

MEMO Organisation Modelling Language (1): focus on organisational structure

Frank, Ulrich

In: ICB Research Reports - Forschungsberichte des ICB / 2011

This text is provided by DuEPublico, the central repository of the University Duisburg-Essen.

This version of the e-publication may differ from a potential published print or online version.

DOI: <https://doi.org/10.17185/duepublico/47066>

URN: <urn:nbn:de:hbz:464-20180918-070152-6>

Link: <https://duepublico.uni-duisburg-essen.de/servlets/DocumentServlet?id=47066>

License:

As long as not stated otherwise within the content, all rights are reserved by the authors / publishers of the work. Usage only with permission, except applicable rules of german copyright law.

Source: ICB-Research Report No. 48, December 2011

Ulrich Frank



MEMO Organisation Modelling Language (1): Focus on Organisational Structure

ICB-RESEARCH REPORT

Die Forschungsberichte des Instituts für Informatik und Wirtschaftsinformatik dienen der Darstellung vorläufiger Ergebnisse, die i. d. R. noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar.

The ICB Research Reports comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

Alle Rechte vorbehalten. Insbesondere die der Übersetzung, des Nachdruckes, des Vortrags, der Entnahme von Abbildungen und Tabellen – auch bei nur auszugsweiser Verwertung.

All rights reserved. No part of this report may be reproduced by any means, or translated.

Authors' Address:

Ulrich Frank

Lehrstuhl für Wirtschaftsinformatik
und Unternehmensmodellierung

Institut für Informatik und
Wirtschaftsinformatik (ICB)

Universität Duisburg-Essen

Universitätsstr. 9

D-45141 Essen

ulrich.frank@uni-due.de

ICB Research Reports

Edited by:

Prof. Dr. Heimo Adelsberger

Prof. Dr. Peter Chamoni

Prof. Dr. Frank Dorloff

Prof. Dr. Klaus Echtele

Prof. Dr. Stefan Eicker

Prof. Dr. Ulrich Frank

Prof. Dr. Michael Goedicke

Prof. Dr. Volker Gruhn

PD Dr. Christina Klüver

Prof. Dr. Tobias Kollmann

Prof. Dr. Bruno Müller-Clostermann

Prof. Dr. Klaus Pohl

Prof. Dr. Erwin P. Rathgeb

Prof. Dr. Enrico Rukzio

Prof. Dr. Albrecht Schmidt

Prof. Dr. Rainer Unland

Prof. Dr. Stephan Zelewski

Contact:

Institut für Informatik und
Wirtschaftsinformatik (ICB)

Universität Duisburg-Essen

Universitätsstr. 9

45141 Essen

Tel.: 0201-183-4041

Fax: 0201-183-4011

Email: icb@uni-duisburg-essen.de

ISSN 1860-2770 (Print)

ISSN 1866-5101 (Online)

Abstract

Organisation models are at the core of enterprise model, since they represent key aspects of a company's action system. Within MEMO, the Organisation Modelling Language (OrgML) supports the construction of organisation models. They can be divided into two main abstractions: a static abstraction is focusing on the structure of an organisation that reflects the division of labour with respect to static responsibilities and a dynamic abstraction that is focusing on models of business processes. In this report, those concepts of the OrgML are presented that serve modelling static aspects of an organisation. They cover prevalent concepts used in organisational charts as well as concepts required for specifying temporal units of work, roles and committees. The description of these modelling concepts comprises three parts: A meta model serves to specify the abstract syntax and semantics. A language description includes an overview of all concepts and the corresponding notational elements (concrete syntax). Finally, the use of the concepts is illustrated through various examples.

