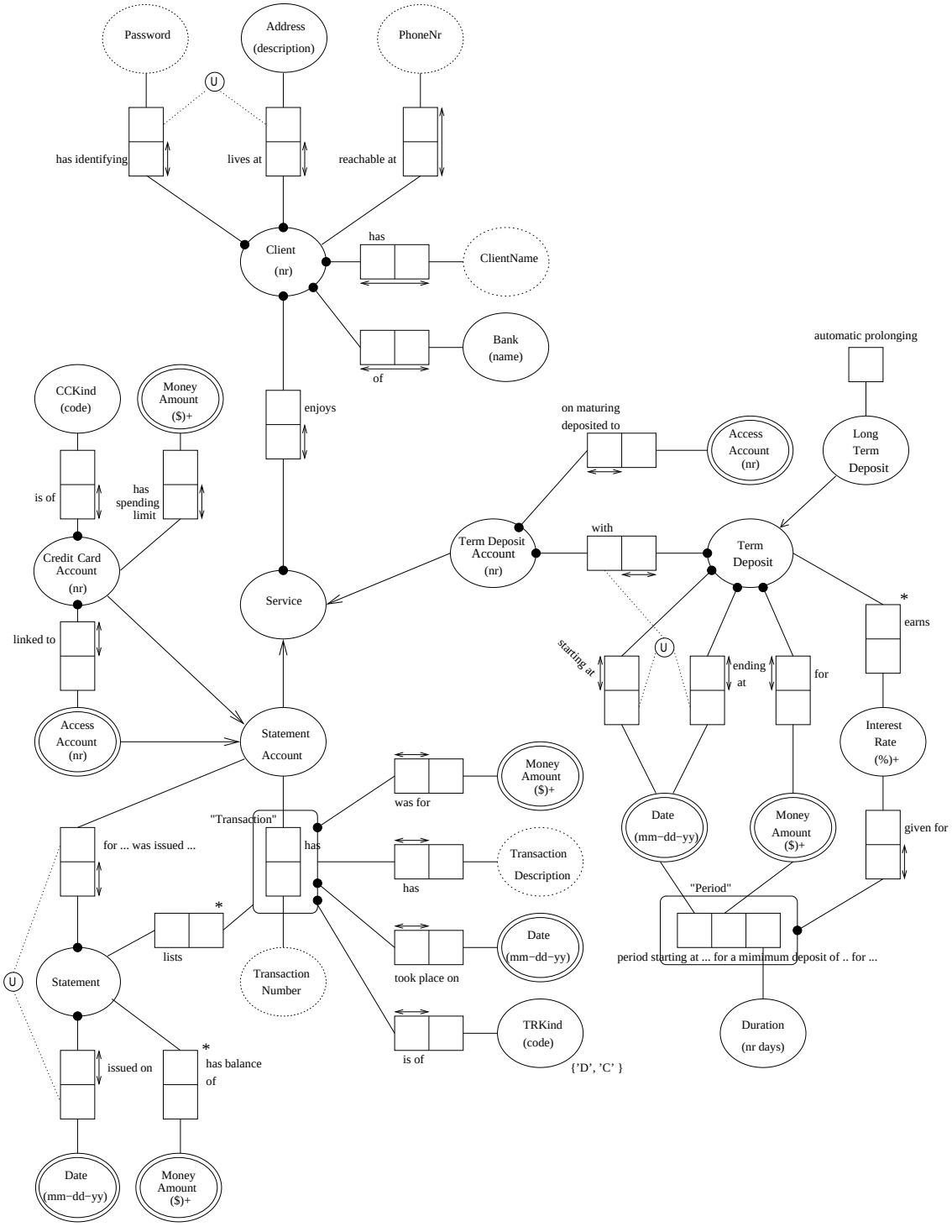



Figure 5.16: Complete diagram of the Bank domain

(the third dimension) are now missing. Schema abstraction is purely a syntactical issue, and thus carries no formal semantics. From the point of view of a modeler (and a participant of the universe of discourse), the abstractions do have a conceptual meaning. The abstractions represent certain choices of importance within the universe of discourse.

An (E)ER view can easily be derived as well by uniting all clusterings except for the lowest ones, but interpreting these as attributes. The (E)ER view on this domain is given in Figure 5.17. The version we used there is based on the one discussed in [BCN92]. Differing extended ER versions use different notations for this concept [EWH85, EN94, EGH+92]. The names for attributes in this diagram are simply based on the verbalizations given in the ORM schema. For most ER modelers, the concept of using elaborate verbalizations is new. One could allow for the specification of specific attribute names to, for example, abbreviate with minimum deposit of MoneyAmount ($)+ to MinDeposit. In this article we do not discuss naming conventions in detail but rather focus on the underlying conceptual issues. In [BBMP95] we have provided a more detailed study of the relationship between different ER versions and ORM. A detailed case study is also presented there, in which the different concepts underlying these modeling techniques are related, together with a mapping of the (graphical) concepts between the two classes of data modeling techniques.

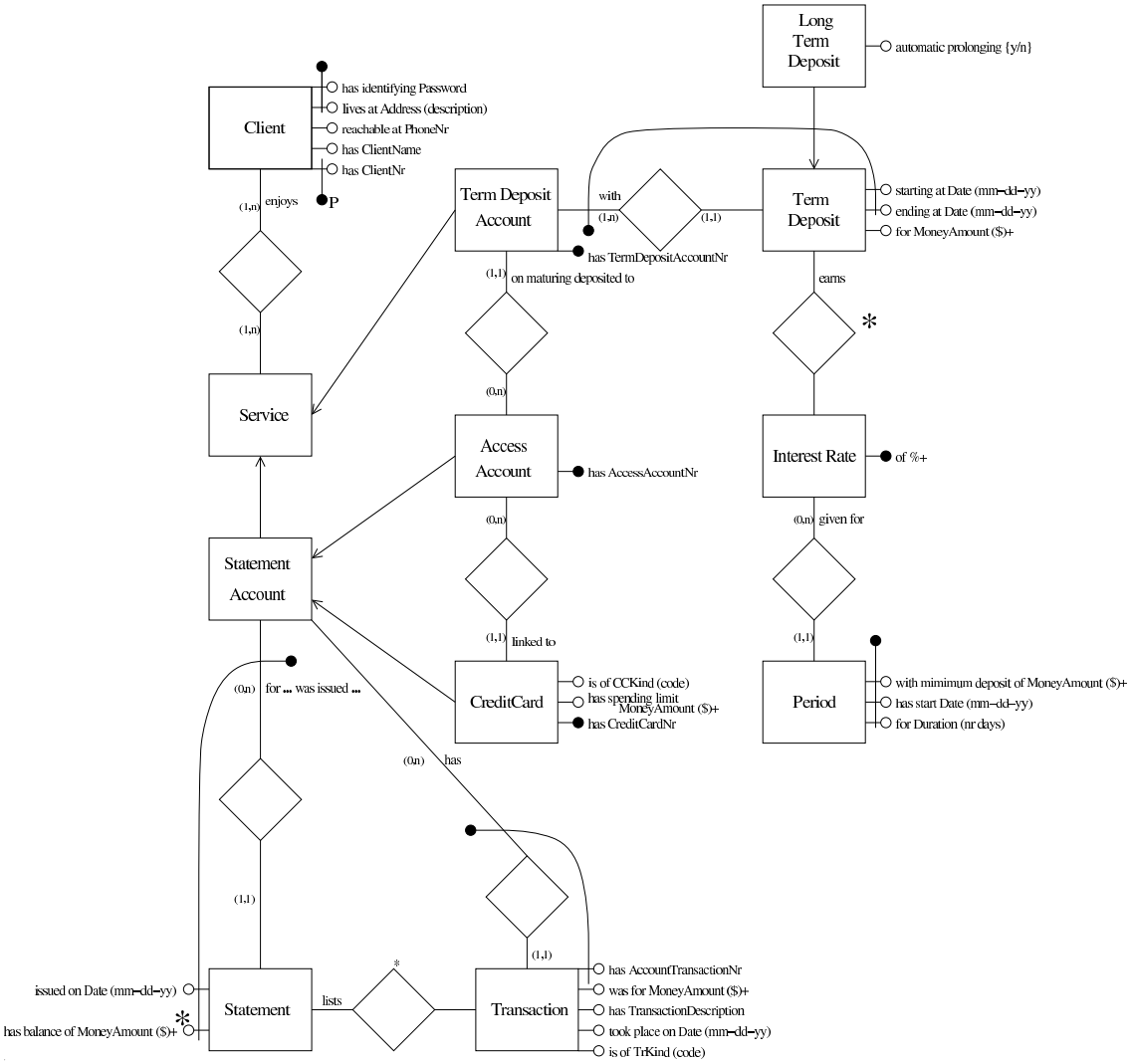



Figure 5.17: Complete ER diagram of the Bank domain

## 5.4 Set types

In set theory we use $\wp(X)$ to denote the set consisting of all subsets of $X$ (see for instance [Lev79]). When modeling complex domains, we sometimes have the need to model set types being types whose instances can be regarded as being sets of other instances. This notion is the same as the notion of grouping introduced in the IFO data model [AH87]). An illustrative example, taken from [HW94], is shown in Figure 5.18. A Convoy is taken to consist of a set of Ships, where this set of ships really *identifies* the convoy. In other words, if two convoys contain the same set of ships, they really are the same convoy. This is actually similar to the existentiality axiom from set theory:

$$\forall_i \, [i \in X \Leftrightarrow i \in Y] \Rightarrow X = Y$$

In Figure 5.18 the existential uniqueness is expressed by the circle with the two horizontal bars. If there would be only one bar, this would be normal uniqueness of the associated role. The extra (slightly shorter) bar signifies this to be an *existential* uniqueness constraint.

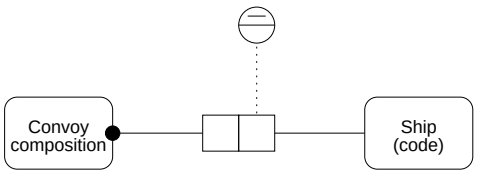



Figure 5.18: Convoy of ships

Formally, we introduce existential uniqueness by first identifying the variety an instance of a fact type may have with regards to a set of roles. Let $f \in \hat{\mathcal{FC}}$ be a fact type and $R \subseteq \mathsf{RolesOf}(f)$ be a set of roles involved in this fact type. The variety of instances $i \in \mathsf{Pop}(f)$ with regard to their roles $R$ is defined as:

$$\mathsf{Variety}(i, f, R) \triangleq \{ j[R] \mid j \in \mathsf{Pop}(f) \wedge j[\,\overline{R}\,] = i[\,\overline{R}\,] \}$$

For a set of roles $R$ of some fact type $f$, we can now express the existantial uniqueness constraint as:

$$\mathsf{ExtUnique}(f : R) \triangleq \forall_{i,j \in \mathsf{Pop}(f)} \left[ \forall_e \, [e \in \mathsf{Variety}(i, f, R) \Leftrightarrow e \in \mathsf{Variety}(j, f, R)] \Rightarrow j[\,\overline{R}\,] = i[\,\overline{R}\,] \right]$$

where $\overline{R} = \mathsf{RolesOf}(f) - R$. Note the correspondance to existantiality from set theory. A more compact form (which we are allowed to use due to the existantiality axiom from set theory) would be:

$$\mathsf{ExtUnique}(f : R) \triangleq \forall_{i,j \in \mathsf{Pop}(f)} \left[ \mathsf{Variety}(i, f, R) = \mathsf{Variety}(j, f, R) \Rightarrow j[\,\overline{R}\,] = i[\,\overline{R}\,] \right]$$

Using the abstraction mechanism from the previous section, we are able to more introduce a number of shorthand notations for set types. These are depicted in Figure 5.19.

## 5.5 Multi-set types

A variation of sets is a multi-set. In a multiset, elements can occur multiple times. Using the existantial uniqueness constraints, a multi-set can be modeled as depicted in Figure 5.20. In the depicted domain, a train composition class is defined as a multi-set of types of carriages.

Using the abstraction mechanism, we are again able to more introduce a number of shorthand notations for multi-set types. These are depicted in Figure 5.21.

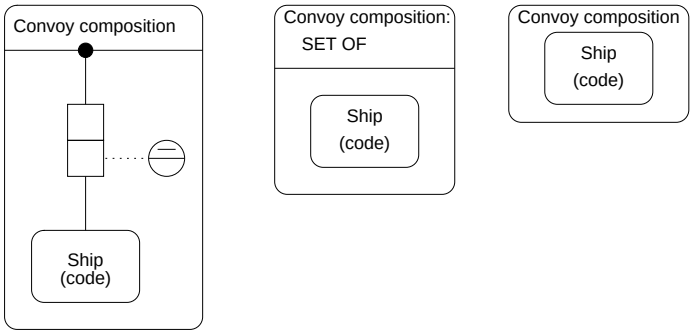



Figure 5.19: Shorthand notations for convoys of ships

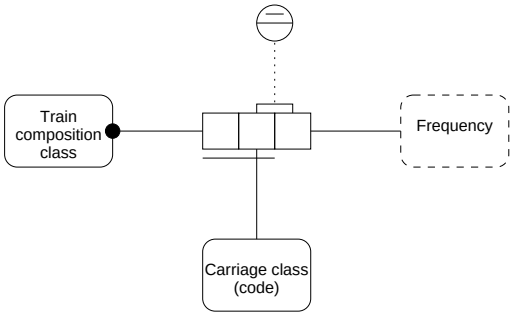



Figure 5.20: Train composition classes

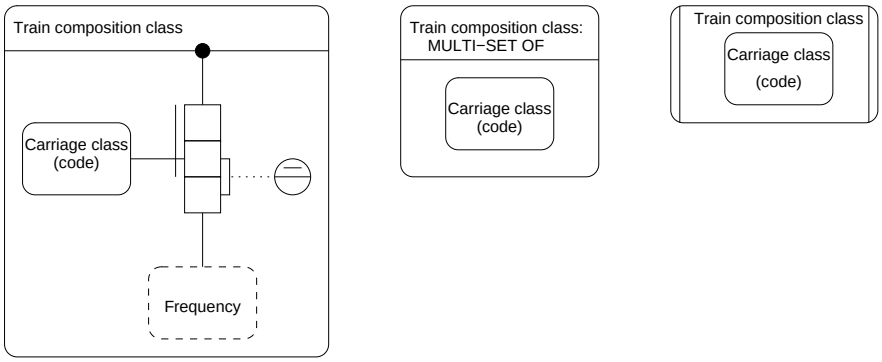



Figure 5.21: Shorthand notations for train composition classes

## 5.6 Sequence types

A specific train consists of a sequence of carriages. To model this compactly, we introduce the notion of a sequence type. This leads to the situation as depicted in Figure 5.22.

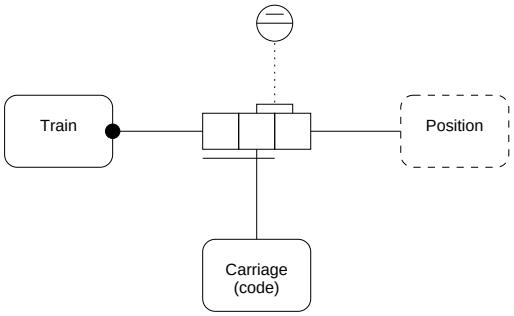



Figure 5.22: Train as a sequence of carriages

Using the abstraction mechanism, we are again able to more introduce a number of shorthand notations for sequence types. These are depicted in Figure 5.23.

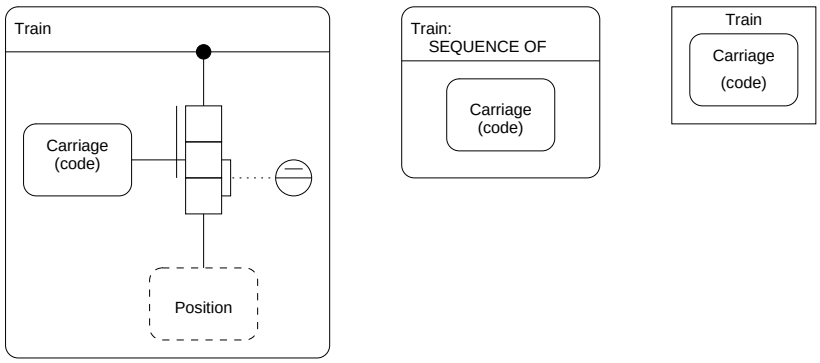



Figure 5.23: Shorthand notations for trains as sequences of carriages

## 5.7 Schema types

In some situations we need types whose instances are really entire populations of other (smaller) schemas. An example of such a situation is shown in Figure 5.24.

Using the abstraction mechanism, we are again able to more introduce a number of shorthand notations for schema types. These are depicted in Figure 5.25.

# Questions

1. Given the following populations: Pop(Carnivore) $= \{a, b, c\}$, Pop(Omnivore) $= \{d, e\}$ and Pop(Herbivore) $= \{f, g\}$. What are the populations of Animal, Flesh eater and Plant eater?
2. To have electrical power supplied to one's premises (i.e. building and grounds), an application must be lodged with the Electricity Board. The following tables are extracted from

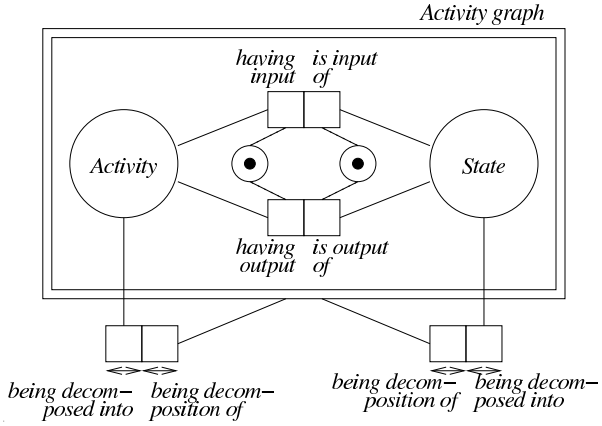



Figure 5.24: Activity graphs usign a schema type

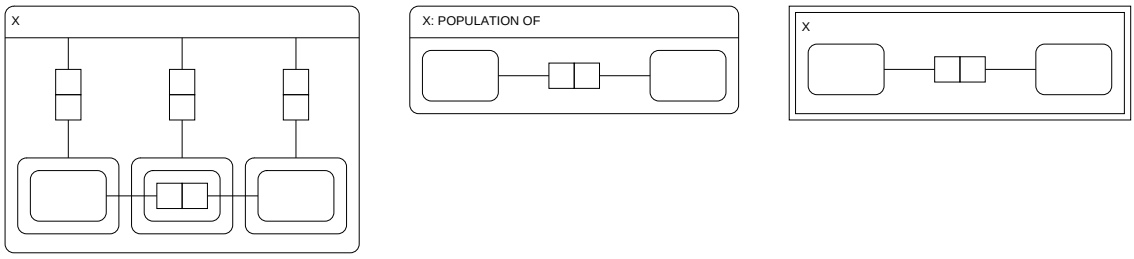



Figure 5.25: Shorthand notations for schema types

an information system used to record details about any premises for which power has been requested.

The following abbreviations are used: *premises# = premises number*, *qty = quantity*, *nr = number*, *commercl = commercial*. Each premises is identified by its premises#.

The electricity supply requested is exactly one of three kinds: "new" (new connection needed), "modify" (modifications needed to existing connection), or "old" (reinstall old connection). "Total amps" is the total electric current measured in Amp units. "Amps/phase" is obtained by dividing the current by the number of phases.

| premises# | city | kind of premises | kind of business | dog on premises | breed of dog | qty of breed | supply needed |
|---|---|---|---|---|---|---|---|
| 101 | Brisbane | domestic | . | yes | Terrier | 2 | new |
| 202 | Brisbane | commercl | car sales | no | . | . | modify |
| 303 | Ipswich | domestic | . | yes | Alsatian | 1 | old |
| | | | | | Poodle | 1 | |
| 404 | Redcliffe | commercl | security | yes | Alsatian | 3 | new |
| | | | | | Bulldog | 2 | |
| 505 | Brisbane | domestic | . | no | . | . | modify |
| 606 | Redcliffe | commercl | bakery | no | . | . | old |
| ... | ... | ... | ... | ... | ... | ... | ... |

Further details about new connections or modifications:

| premises# | load applied for (if known) | | | wiring completed? | expected date for wiring completion |
|---|---|---|---|---|---|
| | total amps | nr phases | amps/phase | | |
| 101 | 200 | 2 | 100 | no | 30-06-03 |
| 202 | 600 | 3 | 200 | yes | . |
| 404 | . | . | . | no | 01-08-03 |
| 505 | 160 | 2 | 80 | no | 30-06-03 |
| ... | ... | ... | ... | ... | |

The population is significant with respect to mandatory roles. Each premises has at most two breeds of dog.

Produce a fact-based model for this domain. Use specialization when needed. Include *uniqueness*, *mandatory role*, *subset*, *occurrence frequency* and *equality constraints*, as well as *value type constraints* that are relevant. Provide meaningful names.

If a fact type is derived it should be asterisked on the diagram and a derivation rule should be supplied.

Produce both a flat fact-based model, as well as a version that uses abstraction/decomposition to split this domain into more comprehensible chunks.

# Bibliography


[AH87] S. Abiteboul and R. Hull. IFO: A Formal Semantic Database Model. *ACM Transactions on Database Systems*, 12(4):525–565, December 1987.

[BBMP95] G.H.W.M. Bronts, S.J. Brouwer, C.L.J. Martens, and H.A. (Erik) Proper. A Unifying Object Role Modelling Approach. *Information Systems*, 20(3):213–235, 1995.

[BCN92] C. Batini, S. Ceri, and S.B. Navathe. *Conceptual Database Design – An Entity–Relationship Approach*. Benjamin Cummings, Redwood City, California, USA, 1992.

[CP96] P.N. Creasy and H.A. (Erik) Proper. A Generic Model for 3–Dimensional Conceptual Modelling. *Data & Knowledge Engineering*, 20(2):119–162, 1996.

[EGH+92] G. Engels, M. Gogolla, U. Hohenstein, K. Hülsmann, P. Löhr-Richter, G. Saake, and H.-D. Ehrich. Conceptual modelling of database applications using an extended ER model. *Data & Knowledge Engineering*, 9(4):157–204, 1992.

[EN94] R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems*. Benjamin Cummings, Redwood City, California, USA, 1994. Second Edition.

[EWH85] R. Elmasri, J. Weeldreyer, and A. Hevner. The category concept: An extension to the entity–relationship model. *Data & Knowledge Engineering*, 1:75–116, 1985.

[Hal95] T.A. Halpin. *Conceptual Schema and Relational Database Design*. Prentice–Hall, Englewood Cliffs, New Jersey, USA, 2nd edition, 1995.

[Hal01] T.A. Halpin. *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Mateo, California, USA, 2001. ISBN 1558606726

[HP95] T.A. Halpin and H.A. (Erik) Proper. Subtyping and Polymorphism in Object–Role Modelling. *Data & Knowledge Engineering*, 15:251–281, 1995.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[HW94] A.H.M. ter Hofstede and Th.P. van der Weide. Fact Orientation in Complex Object Role Modelling Techniques. In T.A. Halpin and R. Meersman, editors, *Proceedings of the First International Conference on Object–Role Modelling (ORM–1)*, pages 45–59, July 1994.

[HW97] A.H.M. ter Hofstede and Th.P. van der Weide. Deriving Identity from Extensionality. *International Journal of Software Engineering and Knowledge Engineering*, 8(2):189–221, June 1997.

[Lev79] A.Y. Levy. *Basic Set Theory*. Springer, Berlin, Germany, EU, 1979.

# Chapter 6

# The Act of Modelling

Version: 13-04-05

In this chapter, which is based on the work reported in [PBH04, HBP05], we turn to the question *how to model a domain*; a question to which there is no simple, one-size-fits-all answer.

## 6.1 What to model?

- Modeling goal
- Intended audience
- Viewpoint

## 6.2 The modeling challenge

### 6.2.1 Goal-bounded and communication-driven

Some modeling approaches, such as NIAM [NH89] and ORM [Hal01], suggest or prescribe a detailed procedure. Practice shows, however, that experienced modelers frequently deviate from such procedures [Ver93]:

> In most cases, [the information engineers] stated that they preferred to pay attention to a specific part of the problem domain, usually to fill clear lacunae in their insights in the problem domain. Their momentary needs strongly influenced the order in which the several modelling techniques were used. Modelling techniques were used as a means to increase insights or to communicate insights, be it in the problem domain itself or in a specific solution domain.

Yet deviating from a modeling procedure should be done with some caution. While a predefined modeling procedure should never become "an excuse to stop thinking", situational specificity should not become an excuse for taking an ad-hoc approach to the modeling effort. A more stable anchor is needed upon which modelers can base themselves when making decisions during the modeling process. We believe that domain modeling requires a goal-bounded and communication-driven approach. With goal-bounded we hint at the fact that when modeling a domain, a modeler is confronted with a plethora of modeling decisions. These decisions range from the modeling approach used, the intended use of the results, to decisions pertaining to the model itself. For example:

- What parts of the domain should be considered relevant?
- What is the desired level of detail and formality?
- To what level should all stakeholders agree upon the model?
- Should the model be a representation of an actual situation (*system analysis*) or of a desired situation (*design*)?
- Should the model be a representation of *what* a system should do, or should it be a representation of *how* a system should do this?
- Should a certain phenomenon in the domain be modeled as a relationship, or is it an object on its own?

Having an explicit, and articulated, understanding of the modeling goals provides modelers with guidance in making the right decisions with regards to the above mentioned issues. Modeling goals, therefore, essentially provide the *means* to *bound* modeling space.

In most situations where a domain needs to be modeled, the modeler cannot merely passively observe the domain. Modelers will need to interact with representatives from the domain. These representatives then become *informers* (who are likely to also have a stake with regards to the system being developed). Therefore, modelers will need to communicate intensively with the informers in order to refine the model. What is more, numerous domain models that are produced during system development will need to be accepted and agreed upon –validated– by the informers (being stakeholders of the future system). The claim has often been voiced that in modeling practice, 'the process is just as important as the end result', suggesting that a correct end-result is not always a guarantee for success. A domain model should ideally be a product of a *shared understanding of a domain's stakeholders*. It requires a 'buy-in' by all stakeholders involved. A domain model that is correct from a theoretical or technical point of view but does not have the required support from the key stakeholders is arguably worse than a domain model with some flaws that does have such support.

A modeling process can thus be seen as a communication-driven process [FW04b, VHP03]. The principles of natural language driven modeling approaches [NH89, EKW92, Kri94, Hal01] can be used as a basis for shaping the communication process between informer and modeler.

### 6.2.2 Aspects of a method

When considering a modeling approach or method, several aspects thereof can be discerned [SWS89, WH90]. An important distinction to be made is that between a product oriented perspective and a process oriented perspective. In terms of the framework presented in [SWS89, WH90] these are referred to as the *way of modeling* and *way of working*, respectively:

**Way of modeling –** The way of modelling provides an abstract description of the underlying modelling concepts together with their interrelationships and properties. It structures the models which can be used in the information system development, i.e. it provides an abstract language in which to express the models.

**Way of working –** The way of working structures the way in which an information system is developed. It defines the possible tasks, including sub-tasks and ordering of tasks, to be performed as part of the development process. It furthermore provides guidelines and suggestions (heuristics) on how these tasks should be performed.

In the case of domain modeling, the *way of working* represents the process followed when modeling a domain. In the following sections, we will mainly elaborate on this aspect. The *way of modeling* used for domain modeling is likely to be prescribed by a diagramming technique such as ORM diagrams [Hal01], ER diagrams [Che76] or UML class diagrams [BRJ99].

### 6.2.3 The process of modeling

In general, the goals underlying (business) domain modeling are [BPH04]-

1. articulate clear and concise meanings of business domain concepts and
2. achieve a shared understanding of the concepts among relevant stakeholders.

Based on the results reported in [Hop03], we consider domain modeling in the context of system development to chiefly concern three streams of (mutually influencing) activities-

**Scoping environments of discourse –** The aim of this stream of activities is to scope the environments of discourse that are relevant to the system being developed, and determine the set of actors associated to each of these environments.

**Concept specification –** For each of the identified environments of discourse, the relevant business domain concepts should be specified in terms of their:

- meaning
- relationships to other concepts (and the constraints governing these relationships)
- possible names used to refer to them

**Concept integration –** The concepts as identified and defined in the different environments of discourse may well clash. As a part of this, homonyms and synonyms are likely to hold between different terminologies. The aim of this stream of activities is to determine how to deal with this, and act upon it.

Since these streams of activities can be expected to influence each other, it is not likely that they can be executed in a strict linear order.

In general, the processes that aim to arrive at a set of concepts together with their meaning and names, are referred to as *conceptualization processes* [Hop03]. When, as in the context of software development, conceptualization is performed deliberately, as a specific task and with a specific goal in mind, it is referred to as an *explicit conceptualization process*. The above mentioned stream of activities called concept specification is such an explicit conceptualization process. In [Hop03, BPH04] a reference model for conceptualization processes is provided. This reference model distinguishes five streams of activities or *phases*:

**Assess domain and acquire raw material –** Domain modeling always begins with a brief scan or assessment of the domain to get a feeling for scope, diversity and complexity of the domain, as well as to identify the relevant stakeholders for the domain (usually but not necessarily a subset of the project stakeholders). In addition, the activity aims to bring together input documents of all sorts that provide a basic understanding of the environment of discourse that is relevant to the environment of discourse under consideration.

**Scope the concept set –** In this phase, formal decisions are to be made regarding the concepts that somehow play a role in the environment of discourse and how these concepts interrelate.[1]

**Select relevant concepts –** The goal of this phase is to focus on those concepts in the environment of discourse that bear some relevance to the system to be developed. These are the concepts that should be defined and named formally in the next step.

**Name and define concepts –** All of the concepts selected in the previous phase should be named and defined. Defining the concepts may also include the identification of rules/laws/constraints governing instances of the defined concepts.

**Quality checks –** Final quality checks on the validity, consistency and completeness of the set of defined concepts.

These streams should essentially be regarded as sub-streams of the concept specification stream.

[1] In an earlier version of this framework, this was referred to as *scoping the universe of discourse*.

## 6.3 Ambition levels for modeling

We have made a distinction between four levels of ambition at which a modeler may approach the task of modeling a domain. These levels can also be regarded as the order in which a novice modeler may learn the art of domain modeling:

**Singular –** This level of ambition corresponds to the modeling approaches as described in e.g. NIAM [NH89] and ORM [Hal01]. It involves the modeling of a *single* environment of discourse based on complete input; usually in terms of a *complete* verbalization of (*only*) the relevant parts of the domain.

**Elusive –** At this level of ambition, modelers need to cope with the unavoidable iterative nature of the modeling process. As a modeling and/or system development process proceeds, the insight into the domain may increase along the way. This replaces the idealized notion of completeness of input with one of incremental input. The increments in the model are not related to a changing domain, but rather to improved ways of conceptualizing it.

**Pluriform –** At this next level of ambition, we recognize the fact that when developing a realistic system, we do not simply deal with one single unified environment of discourse (and related terminologies and concepts), but rather with a number of interrelated environments of discourse [PH04].

**Evolving –** The final ambition level recognizes the fact that domains themselves are not stable; they evolve over time [PH04]. As a result, what may have started out as a correct model of a domain, may become obsolete due to changes in the domain. New concepts may be introduced, or existing ones may cease to be used. However, subtle changes may occur as well, such as minor changes in the meaning of concepts, or the forms used to represent them.

In the next section, we will discuss domain modeling at the *singular*, *elusive* and *pluriform* levels of ambition. The *evolving* level is omitted for now.

## 6.4 Meeting the challenge

This section aims to discuss the domain modeling process with respect to three of the identified levels of ambition- *singular*, *elusive* and *pluriform*. We will structure our discussion by using the framework of activity streams for domain modeling as introduced in the previous section.

### 6.4.1 Modeling a singular domain

At this level of ambition we are only interested in the modeling of a single environment of discourse based on complete input. In terms of the above framework for domain modeling, this ambition level assumes that-

- No (further) scoping of the environment of discourse is needed
- The domain has been assessed and raw material is available
- Concept integration only needs to take place within the given environment of discourse

Natural language driven modeling approaches like NIAM [NH89] and ORM [Hal01] concern elaborately described ways of executing a domain modeling process at this ambition level. For example, the modeling procedure as described in ORM [Hal01] identifies the following steps:

**Step 1 – Transform familiar examples into elementary facts** This step involves the verbalization in natural language of samples take from the domain.

**Step 2 – Draw the fact types and apply a population check** In this step, a first version of the schema is drawn. The plausibility of the schema is validated by adding a sample population to the schema.

**Step 3 – Trim schema and note basic derivations** In this step, the schema is checked to see if any of the identified concepts are basically the same, and should essentially be combined. Furthermore, derivable concepts (e.g. sales-price = retail-price + mark-up) are identified.

**Step 4 – Add uniqueness constraints and check the arity of fact types** At this point, it is determined how many times an instance of an identified concept can play specific roles. For example, is a *person* allowed to *own* more than one *car*?

**Step 5 – Add mandatory role constraints and check for logical derivations** This step completes the basic set of arity constraints on the relationships in the schema, by stating wether or not instances of a concept *should* play a role. For example, for each *car*, the *year of construction* should be specified.

**Step 6 – Add value, set-comparison, and subtyping constraints** The ORM diagramming technique provides a rich set of graphical constraints. This step is aimed at specifying these constraints.

**Step 7 – Add other constraints, and perform final checks** Finally, there may be some constraints in the domain that cannot be expressed graphically. In this last step, these constraints can be specified and

In terms of our framework for domain modeling processes, this procedure constitutes a rather specific way of executing the *concept specification* stream of activities. It is really geared towards the (conceptual) analysis of a domain in order to design a database, rather than a general analysis of concepts playing a role in a domain. The procedure presented above is not applicable to all situations and all modelers.

Even though the above order is very explicit, and therefore well suited for educational purposes, a goal-bounded approach to domain modeling requires a more refined view. The key question concerns the *goal* for which a domain is modeled. During the *definition* phase of the software development life-cycle, when the main goal is to support requirements engineering activities, the seven steps as described above are likely to be overkill. In such a context, modelers are likely to skip steps 6 and 7. The modeling procedure as discussed in [Hal01], also requires modelers to identify how concepts (such as car, co-workers, patient, etc.) are identified in a domain (e.g. by means of a registration number, employee number, patient number, etc.). During the definition phase, these identification mechanisms are not likely to be relevant (*yet*).

During the *design* phase of a software system most of the seven identified steps are indeed needed. However, experienced modelers are also likely to merge steps 1-3, steps 4-5, as well as steps 6-7, into three big steps. The resulting three steps will generally be executed consecutively on a 'per fact' base. In other words:

1. For each fact type, execute 1-3
2. For each fact type, execute 4-5
3. For each fact type, execute 6-7

Some more empirical background to this, experience based observation, can be found in e.g. [Ver93, page 161].

> The order in which the various modeling tasks are performed differs to a large extent. A clear distinction exists between prescribed modeling knowledge and applied modeling knowledge, in this respect. Whereas an almost strictly *linear* order of performing modeling tasks is prescribed, a very *opportunistic* order is actually used. This order seems to be determined by at least two essentially different factors- the problem domain and the information engineer.

Note that when an initial domain model already exists, e.g. as produced in the definition phase in support of requirements engineering, this will have to be used as a starting point for completion. In other words, in practice, a domain model is likely to develop incrementally along with the software development life cycle.

ORM is not the only modeling approach that is based on analysis of natural language. However, providing a full survey of such approaches is beyond the scope of this article. Nevertheless, two approaches are worth mentioning here. In [EKW92] the Object-Oriented Systems Analysis method is presented. It uses a natural-language based approach to produce an Object-Relationship Model (accidently also abbreviated as ORM) that serves as a basis for further analysis. The *way of working* used is not unlike that of ORM. Its *way of modeling*, however, has a more sketchy nature and has been worked out to a lesser degree. The KISS approach, as reported in [Kri94], also uses natural language analysis as its basis. It provides some support in terms of a *way of working*, but does this in a rather prescriptive fashion that presumes some very particular (and limited) intermediary goals. A wide spectrum of modeling concepts are introduced (*way of modeling*) covering a wide range of diagraming techniques (not unlike the UML [BRJ99]).

Independent of the approach used, a modeling process always needs to be flanked by a continuous communication process with the stakeholders [VHP03]. Communication brings along the aspect of documentation. Modeling itself can hardly do without face-to-face discussions; however, the (intermediate) results need to be recorded in such a way that they can be communicated effectively to the stakeholder community [FW02, Fre97]. In this respect we could argue that any modeling approach also needs a *way of communication/documenting*. Since documentation serves the purpose of communication, the documentation *language* should align with the accepted language concepts in the domain. In practice it turns out that graphical notations such as ORM or UML diagrams are not the most obvious way to communicate a model to stakeholders, since most domain stakeholders do not comprehend this kind of "IT language". Often, it is better to use more intuitively readable diagrams and natural language to communicate concepts and their relationships and constraints, while occasionally, a more mathematical or algorithmic style may be useful in certain expert domains.

# Chapter 7

# Natural-Language Foundations of Information-Systems Modeling

Version: 12-05-05

In this chapter, we will essentially use the ORM philosophy to model key aspects of information-systems. This requires the immediate introduction of some new concepts into our ontology, such as: agentive, experiencing , circumstantial and predicative role, action, predication, agent, subject and context elements. These concepts allow us to reason about such things as: *when* does something happens (triggering), *what* happens (action), *who/what* makes it happen (agentr), *who/what* does it happen *to* (subject) in *which/what* circumstances (context).

With these new concepts, we can typically take ORM domain models and "annotate" them in terms of the refined concepts. We will base this process of annotation on linguistic foundations, much in the same vein as the modeling approach from the *Domain Modeling* course. Based on these annotated ORM models, we will be able to mechanically derive process models in a modeling notation (ArchiMate [Lo05]) that is particularly suited for the modeling of business processes. This will be the focus of the *next* chapter.

## 7.1 Classes of roles

We consider organizational systems, i.e. open active systemor work-systems:

- So there is activity going on.
- In other words, there are active elements.
- A specific class of concepts should therefore be: actions.
- Even more, these actions are performed by *agents*, and are performed on *subjects*. We want to be able to 'talk' about these these agents and subjects.
- In other words, we actually need three additional classes of concepts: actions, agents and subjects.

Let us now analyze this closer from a natural language perspective. Consider, once again, the following domain:

> A person with name Erik is writing a letter to his loved one, at the desk in a romantically lit room, on a mid-summer's day, using a pencil, while the cat is watching.

with elementary facts:

A person is writing a letter
This person has the name Erik
This letter has a romantic nature
This letter has intended recipient Erik's loved one
The writing of this letter by Erik, occurs on a mid-summer's day
The writing of this letter by Erik, is done using a pencil
The writing of this letter by Erik, is done while the cat is watching
The writing of this letter by Erik, is taking place at a desk
This desk is located in a room
This room is romantically lit

As mentioned before, within these elementary facts, several *players* can be discerned. In the above example, we can isolate the players and facts as follows:

[A person] is writing [a letter]
[This person] has [the name Erik]
[This letter] has a [romantic nature]
[This letter] has intended recipient [Erik's loved one]
[The writing of this letter by Erik], occurs on a [mid-summer's day]
[The writing of this letter by Erik], is done using [a pencil]
[The writing of this letter by Erik], is done while [the cat] is watching
[The writing of this letter by Erik], is taking place at [a desk]
[This desk] is located in [a room]
[This room] is lit in [a romantic] way

The writing of the letter is the central fact in the above domain. All players in the facts describing the above domain are players in this domain. What are the actions, agents and subjects? Several degrees of activeness exist with regards to the role player plays in a fact/domain:

- Some roles will be more active than others.
- This is where we find inspiration in theories regarding verbs and the 'things' that may play a (functional) role in these verbs.
- We limit ourselves to those classes that are indeed relevant when considering activity in systems.

In decreasing scale of activity:

**Agentive role –** A role where the player is regarded as carrying out an activity.

In the example domain: The person.

Two sub-classes may be identified:

**Initiating role –** An agentive role, where the player is regarded as being the initiator of the activity.

**Reactive role –** A non-initiating agentive role.

**Experiencing role –** A role where the player is regarded as experiencing/undergoing an activity.

In the example domain: a letter, a loved one and the cat.

Three sub-classes may be identified:

**Patientive role –** An experiencing role, where the player is regarded as purposely undergoing changes (including its very creation)

**Receptive role –** An experiencing role, where the player is regarded as the beneficiary/recipient of the results of the activity

**Observative role –** An experiencing role, where the player is regarded as observing/witnessing the activity

**Contextual role –** A role where the player is regarded as being a part of the context in which the activity takes place.

Four sub-classes may be identified:

**Instrumental role –** A role where the player is regarded as being an instrument in an activity.

In the example domain: a desk and a pencil.

**Locative role –** A role, where the player is regarded as being the location of an activity, in terms of a spatial or temporal orientation.

In the example domain: the desk, the room and mid-summer's day

**Catalysing role –** A role, where the presence of the player is regarded as being beneficial (either in a positive or a negative way) to an activity.

In the example domain: the room lit in a romantic way.

**Predicative role –** A role where the player is regarded as being a predicate on some other player.

In the example domain: the name Erik.

The choice between these different levels of role is subjective. It depends on the viewer. The players of the four main classes of roles are regarded as *agents*, *subjects*, *context elements*, and *predicators* respectively.

In the example domain, the writing of the letter by the person can be regarded as a key activity in the domain. In other words, writing is an action, while the person is the agent and the letter is the subject. We may regard the cat and the loved one as a subject as well. What about the pen, the name Erik, the desk, etc? They are really players in *predications* over the other players. The fact that a pen is used by the person to write the letter is a *predication* of the writing *action*.

If we were to zoom in on a sub-domain of the above sketched domain, we could actually find that what was a subject in the super-domain is an agent in the sub-domain. Consider, for example, the sub-domain:

[The writing of this letter by Erik], is done while [the cat] is watching

When considered in isolation, one may quite easily argue that the primary action here is the watching, which is something that is being done by the cat. This really makes the cat into an agent rather than the subject, while the thing that is being watched (the writing) becomes the subject. This really means that our notions of agent, subject, action and predication are really to be taken *relative* to the domain under consideration, which is in line with the subjective approach to modeling as taken in this textbook.

Formally, we presume the existance of sets $\mathcal{AR}, \mathcal{ER}, \mathcal{CR}, \mathcal{PR} \subseteq \mathcal{RO}$ with agent and subject *roles* respectively. These classes of roles form a partition of the roles:

**[S47]** $\mathcal{AR}$, $\mathcal{ER}$, $\mathcal{CR}$ and $\mathcal{PR}$ are a partition of $\mathcal{RO}$.

With this we can also define the sets of actions and predications, respectively, as:

$$\begin{array}{lcl} \mathcal{AN} & \triangleq & \{\mathsf{Fact}(r) \mid r \in \mathcal{AR}\} \\ \mathcal{PN} & \triangleq & \mathcal{FC} - \mathcal{AN} \end{array}$$

In other words, all facts that have an agentive role are regarded as actions. The other facts are (pure) predications.

Typing should adhere our refined ontology. In other words:

**[S48]** For all $X \in \{\mathcal{FC}, \mathcal{RO}, \mathcal{PC}\}$ we have:

$$x \,\mathsf{HasType}\, y \Rightarrow (x \in X \Leftrightarrow y \in X)$$

All experiencing roles must be involved in some action:

**[S49]** $\forall_{d \in \mathcal{ER}} [\mathsf{Fact}(d) \in \mathcal{AN}]$

For the example given above, we would have:

Action: [Agent: A person] is writing [Subject: a letter]
Predication: [This person] has [the name Erik]
Predication: [This letter] has a [romantic nature]
Predication: [This letter] has intended recipient [Erik's loved one]
Predication: [The writing of this letter by Erik], occurs on a [mid-summer's day]
Predication: [The writing of this letter by Erik], is done using [a pencil]
Predication: [The writing of this letter by Erik], is done while [the cat] is watching
Predication: [The writing of this letter by Erik], is taking place at [a desk]
Predication: [This desk] is located in [a room]
Predication: [This room] is lit in [a romantic] way

Finally, a word of warning:

- Natural language also harbors 'conceptual prejudice'
- Most, if not all, Indo-European languages are rather state oriented: 'Charles the bold' stands on 'the floor'
- Some North-American native languages are activity oriented: 'Dances with wolves' stands on 'what supports us when walking'
- Imagine the impact on the way we view & design systems/organizations: Is nature state oriented or a continuous flow of activities?

## 7.2 Activity types

Based on the above discussions regarding actions and predications, we can identify which roles are of which class, and mark this in the ORM model. For the example from Figure 4.8, we have the situation as depictedin Figure 7.1.

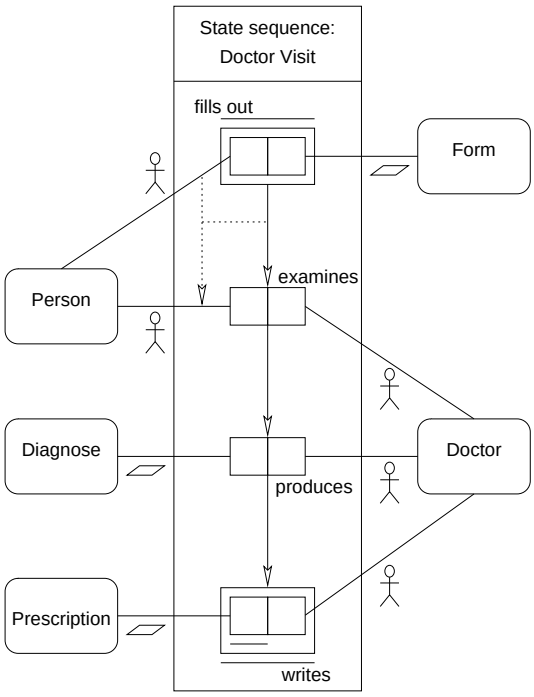



Figure 7.1: Compact model of a visit to a Doctor with alternative semantics

Those state-sequence types which comprise a number of action types (or other activity types) are considered to be (complex) *activity* types. Action types on their own are (atomic) *activity* types.

## Questions

1. Given the situation:

   > A person with name Erik is writing a letter to his loved one, at the desk in a romantically lit room, on a mid-summer's day, using a pencil, while the cat is watching.

   Produce a graph consisting of entities and relationships depicting this domain.

2. Consider the following domain:

   > Docent Proper voert de vakgegevens van Architectuur en Alignment in in het management informatiesysteem.

   Wat zijn hier de entiteiten en de relaties? Wat zijn de acties, actoren, actanden en predications.?

3. Stel je maakt een ontwerp voor een geldautomaat. Wat zijn voor dat domein de belangrijkste systeem entiteiten en hun onderlinge relaties? Hoe werken ze samen? Wat zijn hier de entiteiten en de relaties? Wat zijn de acties, actoren, actanden en predications.?

4. Proof Corollary 3.4.1 (page 53).

5. Proof Corollary 3.5.1 (page 55).

6. Consider the case from Question 3.9.

   Answer the following questions:

   (a) (Re)produce elementary facts for this domain.
   (b) What are the actions, actors and actands?

## Bibliography

[Lo05] M.M. Lankhorst and others. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Germany, EU, 2005. ISBN 3540243712

# Part I

# Apendixes

# Appendix A

# Mathematical Notations

The mathematical notations used in the DAVINCI series are explained briefly in this appendix.

## A.1 Sets

In addition to the set operations: $\cup, \cap, \setminus, \subseteq$ with their usual meaning, we also define:

$$\begin{aligned} A \subset B &\triangleq A \subseteq B \wedge A \neq B \\ A \not\subset B &\triangleq \neg A \subset B \end{aligned}$$

The power set of a set $A$, i.e. the set of all subsets of $A$, is denoted as $\wp(A)$, where:

$$\wp(A) \triangleq \{ B \mid B \subseteq A \}$$

## A.2 Functions

A partial function $f$ from $A$ to $B$ is defined by $f : A \rightarrowtail B$. Formally, it is a relation $f \subseteq A \times B$ such that $\langle a, b\rangle \in f \wedge \langle a, c\rangle \in f \Rightarrow b = c$. This property makes it possible to write $f(a) = b$ instead of $\langle a, b\rangle \in f$.

A function $f$ is a set of binary tuples. The first and second values of these binary tuples are identified as:

$$\begin{aligned} \pi_1(f) &\triangleq \{ a \mid \langle a, b\rangle \in f \} \\ \pi_2(f) &\triangleq \{ b \mid \langle a, b\rangle \in f \} \end{aligned}$$

The following abbreviations are used for (partial) functions:

$$\begin{aligned} \mathtt{dom}(f) &\triangleq \pi_1(f) \\ \mathtt{ran}(f) &\triangleq \pi_2(f) \\ f(a)\downarrow &\triangleq a \in \mathtt{dom}(f) \\ f(a)\uparrow &\triangleq a \notin \mathtt{dom}(f) \end{aligned}$$

For unary functions, we will write $f\downarrow a$, and $f\uparrow a$, instead of $f(a)\downarrow$, and $f(a)\uparrow$ respectively. Further, $f_1, \ldots, f_n\downarrow a_1, \ldots, a_m$ is employed as an abbreviation for: $\forall_{1\leq i\leq n} \forall_{1\leq j\leq m} [f_i\downarrow a_j]$.

A total function $f$ from $A$ to $B$ is defined by $f : A \rightarrow B$. Formally, $f : A \rightarrowtail B$ for which $\mathtt{dom}(f) = A$.

## A.3 Relations

If $R \subseteq X \times X$ is a relation, then we will use: $x\, R\, y\, R\, z$ as an abbreviation of: $x\, R\, y \wedge y\, R\, z$.

# Appendix B

# Answers to questions

## B.1 Questions from Chapter 1

Version: 17-01-06

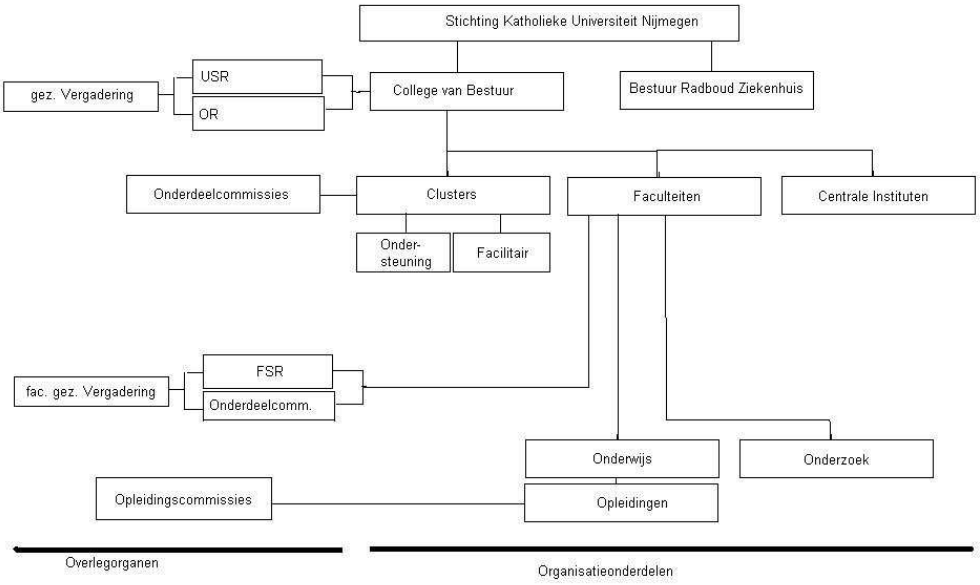
1. What is an organization? Give some examples of groupings of people that are not an organization.

   **Answer:**
   An organization exists of actors which functionally interact with each other and thereby play certain roles. Examples of groups of people who do not form an organization are, for example:

   (a) All passengers in one train.
   (b) A group of cyclists who are waiting for a traffic light.
   (c) Shopping people in a store.

2. Produce a model of the hierarchical structure of a university (faculties, departments, schools, etc). Why is the model organized this way?

   **Answer:**
   A graphical representation of the hierarchical structure of a university, in this specific case the Radboud University Nijmegen, may look like this:



This model looks this way, because:

(a) This way, the university can be considered as a whole in a clear way (holistic).

(b) Organizational parts are clearly separated from the consulting bodies.

(c) The power hierarchy is clearly displayed: the higher the position in the schema, the greater the power is.

3. Produce a model of the educational process of attending a course at a university. What are the contributions of the different elements in this process.

**Answer:**
A sketch of attending a course at a university, written in natural language.

A lecturer offers a course. Students and other interested people enroll in the course. The lecturer presents his lectures. Practical work, which may be part of the course, could be taken care of by either the lecturer, a teachers' assistant or a PhD-student.

During a course a student or other interested person may resign from the course (for several reasons). Those people who stay until the end of the course should have gained some new knowledge and skills. These are tested by the lecturer, probably by a test or an assignment. To transfer the knowledge and skills from a lecturer to the students, a book or reader may be used.

Roles:

**Lecturer:** the person who offers the course, the material used during the course, assignments and tests.

**Student/interested person:** enrolls in the course, learns from the materials supplied by the lecturer and participates in available practical assignments

**Teachers' Assistant/PhD student:** delivers assistance to the practical work which comes with the course.

**Book/reader/college materials:** provide a wide base of knowledge, which supports the lectures of the lecturer.

4. Describe why it is important to realize that organizations can be part of yet other organizations. Use the term 'level of abstraction' in your answer.

**Answer:**
Organizations consist of actors who try to get something done together. An organization can itself be an actor. According to that line of reasoning, an organization can contain other organizations. At a very low level, actors are humans. At some more abstract level, actors are organizations, or organizational parts, which can be actors themselves. This way an organizational hierarchy can be distinguished.

5. If two people were to produce a model of the same organization. Would you expect them to produce the same model? If not, why do you think these models would differ?

**Answer:**
It is to be expected that, when two people are asked to produce a model of an organization, some part of it will be equal. However, another part will be different. This is a consequence of the interpretation of the domain by the modelers and the modeling choices the modeler makes. Concrete examples of such chooses are the used level of abstraction when describing the organization, the used modeling technique or method and the specified border of the organizations' domain.

## B.2 Questions from Chapter 2

Version: 16-03-05

1. How are the terms 'organization', 'domain' and 'universe' be related to each other, given the definitions provided in this textbook?

- Describe this relation in natural language.
- Describe this relation in the formal language given in this chapter.

**Answer:**
In natural language: The universe is the world under investigation by a viewer. A domain is a part or aspect of the universe in which the viewer is interested. A domain always has a direct environment. An organization is positioned in a universe, therefore being also part of a universe, combining at least one domain. So, here, a universe can be seen as the whole, containing organizations. Organizations contain domains.

In formal language: Let $U$ be a universe observed by some viewer $v$. Let furthermore $D$ be a domain with direct environment $E_D$ (as part of a conception $C$) as it is observed by $v$ in $U$. Then we formally have: $U \models^{\mathsf{d}}_v \langle C : E_D : D \rangle$.

The viewer $v$ may see (in the domain $D$) an organizational system $S$ with environment $E_S$. In other words: $U \models^{\mathsf{s}}_v \langle C : E_S : S \rangle$, where $E_S \subseteq E_D \cup D$ and $O \subseteq D$.

2. Proof Axiom **S8** (page 31).
3. Proof Lemma 2.2.1 (page 31).

   **Answer:**
   Let $U \models^{\mathsf{d}}_v \langle C : E : D \rangle$. Suppose $P$ be a maximum connected subset of $E$, while:

   $$\neg\, \exists_{e \in \mathcal{CO}_P, r \in \mathcal{LI}_C, d \in \mathcal{CO}_D} [\{e, d\} = \mathsf{Involved}(r)]$$

   In other words, suppose $P$ is *not* connected to $D$.

   As $P$ is a maximum connected subset of $E$ (and also non-empty), there is no other element in $E$ to which an element from $P$ is connected to. If $P$ is *not* connected to $D$ it would mean that some subset of $E$ is not connected to $D$, which would contradict Axiom **S10**. We can therefore not assume:

   $$\neg\, \exists_{e \in \mathcal{CO}_E, r \in \mathcal{LI}_C, d \in \mathcal{CO}_D} [\{e, d\} = \mathsf{Involved}(r)]$$

   We should therefore conclude:

   $$\exists_{e \in \mathcal{CO}_E, r \in \mathcal{LI}_C, d \in \mathcal{CO}_D} [\{e, d\} = \mathsf{Involved}(r)]$$

4. Not all conceptions of a domain produce models. Why not?

   **Answer:**
   Conceptions need to be described in terms of a description. Conceptions are made by viewers, but are not made on purpose. If a conception is made on purpose, so if a domain is purposely abstracted, it is called a model. Therefore, not all conceptions produce models.
5. Beschouw een Autoproducent, zoals bijvoorbeeld Seat, BMW en Toyota.
   (a) Wat zijn de belangrijkste systemische eigenschappen?
   (b) Beschrijf het primaire gedrag van deze organisatie in termen van interne en externe functies.
6. Give an example of a reactive system, of a responsive system and of an autonomous system (other examples than the ones already given, of course).

   **Answer:**

   - Active System: an example of an active system is the traffic jam information supply system. These are the LED-signs along highways supplying information about which of the highways are jammed, where they are jammed and the length of the jam. Drivers can, based on the supplied information, choose to take a different route than their initial route. This way, the system actively changes the universe.

- Dynamic System: an example of a dynamic system is the Dutch political system. The system keeps on existing, but it changes due to new laws, the abolishing of old laws, etcetera.
- Open System: an example of an open system is a 'smart' chess computer. Consider a chess computer with a set of standard possible sets of movements. Each time you make a move which is unknown to the system, the system remembers the move. After a few games, the chess computer has 'learned' from your input and, as an effect, becomes harder to beat.

7. From a modeling point of view, organizations can be considered as systems containing a.o. concepts and links.
   - Why is it important to be aware of the aspect of subjectivity when creating models?
   - What view does an information system developer have when modeling organizations?
   - Why would an information system developer want to start by creating a model of an organization, instead of directly focusing on modeling an information system?

   **Answer:**

   - A possible logical choice is the 'Machine'-metaphor, since an aspect of that metaphor is 'design'. Another aspect is standardization (by generalization): if a system is described as a collection of concepts and links, these elements can be considered as generalizations of several parts of the system.
   - Models are based on conceptions, which are viewer-bound. Therefore, models are described from the modelers' point of view. These models also include the worldview of the modeler, and are thus a subjective representation of a domain.
   - An information system developer describes a domain from a certain point of view, namely his own point of view, and with a certain purpose, namely to create a model of the domain which on which an information system shall/could be based. So, he only look from one point of view within his own point of view to the domain.
   - By creating a model of an organization before creating a model of a possible information system which should be deployed in the organization, the developer is forced to gain knowledge about the organization in which he operates. This should lead to better insight in several aspects of the organization, like structure, evolution, culture, etcetera. The use of this knowledge when creating an information system should lead to a better 'fitting' system.

8. Waarom zullen verschillende mensen wanneer ze verschillende domeinen modelleren toch verschillende modellen opleveren? Hoe kun je deze situatie verbeteren? Waarom zou je dit willen verbeteren?
9. Stel $U \models^{\text{s}}_{v} S' \subset S$, bewijs/beargumenteer dan dat:

$$|\mathcal{LI}_{S'}| + 1 \geq |\mathcal{CO}_{S'}|$$

10. Beschouw Axiom **S3** en Axiom **S4**. Daarin worden de voorwaarden voor een gesloten, connected graph gegeven. Stel dat we te maken hebben met een niet-connected graph.
    - Is het waarschijnlijk dat we het hier over hetzelfde Universe hebben? Beargumenteer je antwoord.
    - In een connected graph is de relatie tussen het aantal concepten en links $|\mathcal{LI}_C| + 1 \geq |\mathcal{CO}_c|$.

      Kun je ook een soortgelijke relatie onderkennen als een graph niet connected is? Zo ja, welke? Zo nee, waarom niet?

11. Proof Corollary 2.2.1 (page 34).

    **Answer:**
    Because of Axiom **S12**, we already know:

    $$\exists_{u \in \mathcal{CO}_C} \forall_{x \in \mathcal{CO}_C} [u \rightrightarrows_C x]$$

    But is there only one such $u$?

    Suppose we have $u_1, u_2 \in \mathcal{CO}_C$ such that $u_1 \neq u_2$ while:

    $$\forall_{x \in \mathcal{CO}_C} [u_1 \rightrightarrows_C x \wedge u_2 \rightrightarrows_C x]$$

    In other words, both $u_1$ and $u_2$ are tops of the decomposition hierarchy.

    This allows us to derive:

    $$\begin{array}{ll} u_1 \neq u_2 \wedge \forall_{x \in \mathcal{CO}_C} [u_1 \rightrightarrows_C x \wedge u_2 \rightrightarrows_C x] & \Rrightarrow \{\text{Since } u_1, u_2 \in \mathcal{CO}_C\} \\ u_1 \rightrightarrows_C u_2 \wedge u_2 \rightrightarrows_C u_1 & \Rrightarrow \{u_1 \neq u_2\} \\ u_1 \rightarrow_C u_2 \wedge u_2 \rightarrow_C u_1 & \Rrightarrow \{\text{Transitivity of } \_ \rightarrow_C \_\} \\ u_1 \rightarrow_C u_1 & \square \end{array}$$

    As stipulated by Axiom **S11** (page 32), decompositions are acyclic. Therefore, we are noy allowed to have $u_1 \rightarrow_C u_1$. We therefore cannot have $u_1, u_2 \in \mathcal{CO}_C$ such that $u_1 \neq u_2$ while:

    $$\forall_{x \in \mathcal{CO}_C} [u_1 \rightrightarrows_C x \wedge u_2 \rightrightarrows_C x]$$

    Therefore there can only be one such $u$.

12. Suppose you are requested by a large organization (a holding company holding some daughter companies) to create more insight into their own activities by creating some models of their organization. The focus of this models must, according to the board of directors, be on their internal information flows, since the organization has the impression that a lot of business efficiency is lost due to an incompetent set of information systems. Keeping in mind what is explained in the two previous chapters, give an impression of:
    (a) Where would you start modeling?
    (b) What would you model?
    (c) Why model that?

    **Answer:**

    (a) When a model of a very large organization is required, one should start by focusing on the problem, instead of focusing on a to-be-created model. The first logical step is gaining some knowledge of the whole of the organization. This can indeed be done by creating a (holistic) sketch of the complete organization.
    (b) When that sketch is made, and if the board agrees on that sketch, the modeler can start focusing on several aspects concerning information flows. Perhaps an iterative approach might be useful: expanding a model step by step, keeping the whole of the organization in mind.
    (c) Before an analysis can be made of a problem, or before a problem can be identified or localized, some research has to be done. A board of directors can have some ideas of where the problem may lie, but they may be wrong as well. Simply following orders of some board may then result in the wrong conception of the organization, and thus to a correct solution for some problem, but not to a solution for their problem. In fact, the solution then does not 'fit', or it changes, making it perhaps even more difficult to find. Therefore, a model of the whole should be made: to gain insight in the organization and their problem.

13. Proof Corollary 2.3.1 (page 38).

**Answer:**
Suppose $U \models^{\mathsf{s}}_{v} S' \subset S$, then:

$$
\begin{array}{ll}
U \models^{\mathsf{s}}_{v} S' \subset S & \Rrightarrow \{\text{definition of sub-system}\} \\
U \models^{\mathsf{s}}_{v} S' & \Rrightarrow \{\text{definition of abbreviation}\} \\
\exists_{C,E}\,[U \models^{\mathsf{s}}_{v} \langle C : E : S' \rangle] & \Rrightarrow \{\text{Axiom } \mathbf{S19}\} \\
\exists_{C,E}\,[U \models^{\mathsf{m}}_{v} \langle C : E : S' \rangle] & \Rrightarrow \{\text{Axiom } \mathbf{S18}\} \\
\exists_{C,E}\,\left[U \models^{\mathsf{d}}_{v} \langle C : E : S' \rangle\right] & \Rrightarrow \{\text{Axiom } \mathbf{S13}\} \\
\exists_{d \in S'} \forall_{x \in \mathcal{CO}_{S'}}\,[d \rightrightarrows_{S'} x] & \Box
\end{array}
$$

14. Consider a home cinema set.
    (a) Describe the systems elements.
    (b) Distinguish proper sub-systems.
    (c) Can you derive typical aspect systems and component systems?

    Explain your answers.

    **Answer:**

    (a) A home cinema set usually contains of some elements of input (a receiver, a DVD-player, a VCR, a game-console, a remote control, etcetera), and some elements of output (a television screen, and some speakers). The links between these elements are the cables and the radio- or infra-red signals broadcasted by the RC.
    (b) possible sub-systems are: the receiver, the dvd-player, the VCR, a game-sonsole, a remote-control, a television, a set of speakers, the video-stream, the audio-stream, the input-streams, the output-streams, and the power-supply.
    (c) A list of proper component sub-systems is already given in the first item. Each of these elements can properly function in other settings, and can thus be considered as component sub-system. When looking at aspect systems, input-systems, output-systems, video-stream system, audio-stream system and the power-supply. Please recognize the difference between a speaker and a set of speakers. A speaker can be identified as a sub-system itself, but since a set of speakers (for example in a Dolby 5.1 setting) has certain properties which individual speakers do not have (surround sound), a set of speakers can be considered as a sub-system just as well as a single speaker.

15. Consider a travel agency.
    (a) Describe the most important system characteristics and exposition characteristics.
    (b) Describe its behavior in terms of internal and external functions.

    **Answer:**

    (a) Consider a travel agency as an open active system: it anticipates to a lot of factors in its environment, like customers, travels offered by airline companies, etcetera. On the other hand, large travel agencies can promote certain destinations by promotion, or enforce a quantity rebate at some airline company or hotel. This way they have influence on the world in which they function.
        - system characteristics: The agency is thus considered as an open active system.

- exposition characteristics: The travel agency mainly plays three roles: the customer role (buying tickets at hotels and airline agencies), the vendor-role (selling travels to customers) and the mediator-role (searching the best fitting travel for each customer)

(b)
- Internal functions: examples of internal functions are: the management of the agency, the administration of the agency, etcetera.
- External functions: examples of external functions are: the number of customers interested in traveling, the supply of possible travels by hotels and airline companies, etcetera.

16. Describe why *information* systems contain *data*bases. Use the descriptions of the terms data, information and knowledge as described in Chapter 2 in your description.

    **Answer:**
    Databases contain loads of data. Data, which is only useful when placed in some context and when the data is related to each other in some way. This is usually done by queries built in an information system. The system gains some data from a database and displays it in context to the user of the system. By placing it in context, the data becomes information. Due to the information supplied by the system, the view on the world of the systems' user may be changed (or in other words: it may change the knowledge of the user).

17. Give some examples of:
    (a) Work systems that are not organizational systems.
    (b) Organizational systems.
    (c) Information systems.

18. Describe, in your own words, the differences between knowledge, information and data.

    **Answer:**
    Data are loose building blocks, for example one term or one fact. They are quite meaningless without any context. They are the smallest units of which information can be constructed. Information is data placed in some context, thereby giving it some meaning. This meaning may have influence on someone's knowledge. Knowledge represents someones view on the world.

19. Proof Corollary 2.4.1 (page 43).

    **Answer:**

Suppose $t_1 \rhd t_2 \wedge t_1 \rhd t_3 \wedge t_2 \neq t_3$, then this would lead to the following contradiction:

$$
\begin{aligned}
& t_1 \rhd t_2 \wedge t_1 \rhd t_3 \wedge t_2 \neq t_3 \\
& \quad \Rrightarrow \{\text{since } < \text{ is a complete total order}\} \\
& t_1 \rhd t_2 \wedge t_1 \rhd t_3 \wedge (t_2 < t_3 \vee t_3 < t_2) \\
& \quad \Rrightarrow \{\text{definition of } \rhd\} \\
& t_1 < t_3 \wedge \neg \exists_s [t_1 < s < t_2] \wedge t_1 < t_3 \wedge \neg \exists_s [t_1 < s < t_3] \wedge (t_2 < t_3 \vee t_3 < t_2) \\
& \quad \Rrightarrow \{\text{rewrite}\} \\
& \neg \exists_s [t_1 < s < t_2] \wedge \neg \exists_s [t_1 < s < t_3] \wedge (t_1 < t_2 < t_3 \vee t_1 < t_3 < t_2) \\
& \quad \Rrightarrow \{\text{rewrite}\} \\
& \neg \exists_s [t_1 < s < t_2 \vee t_1 < s < t_3] \wedge (t_1 < t_2 < t_3 \vee t_1 < t_3 < t_2) \\
& \quad \Rrightarrow \{\text{contradiction}\} \\
& \text{FALSUM} \\
& \square
\end{aligned}
$$

Therefore, if $t_1 \rhd t_2 \wedge t_1 \rhd t_3$ it cannot be that $t_2 \neq t_3$. In other words, we must have $t_2 = t_3$.

20. Proof Corollary 2.4.2 (page 44).

**Answer:**
We provide the prove of $H_E(t)(t) \subseteq H_C(t)$. The prove for $H_E(t)(t) \subseteq H_C(t)$ goes analogously.
Suppose $U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle$, then:

$$
\begin{aligned}
& U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle && \Rrightarrow \{H_E \text{ is a function to } \wp\mathcal{O}(H_C)\} \\
& H_E(t) \in \wp\mathcal{O}(H_C) && \Rrightarrow \{\text{definition of powerset}\} \\
& H_E(t) \subseteq H_C && \Rrightarrow \{\text{rewrite}\} \\
& h \in H_E(t) \Rightarrow h \in H_C(t) && \Rrightarrow \{h \text{ is a function } h : \mathcal{TI} \rightarrowtail \mathcal{EL}\} \\
& \left\{ h(t) \;\middle|\; h \in H_E(t) \right\} \subseteq \left\{ h(t) \;\middle|\; h \in H_C \right\} && \Rrightarrow \{\text{definition of shorthand } H(t)\} \\
& H_E(t)(t) \subseteq H_C(t) && \square
\end{aligned}
$$

21. Proof Corollary 2.4.3 (page 44).

**Answer:**
Let $U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle$, then:

$$
\begin{aligned}
& U \models^{\mathsf{d}}_v \langle H_C : H_E : H_D \rangle && \Rrightarrow \{\text{Axiom } \textbf{S21} \text{ (page 44)}\} \\
& \forall_{t \in \mathcal{TI}} \left[ U \models^{\mathsf{d}}_v \langle H_C(t) : H_E(t)(t) : H_D(t)(t) \rangle \right] && \Rrightarrow \{\text{Axiom } \textbf{S6} \text{ (page 31)}\} \\
& \forall_{t \in \mathcal{TI}} \left[ H_E(t)(t) \cap H_D(t)(t) = \emptyset \right] && \square
\end{aligned}
$$

22. Proof Lemma 2.4.1 (page 44).

**Answer:**
Let $U \models^{\mathrm{d}}_{v} \langle H_C : H_E : H_D \rangle$, while $\neg\, \forall_{t \in \mathcal{TI}} [H_E(t) \cap H_D(t) = \emptyset]$, then:

$$\begin{aligned}
&\neg\, \forall_{t \in \mathcal{TI}} [H_E(t) \cap H_D(t) = \emptyset] && \Rrightarrow \{\text{rewrite}\}\\
&\exists_{t \in \mathcal{TI}} [H_E(t) \cap H_D(t) \neq \emptyset] && \Rrightarrow \{\text{rewrite}\}\\
&\exists_{t \in \mathcal{TI}}\, \exists_h [h \in H_E(t) \cap H_D(t)] && \Rrightarrow \{\text{rewrite}\}\\
&\exists_{t \in \mathcal{TI}}\, \exists_h [h \in H_E(t) \wedge h \in H_D(t)] && \Rrightarrow \{\text{rewrite}\}\\
&\exists_{t \in \mathcal{TI}}\, \exists_h [h(t) \in H_E(t)(t) \wedge h(t) \in H_D(t)(t)] && \Rrightarrow \{\text{rewrite}\}\\
&\exists_{t \in \mathcal{TI}}\, \exists_e [e \in H_E(t)(t) \wedge e \in H_D(t)(t)] && \Rrightarrow \{\text{rewrite}\}\\
&\exists_{t \in \mathcal{TI}} [H_E(t)(t) \cap H_D(t) \neq \emptyset] && \square
\end{aligned}$$

This result would contradict Corollary 2.4.3. In other words, we must have $\forall_{t \in \mathcal{TI}} [H_E(t) \cap H_D(t) = \emptyset]$.

## B.3 Questions from Chapter 3

1. Given the situation:

   > A person with name Erik is writing a letter to his loved one, at the desk in a romantically lit room, on a mid-summer's day, using a pencil, while the cat is watching.

   Produce a graph consisting of concepts and links depicting this domain.
2. Stel je maakt een ontwerp voor een geldautomaat. Wat zijn voor dat domein de belangrijkste concepten en hun onderlinge links? Hoe werken ze samen?
3. Proof Corollary 3.4.1 (page 53).

   **Answer:**
   We will prove $\mathcal{FC}_t = \mathsf{Fact}(\mathcal{RO}_t)$. The prove of $\mathcal{PC}_t = \mathsf{Player}(\mathcal{RO}_t)$ goes analogously.

$$\begin{aligned}
&f \in \mathsf{Fact}(\mathcal{RO}_t) && \equiv \{\text{Definition of } \mathsf{Fact}\}\\
&\exists_{r \in \mathcal{RO}_t} [f = \mathsf{Fact}(r)] && \equiv \{\text{Axiom } \mathbf{S29}\}\\
&\exists_{r \in \mathcal{RO}_t} [f = \mathsf{Fact}(r) \wedge \mathsf{Fact}(r) \in \mathcal{FC}_t] && \equiv \{\text{Rewrite}\}\\
&\exists_{r \in \mathcal{RO}_t} [f = \mathsf{Fact}(r) \wedge f \in \mathcal{FC}_t] && \equiv \{\text{Rewrite}\}\\
&\exists_{r \in \mathcal{RO}_t} [f = \mathsf{Fact}(r)] \wedge f \in \mathcal{FC}_t && \equiv \{\text{Definition of } \mathcal{FC}\}\\
&f \in \mathcal{FC} \wedge f \in \mathcal{FC}_t && \equiv \{\text{Definition of } \mathcal{FC}_t\}\\
&f \in \mathcal{FC}_t && \square
\end{aligned}$$

4. Proof Corollary 3.5.1 (page 55).

   **Answer:**

$$\begin{aligned}
&x \in \mathcal{IN} && \Rrightarrow \{\text{Definition of } \mathcal{IN}\}\\
&x \in \{x \mid \exists_y [x\ \mathsf{HasType}\ y]\} && \Rrightarrow \{\text{Rewrite}\}\\
&\exists_y [x\ \mathsf{HasType}\ y] && \Rrightarrow \{\text{Definition of } \mathcal{TP}\}\\
&\exists_{y \in \mathcal{TP}} [x\ \mathsf{HasType}\ y] && \square
\end{aligned}$$

Therefore: $\forall_{x \in \mathcal{IN}} \exists_{y \in \mathcal{TP}} [x \, \mathsf{HasType} \, y]$

5. Proof Corollary 3.5.2 (page 55).

   **Answer:**
   We will prove: $\hat{\mathcal{PL}} \triangleq \mathsf{Player}(\hat{\mathcal{RO}})$, the other proofs are analogously.

$$
\begin{array}{ll}
p \in \hat{\mathcal{PL}} & \equiv \left\{\text{Definition of } \hat{\mathcal{PL}}\right\} \\
p \in \mathcal{PL} \wedge p \in \mathcal{TP} & \equiv \{\text{Definition of } \mathcal{PL}\} \\
p \in \mathsf{Player}(\mathcal{RO}) \wedge p \in \mathcal{TP} & \equiv \{\text{Definition of } \mathsf{Player}\} \\
\exists_{r \in \mathcal{RO}} [p = \mathsf{Player}(r) \wedge p \in \mathcal{TP}] & \equiv \{\text{Rewrite}\} \\
\exists_{r \in \mathcal{RO}} [p = \mathsf{Player}(r) \wedge \mathsf{Player}(r) \in \mathcal{TP}] & \equiv \{\text{Axiom } \mathbf{S34}\} \\
\exists_{r \in \mathcal{RO}} [p = \mathsf{Player}(r) \wedge r \in \mathcal{TP}] & \equiv \{\text{Rewrite}\} \\
\exists_{r \in \mathcal{RO} \cap \mathcal{TP}} [p = \mathsf{Player}(r)] & \equiv \left\{\text{Definition of } \hat{\mathcal{RO}}\right\} \\
\exists_{r \in \hat{\mathcal{RO}}} [p = \mathsf{Player}(r)] & \equiv \{\text{Definition of } \mathsf{Player}\} \\
\mathsf{Player}(\hat{\mathcal{RO}}) & \square
\end{array}
$$

   Note: for the Fact versions Axiom **S33** should be used rather than Axiom **S34**.

6. Proof Corollary 3.5.3 (page 56).

   **Answer:**
   Let $f \in \hat{\mathcal{FC}}$, then:

$$
\begin{array}{ll}
i \in \mathsf{Pop}(f) \wedge f \in \hat{\mathcal{FC}} & \equiv \{\text{Definition of } \mathcal{FC}\} \\
\exists_r [\mathsf{Fact}(r) = f \wedge i \in \mathsf{Pop}(f)] & \equiv \{\text{Definition of } \mathsf{RolesOf}\} \\
\exists_{r \in \mathsf{RolesOf}(f)} [i \in \mathsf{Pop}(\mathsf{Fact}(r))] & \equiv \{\text{Axioms } \mathbf{S37} \text{ and } \mathbf{S38}\} \\
\exists_{r \in \mathsf{RolesOf}(f)} [i \in \mathsf{Fact}(\mathsf{Pop}(r))] & \equiv \{\text{Definition of } \mathsf{Pop}\} \\
i \in \mathsf{Fact}(\mathsf{Pop}(\mathsf{RolesOf}(f))) & \square
\end{array}
$$

7. Proof Corollary 3.5.4 (page 56).

   **Answer:**

Let $f \in \check{\mathcal{FC}}$, then:

$$
\begin{aligned}
& x \in \mathsf{Types}(\mathsf{RolesOf}(f)) && \equiv \{\text{Definition of Types}\} \\
& x \in \hat{\mathcal{TP}} \wedge \exists_{r \in \mathsf{RolesOf}(f)} [x \in \mathsf{Types}(r)] && \equiv \{\text{Rewrite}\} \\
& x \in \hat{\mathcal{TP}} \wedge \exists_{r \check{\mathcal{RO}}} [x \in \mathsf{Types}(r) \wedge r \in \mathsf{RolesOf}(f)] && \equiv \{\text{Definition of RolesOf}\} \\
& x \in \hat{\mathcal{TP}} \wedge \exists_{r} [x \in \mathsf{Types}(r) \wedge \mathsf{Fact}(r) = f] && \equiv \{\text{Since } r \in \check{\mathcal{RO}} \text{ and Axiom } \mathbf{S33}\} \\
& x \in \hat{\mathcal{RO}} \wedge \exists_{r} [x \in \mathsf{Types}(r) \wedge \mathsf{Fact}(r) = f] && \equiv \{\text{Definitions of Types and Pop}\} \\
& x \in \hat{\mathcal{RO}} \wedge \exists_{r} [r \in \mathsf{Pop}(x) \wedge \mathsf{Fact}(r) = f] && \equiv \{\text{Definitions of Fact}\} \\
& x \in \hat{\mathcal{RO}} \wedge f \in \mathsf{Fact}(\mathsf{Pop}(x)) && \equiv \{\text{Axioms } \mathbf{S37} \text{ and } \mathbf{S38}\} \\
& x \in \hat{\mathcal{RO}} \wedge f \in \mathsf{Pop}(\mathsf{Fact}(x)) && \equiv \{\text{Definition of Types and Pop}\} \\
& x \in \hat{\mathcal{RO}} \wedge \mathsf{Fact}(x) \in \mathsf{Types}(f) && \equiv \{\text{Definition of RolesOf}\} \\
& x \in \mathsf{RolesOf}(\mathsf{Types}(f)) && \square
\end{aligned}
$$

8. Proof Corollary 3.6.1 (page 58).
9. Consider the following case:

   Een onderneming produceert en verkoopt een tiental soorten gevulde chocoladeartikelen. De verkoop geschiedt aan grossiers tegen prijzen die voor lange tijd vast zijn. In verband met achteruitgang in kwaliteit wordt op de verpakking een uiterste verkoopdatum vermeld. Alle afleveringen geschieden met eigen auto's.

   Voor de produktie van chocolade importeert de inkoopafdeling van de onderneming verschillende soorten cacaobonen uit tropische landen. Daartoe worden inkoopcontracten afgesloten die de behoefte voor ca. een half jaar dekken. De cacaobonenprijs is aan sterke schommelingen onderhevig. De ingekochte partijen hebben belangrijk uiteenlopende vetgehaltes, hetgeen mede in de inkoopprijs tot uitdrukking komt.

   De cacaobonen ondergaan afzonderlijk per partij in de voorbewerkingsafdeling enkele machinale bewerkingen, zoals zuiveren, schillen, breken, branden, malen en walsen.

   Aan het onstane halffabrikaat worden door de afwerkingsafdeling suiker, smaakstoffen en – in verhouding tot het vetgehalte – cacaoboter toegevoegd. Het aldus verkregen halffabrikaat is cacaomassa van een bepaalde standaardkwaliteit, dat in speciaal daartoe geconditioneerde opslagtanks wordt bewaard. De verschillende benodigde vulsels worden ingekocht bij derden. Naar rato van de ontwikkeling van de verkoop en de gewenste voorraadvorming worden de eindprodukten gemaakt. Dit geschiedt in één arbeidsgang met behulp van automatische vorm-, vul- en droogmachines.

   In de pakafdeling worden de goedkopere soorten gevulde chocolade automatisch en de duurdere soorten met de hand in sierdozen verpakt, waarna opslag in een magazijn volgt. Bij alle bewerkingen ontstaan gewichtsverliezen.

   In verband met de kwaliteitsachteruitgang kunnen de grossiers de niet tijdig door hen verkochte artikelen retourneren, mits dit gebeurt binnen 10 dagen na de uiterste verkoopdatum; meestal geschiedt deze teruglevering via de chauffeurs. De teruggenomen artikelen worden vernietigd. Creditering vindt plaats voor 20 van de door hen betaalde prijs. Verrekening hiervan geschiedt slechts bij gelijktijdige nieuwe afname.

Elk van de artikelen is voorzien van een of twee cadeaubonnen, afgedrukt op de verpakking. De waarde van deze bonnen is €0,10 per stuk. Op de artikelen met een prijs tot €5,- komt één, op de overige artikelen (tussen €5,- en €11,-) komen twee bonnen voor. Op deze bonnen kunnen cadeau-artikelen (hand- en theedoeken e.d.) zonder bijbetaling worden verkregen.

Voorts kunnen op deze bonnen meer duurzame gebruiksgoederen tegen verlaagde prijs worden verkregen. Hiervoor wordt elk halfjaar een folder uitgegeven, waarin per artikel is aangegeven hoeveel bonnen moeten worden ingeleverd en hoeveel daarnaast moet worden bijbetaald. In het algemeen is het door de afnemers bij te betalen bedrag iets lager dan de inkoopprijs voor de fabriek. Veelal dient de halfjaarlijkse behoefte door de fabrikant in één keer te worden besteld; latere aanvulling is in het algemeen niet mogelijk.

Op de duurzame gebruiksgoederen wordt veelal garantie of service verleend. Hiervoor is met een gespecialiseerd bedrijf een contract afgesloten waarbij tegen een eenmalig vast bedrag per apparaat de garantie- en serviceverplichtingen worden overgedragen

Answer the following questions:

(a) Produce elementary facts for this domain.

(b) Produce an ORM model for this domain.

## B.4 Questions from Chapter 5

1. Given the following populations: Pop(Carnivore) $= \{a, b, c\}$, Pop(Omnivore) $= \{d, e\}$ and Pop(Herbivore) $= \{f, g\}$. What are the populations of Animal, Flesh eater and Plant eater?

2. To have electrical power supplied to one's premises (i.e. building and grounds), an application must be lodged with the Electricity Board. The following tables are extracted from an information system used to record details about any premises for which power has been requested.

   The following abbreviations are used: *premises# = premises number*, *qty = quantity*, *nr = number*, *commercl = commercial*. Each premises is identified by its premises#.

   The electricity supply requested is exactly one of three kinds: "new" (new connection needed), "modify" (modifications needed to existing connection), or "old" (reinstall old connection). "Total amps" is the total electric current measured in Amp units. "Amps/phase" is obtained by dividing the current by the number of phases.

| premises# | city | kind of premises | kind of business | dog on premises | breed of dog | qty of breed | supply needed |
|---|---|---|---|---|---|---|---|
| 101 | Brisbane | domestic | . | yes | Terrier | 2 | new |
| 202 | Brisbane | commercl | car sales | no | . | . | modify |
| 303 | Ipswich | domestic | . | yes | Alsatian | 1 | old |
| | | | | | Poodle | 1 | |
| 404 | Redcliffe | commercl | security | yes | Alsatian | 3 | new |
| | | | | | Bulldog | 2 | |
| 505 | Brisbane | domestic | . | no | . | . | modify |
| 606 | Redcliffe | commercl | bakery | no | . | . | old |
| ... | ... | ... | ... | ... | ... | ... | ... |

   Further details about new connections or modifications:

| premises# | load applied for (if known) | | | wiring completed? | expected date for wiring completion |
|---|---|---|---|---|---|
| | total amps | nr phases | amps/phase | | |
| 101 | 200 | 2 | 100 | no | 30-06-03 |
| 202 | 600 | 3 | 200 | yes | . |
| 404 | . | . | . | no | 01-08-03 |
| 505 | 160 | 2 | 80 | no | 30-06-03 |
| ... | ... | ... | ... | ... | |

The population is significant with respect to mandatory roles. Each premises has at most two breeds of dog.

Produce a fact-based model for this domain. Use specialization when needed. Include *uniqueness*, *mandatory role*, *subset*, *occurrence frequency* and *equality constraints*, as well as *value type constraints* that are relevant. Provide meaningful names.

If a fact type is derived it should be asterisked on the diagram and a derivation rule should be supplied.

Produce both a flat fact-based model, as well as a version that uses abstraction/decomposition to split this domain into more comprehensible chunks.

## B.5 Questions from Chapter 7

1. Given the situation:

   > A person with name Erik is writing a letter to his loved one, at the desk in a romantically lit room, on a mid-summer's day, using a pencil, while the cat is watching.

   Produce a graph consisting of entities and relationships depicting this domain.

2. Consider the following domain:

   > Docent Proper voert de vakgegevens van Architectuur en Alignment in in het management informatiesysteem.

   Wat zijn hier de entiteiten en de relaties? Wat zijn de acties, actoren, actanden en predications.?

3. Stel je maakt een ontwerp voor een geldautomaat. Wat zijn voor dat domein de belangrijkste systeem entiteiten en hun onderlinge relaties? Hoe werken ze samen? Wat zijn hier de entiteiten en de relaties? Wat zijn de acties, actoren, actanden en predications.?

4. Proof Corollary 3.4.1 (page 53).

5. Proof Corollary 3.5.1 (page 55).

6. Consider the case from Question 3.9.

   Answer the following questions:

   (a) (Re)produce elementary facts for this domain.
   (b) What are the actions, actors and actands?

# Bibliography


[Ack71] R.L. Ackoff. Towards a System of System Concepts. *Management Science*, 17, July 1971.

[AH87] S. Abiteboul and R. Hull. IFO: A Formal Semantic Database Model. *ACM Transactions on Database Systems*, 12(4):525–565, December 1987.

[Alt99] S. Alter. A general, yet useful theory of information systems. *Communications of the Association for Information Systems*, 1(13), 1999.
`http://cais.isworld.org/articles/1-13/default.asp`

[Alt02] S. Alter. The work system method for understanding information systems and information system research. *Communications of the Association for Information Systems*, 9(9):90–104, 2002.
`http://cais.isworld.org/articles/default.asp?vol=9&art=6`

[Avi95] D.E. Avison. *Information Systems Development: Methodologies, Techniques and Tools*. McGraw–Hill, New York, New York, USA, 2nd edition, 1995. ISBN 0077092333

[BB97] F.C. Berger and P. van Bommel. Augmenting a Characterization Network with Semantical Information. *Information Processing & Management*, 33(4):453–479, 1997.

[BBMP95] G.H.W.M. Bronts, S.J. Brouwer, C.L.J. Martens, and H.A. (Erik) Proper. A Unifying Object Role Modelling Approach. *Information Systems*, 20(3):213–235, 1995.

[BCN92] C. Batini, S. Ceri, and S.B. Navathe. *Conceptual Database Design – An Entity–Relationship Approach*. Benjamin Cummings, Redwood City, California, USA, 1992.

[Bem98] T.M.A. Bemelmans. *Bestuurlijke Informatiesystemen en Automatisering*. Kluwer, Deventer, The Netherlands, EU, 7th edition, 1998. In Dutch. ISBN 9026727984

[Ber01] L. von Bertalanffy. *General Systems Theory – Foundations, Development, Applications*. George Braziller, New York, New York, USA, revised edition, 2001. ISBN 0807604534

[BFW96] P. van Bommel, P.J.M. Frederiks, and Th.P. van der Weide. Object–Oriented Modeling based on Logbooks. *The Computer Journal*, 39(9):793–799, 1996.

[BH96] A.C. Bloesch and T.A. Halpin. ConQuer: A Conceptual Query Language. In B. Thalheim, editor, *Proceedings of the 15th International Conference on Conceptual Modeling (ER'96), Cottbus, Germany, EU*, volume 1157 of *Lecture Notes in Computer Science*, pages 121–133, Berlin, Germany, EU, October 1996. Springer.

[BHW91] P. van Bommel, A.H.M. ter Hofstede, and Th.P. van der Weide. Semantics and verification of object–role models. *Information Systems*, 16(5):471–495, October 1991.

[BPH04] A.I. Bleeker, H.A. (Erik) Proper, and S.J.B.A. Hoppenbrouwers. The Role of Concept Management in System Development – A practical and a theoretical perspective. In J. Grabis, A. Persson, and J. Stirna, editors, *Forum proceedings of the 16th Conference on*

*Advanced Information Systems 2004 (CAiSE 2004), Riga, Latvia, EU*, pages 73–82, Riga, Latvia, EU, June 2004. Faculty of Computer Science and Information Technology. ISBN 998497670X

[BRJ99] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modelling Language User Guide*. Addison Wesley, Reading, Massachusetts, USA, 1999. ISBN 0201571684

[Bub86] J.A. Bubenko. Information System Methodologies – A Research View. In T.W. Olle, H.G. Sol, and A.A. Verrijn–Stuart, editors, *Information Systems Design Methodologies: Improving the Practice, Amsterdam, The Netherlands, EU*, pages 289–318. North–Holland/IFIP WG8.1, Amsterdam, The Netherlands, EU, 1986.

[BW90] P.D. Bruza and Th.P. van der Weide. Assessing the Quality of Hypertext Views. *ACM SIGIR FORUM (Refereed Section)*, 24(3):6–25, 1990.

[BW91] P.D. Bruza and Th.P. van der Weide. The Modelling and Retrieval of Documents using Index Expressions. *ACM SIGIR FORUM (Refereed Section)*, 25(2), 1991.

[BW92a] P. van Bommel and Th.P. van der Weide. Reducing the search space for conceptual schema transformation. *Data & Knowledge Engineering*, 8:269–292, 1992.

[BW92b] P.D. Bruza and Th.P. van der Weide. Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[Che76] P.P. Chen. The Entity–Relationship Model: Towards a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[Che81] P. Checkland. *Systems thinking, systems practice*. John Wiley & Sons, New York, New York, USA, 1981. ISBN 0471279110

[CHP96] L.J. Campbell, T.A. Halpin, and H.A. (Erik) Proper. Conceptual Schemas with Abstractions – Making flat conceptual schemas more comprehensible. *Data & Knowledge Engineering*, 20(1):39–85, 1996.

[Coh89] B. Cohen. Justification of Formal Methods for System Specification. *Software Engineering Journal*, 4(1):26–35, January 1989.

[CP96] P.N. Creasy and H.A. (Erik) Proper. A Generic Model for 3–Dimensional Conceptual Modelling. *Data & Knowledge Engineering*, 20(2):119–162, 1996.

[EGH+92] G. Engels, M. Gogolla, U. Hohenstein, K. Hülsmann, P. Löhr-Richter, G. Saake, and H.-D. Ehrich. Conceptual modelling of database applications using an extended ER model. *Data & Knowledge Engineering*, 9(4):157–204, 1992.

[EKW92] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object–Oriented Systems Analysis – A model–driven approach*. Yourdon Press, New York, New York, USA, 1992. ASIN 0136299733

[EN94] R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems*. Benjamin Cummings, Redwood City, California, USA, 1994. Second Edition.

[EWH85] R. Elmasri, J. Weeldreyer, and A. Hevner. The category concept: An extension to the entity–relationship model. *Data & Knowledge Engineering*, 1:75–116, 1985.

[Fre97] P.J.M. Frederiks. *Object–Oriented Modeling based on Information Grammars*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1997. ISBN 9090103384

[FVV+98] E.D. Falkenberg, A.A. Verrijn–Stuart, K. Voss, W. Hesse, P. Lindgreen, B.E. Nilsson, J.L.H. Oei, C. Rolland, and R.K. and Stamper, editors. *A Framework of Information Systems Concepts*. IFIP WG 8.1 Task Group FRISCO, IFIP, Laxenburg, Austria, EU, 1998. ISBN 3901882014

[FW02] P.J.M. Frederiks and Th.P. van der Weide. Deriving and paraphrasing information grammars using object–oriented analysis models. *Acta Informatica*, 38(7):437–88, June 2002.

[FW04a] P.J.M. Frederiks and Th.P. van der Weide. Information Modeling: the process and the required competencies of its participants. *Data & Knowledge Engineering*, 2004. To appear in a special issue on the NLDB 2004 conference.

[FW04b] P.J.M. Frederiks and Th.P. van der Weide. Information Modeling: the process and the required competencies of its participants. In F. Meziane and E. Métais, editors, *9th International Conference on Applications of Natural Language to Information Systems (NLDB 2004), Manchester, United Kingdom, EU*, volume 3136 of *Lecture Notes in Computer Science*, pages 123–134, Berlin, Germany, EU, 2004. Springer.

[Hal95] T.A. Halpin. *Conceptual Schema and Relational Database Design*. Prentice–Hall, Englewood Cliffs, New Jersey, USA, 2nd edition, 1995.

[Hal01] T.A. Halpin. *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Mateo, California, USA, 2001. ISBN 1558606726

[HBP05] S.J.B.A. Hoppenbrouwers, A.I. Bleeker, and H.A. (Erik) Proper. Facing the Conceptual Complexities in Business Domain Modeling. *Computing Letters*, 1(2):59–68, 2005.

[HL89] I. van Horenbeek and J. Lewi. *Algebraic specifications in software engineering: an introduction*. Springer, Berlin, Germany, EU, 1989.

[Hof93] A.H.M. ter Hofstede. *Information Modelling in Data Intensive Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1993.

[Hop03] S.J.B.A. Hoppenbrouwers. *Freezing Language; Conceptualisation processes in ICT supported organisations*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 2003. ISBN 9090173188

[HP95] T.A. Halpin and H.A. (Erik) Proper. Subtyping and Polymorphism in Object–Role Modelling. *Data & Knowledge Engineering*, 15:251–281, 1995.

[HP98] A.H.M. ter Hofstede and H.A. (Erik) Proper. How to Formalize It? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 40(10):519–540, October 1998.

[HPW93] A.H.M. ter Hofstede, H.A. (Erik) Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[HPW97] A.H.M. ter Hofstede, H.A. (Erik) Proper, and Th.P. van der Weide. Exploiting Fact Verbalisation in Conceptual Information Modelling. *Information Systems*, 22(6/7):349–385, September 1997.

[HVH97] J.J.A.C. Hoppenbrouwers, B. van der Vos, and S.J.B.A. Hoppenbrouwers. NL Structures and Conceptual Modelling: Grammalizing for KISS. *Data & Knowledge Engineering*, 23(1):79–92, 1997.

[HW92] A.H.M. ter Hofstede and Th.P. van der Weide. Formalisation of techniques: chopping down the methodology jungle. *Information and Software Technology*, 34(1):57–65, January 1992.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[HW94] A.H.M. ter Hofstede and Th.P. van der Weide. Fact Orientation in Complex Object Role Modelling Techniques. In T.A. Halpin and R. Meersman, editors, *Proceedings of the First International Conference on Object–Role Modelling (ORM–1)*, pages 45–59, July 1994.

[HW97] A.H.M. ter Hofstede and Th.P. van der Weide. Deriving Identity from Extensionality. *International Journal of Software Engineering and Knowledge Engineering*, 8(2):189–221, June 1997.

[IEE00] Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471–2000, The Architecture Working Group of the Software Engineering Committee, Standards Department, IEEE, Piscataway, New Jersey, USA, September 2000. ISBN 0738125180
`http://www.ieee.org`

[Iiv83] J. Iivari. Contributions to the theoretical foundations of systemeering research and the PIOCO model. Technical Report 150, University of Oulu, Oulu, Finland, EU, 1983. ISBN 9514215435

[JLB+04] H. Jonkers, M.M Lankhorst, R. van Buuren, S.J.B.A. Hoppenbrouwers, M. Bonsangue, and L. van der Torre. Concepts for Modeling Enterprise Architectures. *International Journal of Cooperative Information Systems*, 13(3):257–288, 2004.

[Jon86] C.B. Jones. *Systematic Software Development using VDM*. Prentice–Hall, Englewood Cliffs, New Jersey, USA, 1986.

[Kri94] G. Kristen. *Object Orientation – The KISS Method, From Information Architecture to Information System*. Addison Wesley, Reading, Massachusetts, USA, 1994. ISBN 0201422999

[Lan71] B. Langefors. *Editorial notes to: Computer Aided Information Systems Analysis and Design*. Studentlitteratur, Lund, Sweden, EU, 1971.

[Lev79] A.Y. Levy. *Basic Set Theory*. Springer, Berlin, Germany, EU, 1979.

[Lin92] P. Lindgreen. A General Framework for Understanding Semantic Structures. In E.D. Falkenberg, C. Rolland, and E.N. El Sayed, editors, *Information System Concepts: Improving the understanding – Proceedings of the second IFIP WG8.1 working conference (ISCO–2), Alexandria, Egypt*, Amsterdam, The Netherlands, EU, April 1992. North–Holland/IFIP WG8.1. ISBN 0444895078

[Lo05] M.M. Lankhorst and others. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer, Berlin, Germany, EU, 2005. ISBN 3540243712

[Mee82] R. Meersman. The RIDL Conceptual Language. Technical report, International Centre for Information Analysis Services, Control Data Belgium, Inc., Brussels, Belgium, EU, 1982.

[Mer03] Meriam–Webster Online, Collegiate Dictionary, 2003.
`http://www.webster.com`

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice–Hall, Englewood Cliffs, New Jersey, USA, 1989. ASIN 0131672630

[PB99] H.A. (Erik) Proper and P.D. Bruza. What is Information Discovery About? *Journal of the American Society for Information Science*, 50(9):737–750, July 1999.

[PBH04] H.A. (Erik) Proper, A.I. Bleeker, and S.J.B.A. Hoppenbrouwers. Object–Role Modelling as a Domain Modelling Approach. In J. Grundspenkis and M. Kirikova, editors, *Proceedings of the Workshop on Evaluating Modeling Methods for Systems Analysis and Design (EMMSAD'04), held in conjunctiun with the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004),*, volume 3, pages 317–328, Riga, Latvia, EU, June 2004. Faculty of Computer Science and Information Technology. ISBN 9984976718

[Pei69a] C.S. Peirce. *Volumes I and II – Principles of Philosophy and Elements of Logic*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969. ISBN 0674138007

[Pei69b] C.S. Peirce. *Volumes III and IV – Exact Logic and The Simplest Mathematics*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969. ISBN 0674138005

[Pei69c] C.S. Peirce. *Volumes V and VI – Pragmatism and Pragmaticism and Scientific Metaphysics*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969. ISBN 0674138023

[Pei69d] C.S. Peirce. *Volumes VII and VIII – Science and Philosophy and Reviews, Correspondence and Bibliography*. Collected Papers of C.S. Peirce. Harvard University Press, Boston, Massachusetts, USA, 1969. ISBN 0674138031

[PH04] H.A. (Erik) Proper and S.J.B.A. Hoppenbrouwers. Concept Evolution in Information System Evolution. In J. Gravis, A. Persson, and J. Stirna, editors, *Forum proceedings of the 16th Conference on Advanced Information Systems 2004 (CAiSE 2004), Riga, Latvia, EU, Riga, Latvia, EU*, pages 63–72, Riga, Latvia, EU, June 2004. Faculty of Computer Science and Information Technology. ISBN 998497670X

[PPY01] M.P. Papazoglou, H.A. (Erik) Proper, and J. Yang. Landscaping the information space of large multi–database networks. *Data & Knowledge Engineering*, 36(3):251–281, 2001.

[Pro94a] H.A. (Erik) Proper. *A Theory for Conceptual Modelling of Evolving Application Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1994. ISBN 909006849X

[Pro94b] H.A. (Erik) Proper. ConQuer–92 – The revised report on the conceptual query language LISA–D. Technical report, Asymetrix Research Laboratory, University of Queensland, Brisbane, Queensland, Australia, 1994.

[Pro97] H.A. (Erik) Proper. Data Schema Design as a Schema Evolution Process. *Data & Knowledge Engineering*, 22(2):159–189, 1997.

[Pro98] H.A. (Erik) Proper. Da Vinci – Architecture–Driven Business Solutions. Technical report, Origin, Utrecht, The Netherlands, EU, Summer 1998.

[Pro01] H.A. (Erik) Proper, editor. *ISP for Large–scale Migrations*. Information Services Procurement Library. ten Hagen & Stam, Den Haag, The Netherlands, EU, 2001. ISBN 9076304882

[Pro04] H.A. (Erik) Proper. *Architecture–driven Information Systems Engineering*. DaVinci Series. Nijmegen Institute for Information and Computing Sciences, University of Nijmegen, Nijmegen, The Netherlands, EU, 2004.

[PW94] H.A. (Erik) Proper and Th.P. van der Weide. EVORM – A Conceptual Modelling Technique for Evolving Application Domains. *Data & Knowledge Engineering*, 12:313–359, 1994.

[PW95] H.A. (Erik) Proper and Th.P. van der Weide. A General Theory for the Evolution of Application Models. *IEEE Transactions on Knowledge and Data Engineering*, 7(6):984–996, December 1995.

[RMD99] V.E. van Reijswoud, J.B.F Mulder, and J.L.G. Dietz. Commucation Action Based Business Process and Information Modelling with DEMO. *The Information Systems Journal*, 9(2):117–138, 1999.

[Rop99] G. Ropohl. Philosophy of Socio–Technical Systems. *In Society for Philosophy and Technology*, 4(3), 1999.

[SFG+00] J.J. Sarbo, J.I. Farkas, F.A. Grootjen, P. van Bommel, and Th.P. van der Weide. Meaning Extraction from a Peircean Perspective. *International Journal of Computing Anticipatory Systems*, 6:209–227, 2000.

[Sim62] H.A. Simon. The architecture of complexity. In *Proceedings of the American Philosophical Society*, volume 106, pages 467–482, 1962.

[Spi88] J.M. Spivey. *Understanding Z: A Specification Language and its Formal Semantics*. Cambridge University Press, Cambridge, United Kingdom, EU, 1988.

[Sto77] J.E. Stoy. *Denotational Semantics: The Scott–Strachey Approach to Programming Language Semantics*. MIT Press, Cambridge, Massachusetts, USA, 1977.

[SWS89] P.S. Seligmann, G.M. Wijers, and H.G. Sol. Analyzing the Structure of I.S. Methodologies, an alternative approach. In R. Maes, editor, *Proceedings of the First Dutch Conference on Information Systems*, 1989.

[TP91] T.H. Tse and L. Pong. An Examination of Requirements Specification Languages. *The Computer Journal*, 34(2):143–152, April 1991.

[Vel92] J. in 't Veld. *Analyse van organisatieproblemen – Een toepassing van denken in systemen en processen*. Stenfert Kroese, Leiden, The Netherlands, EU, 1992. In Dutch. ISBN 9020722816

[Ver93] T.F. Verhoef. *Effective Information Modelling Support*. PhD thesis, Delft University of Technology, Delft, The Netherlands, EU, 1993. ISBN 9090061762

[VHP03] G.E. Veldhuijzen van Zanten, S.J.B.A. Hoppenbrouwers, and H.A. (Erik) Proper. System Development as a Rational Communicative Process. In N. Callaos, D. Farsi, M. Eshagian–Wilner, T. Hanratty, and N. Rish, editors, *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, volume XVI, pages 126–130, July 2003. ISBN 9806560019

[VHP04] G.E. Veldhuijzen van Zanten, S.J.B.A. Hoppenbrouwers, and H.A. (Erik) Proper. System Development as a Rational Communicative Process. *Journal of Systemics, Cybernetics and Informatics*, 2(4), 2004.
`http://www.iiisci.org/Journal/sci/pdfs/P492036.pdf`

[WBW00] B.C.M. Wondergem, P. van Bommel, and Th.P. van der Weide. Matching Index Expressions for Information Retrieval. *Information Retrieval Journal*, 2(4), 2000. To appear.

[WBW01] B.C.M. Wondergem, P. van Bommel, and Th.P. van der Weide. Combining Boolean Logic and Linguistic Structure. *Information & Software Technology*, (43):53–59, 2001.

[WH90] G.M. Wijers and H. Heijes. Automated Support of the Modelling Process: A view based on experiments with expert information engineers. In B. Steinholz, A. Sølvberg, and L. Bergman, editors, *Proceedings of the Second Nordic Conference CAiSE'90 on Advanced Information Systems Engineering, Stockholm, Sweden, EU*, volume 436 of *Lecture Notes in Computer Science*, pages 88–108, Berlin, Germany, EU, 1990. Springer. ISBN 3540526250

[WHB92] Th.P. van der Weide, A.H.M. ter Hofstede, and P. van Bommel. Uniquest: Determining the Semantics of Complex Uniqueness Constraints. *The Computer Journal*, 35(2):148–156, April 1992.

[Win90] J.J.V.R. Wintraecken. *The NIAM Information Analysis Method: Theory and Practice*. Kluwer, Deventer, The Netherlands, EU, 1990.

[WK03] J. Warmer and A. Kleppe. *The Object Constraint Language: Getting Your Models Ready for MDA*. Addison Wesley, Reading, Massachusetts, USA, 2nd edition, 2003. ISBN 0321179366

# List of Symbols

$U \models^{\mathsf{s}}_v S' \subset_a S \triangleq U \models^{\mathsf{s}}_v S' \subset S$ and $(S' \cap \mathcal{LI}) \subset S$ – For **viewer** $v$ viewing **universe** $U$, **system** $S'$ is a **aspect system** of $S$

$U \models^{\mathsf{s}}_v S' \subset_c S \triangleq U \models^{\mathsf{s}}_v S' \subset S$ and $(S' \cap \mathcal{CO}) \subset S$ – For **viewer** $v$ viewing **universe** $U$, **system** $S'$ is a **component system** of $S$

$\mathcal{CE} \triangleq \wp(\mathcal{EE})$ – The set of **conception evolutions**.

$\_ \models^{\mathsf{c}}_{\_} \_ \subseteq \mathcal{UN} \times \mathcal{VW} \times \wp(\mathcal{EL})$ – A relationship expressing which **conception** is held by which **viewer**. The fact that a **viewer** $v$ harbours a **conception** $C$ for **universe** $U$ is expressed as $U \models^{\mathsf{c}}_v C$.

$\mathcal{CO}_X \triangleq X \cap \mathcal{CO}$ – A subset of the set of **concepts**.

$\mathcal{CO} \subseteq \mathcal{EL}$ – The set of **elements** of a **conception** that are **concepts**.

$x \rightrightarrows_C y \triangleq x = y \vee x \rightarrow_C y$ – A derived relationship providing the decomposition of a composed **concept** in some **viewer**'s **conception**. If $x \rightrightarrows_C y$, the **concept** $x$ in **conception** $C$ is decomposed into (possibly amongst others) **concept** $y$, or $x$ and $y$ are equal.

$\_ \rightarrow_{\_} \_ \subseteq \mathcal{CO} \times \wp(\mathcal{EL}) \times \mathcal{CO}$ – A derived relationship providing the decomposition of a composed **concept** in some **viewer**'s **conception**. If $x \rightarrow_C y$, the **concept** $x$ in **conception** $C$ is decomposed into (possibly amongst others) **concept** $y$.

$\mathcal{DC} \subseteq \mathcal{LI}$ – Decomposer **links**.

$\_ \models^{\mathsf{d}}_{\_} \langle \_ : \_ : \_ \rangle \subseteq \mathcal{UN} \times \mathcal{VW} \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL})$ – A relationship expressing which **conception** of a **domain** and **environment** is held by which **viewer** with a particular **interest**. The fact that a **viewer** $v$ harbours a **conception** $C$ of domain $D$ with environment $E$ an for **universe** $U$ is expressed as $U \models^{\mathsf{d}}_v \langle C : D : E \rangle$.

$\mathcal{EE} \triangleq \mathcal{TI} \rightarrowtail \mathcal{EL}$ – The set of **element evolutions**.

$\mathcal{EL}$ – The set of **elements** that may be part of a **conception**.

$\mathsf{From} : \mathcal{LI} \rightarrow \mathcal{CO}$ – The source **concept** of a **link**.

$\mathsf{Involved}(r) \triangleq \{\mathsf{From}(r), \mathsf{To}(r)\}$ – The set comprising the source and destination **elements** of a **link** in a **conception**.

$< \subseteq \mathcal{TI} \times \mathcal{TI}$ – A complete and total order over points in time.

$e \, \mathsf{LinkedTo}_C \, f \triangleq \exists_{l \in \mathcal{LI}_C} [\mathsf{From}(l) = e \wedge \mathsf{To}(l) = f]$ – A (derived) relationship denoting the fact that there exists a **link** from one **concept** to another.

$\mathcal{LI}_X \triangleq X \cap \mathcal{LI}$ – A subset of the set of **links**.

$\mathcal{LI} \subseteq \mathcal{EL}$ – The set of **elements** of a **conception** that are **links** between **concepts**.

$U \models^{\mathsf{m}}_v M \triangleq \exists_{C,E} [U \models^{\mathsf{m}}_v \langle C : E : M \rangle]$ – A relationship expressing which **model** is held by which **viewer**. The fact that a **viewer** $v$ harbours a **model** $M$ of part of **universe** $U$ is expressed as $U \models^{\mathsf{m}}_v M$.

$\_ \models^{\mathsf{m}}_\_ \langle \_ : \_ : \_ \rangle \subseteq \mathcal{UN} \times \mathcal{VW} \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL})$ – A relationship expressing which **model** and **environment** or some part of the **universe** are held by which **viewer**. The fact that a **viewer** $v$ with a **conception** $C$ harbours a **model** $M$ with **environment** $E$ for a part of **universe** $U$ is expressed as $U \models^{\mathsf{m}}_v \langle C : M : E \rangle$.

$t_1 \rhd t_2 \triangleq t_1 < t_2 \wedge \neg \exists_s [t_1 < s < t_2]$ – The next point in time. As $<$ is a complete and total order, there is always a unique next point in time. This allows us to write $\rhd t$.

$U \models^{\mathsf{s}}_v S' \subset S \triangleq U \models^{\mathsf{s}}_v S,\ U \models^{\mathsf{s}}_v S'$ and $S' \subset S$ – For **viewer** $v$ viewing **universe** $U$, **system** $S'$ is a **sub-system** of $S$

$U \models^{\mathsf{s}}_v S \triangleq \exists_{C,E} [U \models^{\mathsf{s}}_v \langle C : E : S \rangle]$ – A relationship expressing which **system** is viewed by which **viewer**. The fact that a **viewer** $v$ views system $S$ in **universe** $U$ is expressed as $U \models^{\mathsf{s}}_v S$.

$\_ \models^{\mathsf{s}}_\_ \langle \_ : \_ : \_ \rangle \subseteq \mathcal{UN} \times \mathcal{VW} \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL}) \times \wp(\mathcal{EL})$ – A relationship expressing which **system** and **environment** are viewed in the **universe** by a **viewer**. The fact that a **viewer** $v$ with **conception** $C$ views **system** $M$ with **environment** $E$ for a part of **universe** $U$ is expressed as $U \models^{\mathsf{s}}_v \langle C : S : E \rangle$.

$\mathcal{TI}$ – Points of time.

$\mathsf{To} : \mathcal{LI} \to \mathcal{CO}$ – The destination **concept** of a **link**.

$\mathcal{UN}$ – The set of **universes**.

$\mathcal{VW}$ – The set of **viewers**.

# Dictionary

**Active system –** A special kind of **system** that is conceived of as begin able to change parts of the **universe**.

**Activity participation –** A **system link** between a **system activity** and one of its **actor**.

**Actor –** A **system element** that is conceived of as having some involvement in a **system activity**. This involvement is a special kind of **system link**, referred to as an **activity participation**.

**Aspect system –** an aspect-system $S'$ of a **system** $S$, is a **sub-system**, where the set of **model links** in $S'$ is a proper subset of the set of the **links** in $S$.

**Autonomous system –** an **open active system** (possibly also a **responsive system**, but not a **reactive system**) where at least one expression is an action. A human being and most (if not all) **organizations** can be regarded as **autonomous systems**.

**Component –** Is an abbreviation for: **component system**.

**Component system –** A component-system $S'$ of a **system** $S$, is a **sub-system**, where the set of **model concepts** in $S'$ is a proper subset of the set of entities in $S$.

**Concept –** Any **element** from a **conception** that is not a **link**.

**Conception –** That what results, in the mind of a **viewer**, when they interpret a **perception** of a **domain**.

**Conception evolution –** The evolution of a **conception**.

**Construction process –** A process aiming to realize and test a **system** that is regarded as a (possibly artificial) artifact that is not yet in operation.

**Data –** Any representation in some language. Data is therefore simply a collection of symbols that may, or may not, have some meaning to some **actor**.

**Decomposer –** The **link** between a composed **concept** and one of its underlying **concepts**.

**Definition –** The **requirements** that should be met by a desired **work system** as well its **system description** including the descriptions of the system's definition, design as well as documentation for the operational system.

These requirements will typically identify: *what* it should do, *how well* it should do this, and *why* it should do so.

**Definition description –** The **description** of a **definition**.

**Deployment –** Is an abbreviation for: **deployment**.

**Deployment process –** A process aiming to make a **system** operational, i.e. to implement the use of the system by its prospective users.

**Description –** The result of a **viewer** denoting a **conception**, using some language to express themselves.

**Design –** The identification and motivation of *how* a **work system** will meet the requirements set out in its definition. The resulting design may (depending on the design goals) range from high-level designs to the detailed level of programming statements or specific worker tasks.

**Design description –** The **description** of a **design**.

**Domain –** Any 'part' or 'aspect' of the **universe** a **viewer** may have an **interest** in.

**Domain evolution –** The evolution of a **domain** over time.

**Dynamic system –** A special kind of **system** that is conceived of as undergoing change in the cause of time.

**Element –** The elementary parts of a **viewer**'s **conception**.

**Element evolution –** The evolution over time of an **element** in the **conception** of a **viewer**.

**Element version –** The version of an **element evolution** as it holds at some point in time. This version is an **element** from a **viewer**'s **conception** of a **universe**.

**Environment –** The environment of a **domain** is that part of a **viewer**'s **conception** of a **universe**, which has a direct **link** to the **domain**.

**Environment evolution –** The evolution of an **environment** over time.

**Human actors –** An **actor** which is a single human being, or essentially a set of human-beings, such as a team.

**Information –** The **knowledge** increment brought about when a **human actor** receives a message. In other words, it is the difference between the **conceptions** held by a **human actor** *after* interpreting a received **message** and the **conceptions** held beforehand.

**Information system –** A **sub-system** of an **organizational system**, comprising the conception of how the communication and information-oriented aspects of an **organization** are composed and how these operate, thus leading to a description of the (explicit and/or implicit) communication-oriented and information-providing actions and arrangements existing within the **organizational system**.

**Interest –** The specific reason(s) why a **viewer** observes a **domain**.

In the case of a **system**, this this is usually a confluence of the **systemic properties** of **interest** to the **system viewer** and the aspects of the **system** that are considered relevant (by the **system viewer** to these **systemic properties**).

**Knowledge –** A relatively stable, and usually mostly consistent, set of **conceptions** posessed by a single (possibly composed) **actor**.

In more popular terms: "an actor's picture of the world".

**Link –** Any **element** from a **conception** that relates two **concepts**.

**Message –** **Data** that is transmitted from one **actor** (the sender) to another **actor** (the receiver).

A message may actually be 'routed' via several **actors** before reaching its actual receiver. For example, when **human actor** exchange messages, they usually need to make use of some other **actor** playing the role of a medium (for example, vibrations in the air, or an e-mail system).

**Model –** A purposely abstracted **domain** (possibly in conjunction with its **environment**) of some 'part' or 'aspect' of the **universe** a **viewer** may have an **interest** in.

For practical reasons, a **model** will typically be consistent and unambiguous with regards to some underlying semantical domain, such as logic.

**Model concept –** A **concept** from a **conception** which is a **model**.

**Modeling –** The act of purposely abstracting a **model** from (what is conceived to be) a part of the **universe**.

**Model link –** A **link** from a **conception** which is a **model**.

**Open active system –** A **system** that is an **open system** as well as an **active system**.

**Open system –** A special kind of **dynamic system** that is conceived as reacting to external triggers, i.e. there may be changes inside the system due to external causes originating from the system's **environment**.

**Organization –** A group of **actors** with a purpose, who:

- interact with each other,
- form a network of roles,
- make use of (the services of) other actors.

An **organization** in itself is an **actor** as well, and may as such participate in yet another **organizations**.

**Organizational system –** A special kind of **work system**, being normally active and open, and comprising the conception of how an **organization** is composed and how it operates (i.e. performing specific actions in pursuit of organizational goals, guided by organizational rules and informed by internal and external communication), where its **systemic property** are that it responds to (certain kinds of) changes caused by the system **environment** and, itself, causes (certain kinds of) changes in the system environment.

**Perception –** That what results, in the mind of a **viewer**, when they observe a **domain** with their senses, and forms a specific pattern of visual, auditory or other sensations in their minds.

**Quality –** Is the totality of **systemic properties** of a **system** that relate to its ability to satisfy stated and/or implied needs.

**Quality property –** A **systemic property**, used to describe and asses the **quality** of a **system**.

**Reactive system –** An **open active system** where each expression of the system is a reaction, and where each impression immediately causes a reaction.

**Requirement –** an essential **quality property** that a **system** or its **system description** has to satisfy.

**Responsive system –** An **open active system** (possibly also a **reactive system**) where it holds for at least one expression that a certain impression or a temporal pattern of impressions is a necessary, but not a sufficient dynamic condition for its occurrence. The receipt of an order is a necessary impression to a "sales system", for the expression "delivery of the ordered goods", but it is not a sufficient condition.

**Sub-system –** A sub-system $S'$ of a **system** $S$, is a **system** where the set of **elements** in $S'$ is a subset of the **elements** in $S$.

**System –** A special **model** of a **system domain**, whereby all the things contained in that model are transitively coherent, i.e. all of them are directly or indirectly related to each other and form a coherent whole.

A system is conceived as having assigned to it, as a whole, a specific characterisation (a non-empty set of **systemic properties**) which, in general, cannot be attributed exclusively to any of its components.

**System activity –** A **system concept** that is conceived of as changing parts of the **universe**.

**System concept –** Any element from a **system** that is a **concept**.

**System description –** The **description** of a **system**.

**System domain –** A **domain** that is conceived to be a **system**, by some **viewer**, by the distinction from its **environment**, by its coherence, and because of its **systemic property**.

**System element –** Any element from a **system**.

**System exposition –** a **description** of all the **elements** of the **system domain** where each **element** is specified by all its relevant aspects and all the roles it plays, being of importance for the **interest** of the **viewer**. (The **system viewer** may conceive one and the same thing in the **system domain** to play more than one role in the **system**.)

**Systemic property –** A meaningful relationship that exists between the **domain** of elements considered as a whole, the **system domain** and its **environment**.

**System link –** Any element from a **system** that is a **link**.

**System type –** A type that determines the potential kinds of **systemic properties**, **elements** of the **system domain** and roles of the **elements** in achieving the **systemic properties**.

**System viewer –** A **viewer** of a **system domain**.

**Universe –** The 'world' under consideration.

**Viewer –** An **actor** perceiving and conceiving (part of) a **domain**.

**Work system –** An **open active system** in which **actors** perform processes using **information**, technologies, and other resources to produce products and/or services for internal or external actors.

# Author Index


**A**

Abiteboul, S., 89

Ackoff, R.L., 41

Alter, S., 19, 41

Avison, D.E., 20

**B**

Batini, C., 88

Bemelmans, T.M.A., 38

Berger, F.C., 11

Bertalanffy, L. von, 23, 28

Bleeker, A.I., 95, 97

Bloesch, A.C., 63

Bommel, P. van, 11, 69

Bonsangue, M., 11

Booch, G., 49, 96, 100

Bronts, G.H.W.M., 11, 77, 88

Brouwer, S.J., 11, 77, 88

Bruza, P.D., 11

Bubenko, J.A., 20

Buuren, R. van, 11

**C**

Campbell, L.J., 11

Ceri, S., 88

Checkland, P., 25

Chen, P.P., 96

Cohen, B., 20

Creasy, P.N., 11, 77, 80

**D**

Dietz, J.L.G., 49

**E**

Ehrich, H.-D., 88

Elmasri, R., 88

Embley, D.W., 49, 96, 100

Engels, G., 88

**F**

Falkenberg, E.D., 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

Farkas, J.I., 11

Frederiks, P.J.M., 11, 96, 100

**G**

Gogolla, M., 88

Grootjen, F.A., 11

**H**

Hülsmann, K., 88

Halpin, T.A., 11, 30, 49, 57, 63, 77, 79, 86, 95, 96, 98, 99

Heijes, H., 96

Hesse, W., 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

Hevner, A., 88

Hofstede, A.H.M. ter, 11, 20, 57, 63, 69, 77, 79, 86, 89

Hohenstein, U., 88

Hoppenbrouwers, J.J.A.C., 11

Hoppenbrouwers, S.J.B.A., 11, 95–98, 100

Horenbeek, I. van, 20

Hull, R., 89

**I**

Iivari, J., 23

**J**

Jacobson, I., 49, 96, 100

Jones, C.B., 20

Jonkers, H., 11

**K**

Kleppe, A., 63

Kristen, G., 49, 96, 100

Kurtz, B.D., 49, 96, 100

**L**

Löhr-Richter, P., 88

Langefors, B., 23

Lankhorst, M.M, 11

Lankhorst, M.M., 101

Levy, A.Y., 89

Lewi, J., 20

Lindgreen, P., 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

**M**

Martens, C.L.J., 11, 77, 88

Meersman, R., 63

Mulder, J.B.F, 49

**N**

Navathe, S.B., 88

Nijssen, G.M., 49, 95, 96, 98

Nilsson, B.E., 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

**O**

Oei, J.L.H., 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

**P**

Papazoglou, M.P., 11

Peirce, C.S., 26

Pong, L., 20

Proper, H.A. (Erik), 10, 11, 20, 30, 42, 57, 63, 77, 80, 88, 95–98, 100

**R**

Reijswoud, V.E. van, 49

Rolland, C., 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

Ropohl, G., 24, 36

Rumbaugh, J., 49, 96, 100

**S**

Saake, G., 88

Sarbo, J.J., 11

Seligmann, P.S., 96

Simon, H.A., 37

Sol, H.G., 96

Spivey, J.M., 20, 63

Stamper, R.K. and, 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

Stoy, J.E., 66

**T**

Torre, L. van der, 11

Tse, T.H., 20

**V**

Veld, J. in 't, 38

Veldhuijzen van Zanten, G.E., 11, 96, 100

Verhoef, T.F., 95, 99

Verrijn–Stuart, A.A., 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

Vos, B. van der, 11

Voss, K., 9, 18, 23, 24, 26, 31, 35, 36, 38, 39, 41

**W**

Warmer, J., 63

Weeldreyer, J., 88

Weide, Th.P. van der, 11, 20, 57, 63, 69, 77, 79, 86, 89, 96, 100

Wijers, G.M., 96

Wintraecken, J.J.V.R., 49

Wondergem, B.C.M., 11

Woodfield, S.N., 49, 96, 100

**Y**

Yang, J., 11

# Subject Index

The following conventions are used in this index:

- A page where a concept is defined: *141*.
- A page where a concept is discussed or mentioned: 141.
- The page in the dictionary where a concept is defined: **141**.


**A**

active system, *40*, 40, 42, 101, **135**, 136

activity participation, **135**, 135

actor, 9, 18, 19, 26, 37, 38, 41, **135**, 135–138

architecting, *10*

aspect system, *38*, 38, 39, 133, **135**

autonomous system, *41*, 41, **135**, 135

**C**

communication, *9*

component, 23, 25, 37, 38, **135**

component system, *38*, 38, 39, 133, **135**, 135

computerized information system, *41*

concept, *29*, 29–32, 35, 36, 39, 101, 133, 134, **135**, 135–137

conception, 9, *26*, 26, 28–32, 35, 36, 40, 42–44, 49, 50, 133, 134, **135**, 135, 136

conception evolution, 44, 133, **135**

construction process, 10, **135**

**D**

data, *9*, **135**, 136

decomposer, 32, 34, **135**

definition, 9, 10, **135**, 135

definition description, 9, **135**

definition process, *9*

deployment, 10, **135**, 135

deployment process, 10, **135**

description, *26*, 26, 35, 36, 39, 49, **135**, 135–137

design, 10, **135**, 136

design description, 10, **136**

design process, *10*

domain, 20, 24, 26, *28*, 28–32, 34–36, 38, 39, 44, 133, 135, **136**, 136–138

domain evolution, 44, **136**

domain modeling, *10*

dynamic system, *40*, 40, **136**, 137

**E**

element, 24–26, 28, *29*, 29, 31, 34–36, 38, 39, 42, 43, 133, 135, **136**, 136–138

element evolution, 43, 44, 50, 133, **136**, 136

element version, 43, **136**

environment, 25, 28–30, *31*, 31, 32, 34–36, 40, 41, 44, 133, 134, **136**, 136–138

environment evolution, 44, **136**

**H**

human actor, 9, **136**, 136

**I**

information, *9*, 19, 41, **136**, 138

information system, 18, 23, 24, 37, 40, 41, *42*, **136**

interest, 26, 28, 31, 35, 39, 133, **136**, 136, 137

**K**

knowledge, *9*, 9, **136**, 136

**L**

link, 23–25, 28, *29*, 29–32, 35, 36, 38, 39, 133–135, **136**, 136, 138

**M**

message, 9, **136**, 136

model, *35*, 35, 36, 134, **136**, 136, 137

model concept, *35*, 38, 135, **136**

model element, *35*

model link, *35*, 38, 135, **136**

modeling, 17, *35*, 35, 39, **136**

**O**

open active system, 19, 40, 41, 135, **136**, 137, 138

open system, *40*, 42, 101, 136, **137**

organization, 17, *18*, 18, 20, 23–26, 35, 37–42, 135, 136, **137**, 137

organizational system, 24, 37, *41*, 42, 101, 136, **137**

**P**

perception, *26*, 26, 135, **137**

**Q**

quality, **137**, 137

quality property, **137**, 137

**R**

reactive system, *41*, 41, 135, **137**, 137

realization process, *10*

requirement, 10, 135, **137**

responsive system, *41*, 41, 135, **137**

**S**

sub-system, 25, 37, *38*, 38–42, 134–136, **137**

system, 10, 17, 23–26, 28, 29, 35, *36*, 36–42, 50, 133–136, **137**, 137, 138

system activity, 135, **137**

system concept, *36*, **137**, 137

system description, 10, *36*, 135, **137**, 137

system domain, *35*, 35–37, 39, **137**, 137, 138

system element, *36*, 135, **137**

system exposition, *39*, 39, **137**

system link, *36*, 135, **138**

system type, *39*, 39, **138**

system viewer, *35*, 39, 136, 137, **138**

systemic property, 25, 26, *35*, 35–39, 41, 136, 137, **138**, 138

**U**

universe, *26*, 26, 28, 31, 32, 35, 36, 40, 42–44, 133–137, **138**

**V**

viewer, 25, *26*, 26, 28–32, 35–37, 39, 42–44, 49, 133–137, **138**, 138

**W**

work system, 10, 17, *19*, 19, 20, 23, 24, 40, *41*, 41, 135, 137, **138**

## The DAVINCI Lecture Notes Series:

The DAVINCI series of lecture notes is concerned with *The Art & Craft of Information Systems Engineering*. On the one hand, this series of lecture notes takes a fundamental view (*craft*) on the field information systems engineering. At the same time, it does so with an open eye to practical experiences (the *art*) gained from information system engineering in industry.

## Main contributors:

P. (Patrick) van Bommel

S.J.B.A. (Stijn) Hoppenbrouwers

G.F.M. (Ger) Paulussen

H.A. (Erik) Proper

Th.P. (Theo) van der Weide