Table of Contents

FIGURES	III
TABLES	IV
TYPOGRAPHICAL CONVENTIONS	V
1 INTRODUCTION	1
2 ORGANISATIONAL STRUCTURE: A TRIVIAL SUBJECT?	3
2.1 GLOSSARY OF TECHNICAL TERMS	4
2.2 SEMANTIC PECULIARITIES.....	8
2.3 SCOPE AND PURPOSE	11
2.4 SPECIFIC REQUIREMENTS	12
3 ANALYSIS OF POTENTIAL LANGUAGE CONCEPTS	14
3.1 ANALYSIS OF KEY TERMS.....	14
3.2 LEVELS OF ABSTRACTION	16
4 LANGUAGE SPECIFICATION	21
4.1 BASIC DESIGN DECISIONS	21
4.2 LANGUAGE CONCEPTS	25
4.3 META MODEL.....	48
4.4 EXCURSUS: SUPPORT FOR MANAGING MULTI-LINGUAL MODEL SYSTEMS	54
4.5 GRAPHICAL NOTATION	55
4.6 EXAMPLE DIAGRAMS	64
5 CONCLUSIONS	73
REFERENCES	74

Figures

FIGURE 1: FOCUS ON LINE OF COMMAND	3
FIGURE 2: FOCUS ON COMPOSITION	4
FIGURE 3: SEMANTIC NET OF CORE TECHNICAL TERMS.....	9
FIGURE 4: REFINEMENT TO SPECIFIC POSITIONS	10
FIGURE 5: REFINEMENT TO SPECIFIC ORGANISATIONAL UNITS	10
FIGURE 6: INSTANTIATION OF ORGANISATIONAL UNIT AND POSITION	16
FIGURE 7: MODEL OF ORGANISATION STRUCTURE ON TYPE LEVEL AND EXCERPT OF CORRESPONDING INSTANCES.....	17
FIGURE 8: EXTENDING THE EXAMPLE BY POSITIONS.....	18
FIGURE 9: ILLUSTRATION OF CORE CONCEPTS	24
FIGURE 10: REPRESENTATION OF SELECTED FEATURES	25
FIGURE 11: EXAMPLE OF CATEGORIES.....	44
FIGURE 12: META MODEL OF CONCEPTS TO MODEL ORGANISATION STRUCTURES	50
FIGURE 13: CONSTRAINTS	52
FIGURE 14: REPRESENTATION OF CATEGORIES.....	62
FIGURE 15: COMMENTS AND CONSTRAINTS.....	63
FIGURE 16: CONNECTORS.....	64
FIGURE 17: FUNCTIONAL ORGANISATION WITHOUT DETAIL.....	65
FIGURE 18: EXAMPLE OF FUNCTIONAL ORGANISATION STRUCTURE DIAGRAM.....	65
FIGURE 19: REPRESENTATION OF MATRIX ORGANISATION	66
FIGURE 20: ORGANISATIONAL CHART WITH ADDITIONAL INFORMATION	67
FIGURE 21: FOCUS ON MANAGEMENT STRUCTURE	68
FIGURE 22: ASSIGNING POSITIONS TO ORGANISATIONAL UNITS.....	69
FIGURE 23: REPRESENTATION OF COMMITTEES AND ROLES.....	70
FIGURE 24: DEFINING AND USING LOCAL TYPES.....	70
FIGURE 25: EXAMPLE OF INTERACTION DIAGRAM.....	71
FIGURE 26: INTERACTION VIA DIFFERENT MEDIA.....	71

Tables

TABLE 1: GLOSSARY OF KEY TERMS	8
TABLE 2: ASSESSMENT OF KEY TERMS	15
TABLE 3: ASSESSMENT OF KEY TERMS	20
TABLE 4: DESCRIPTION OF THE META TYPE ORGANISATION.....	28
TABLE 5: DESCRIPTION OF THE META TYPE ORGANISATIONALUNIT	32
TABLE 6: DESCRIPTION OF THE META TYPE POSITION	34
TABLE 7: DESCRIPTION OF THE META TYPE POSITIONSHARE.....	36
TABLE 8: DESCRIPTION OF THE META TYPE ROLE.....	39
TABLE 9: DESCRIPTION OF THE META TYPE COMMITTEE	41
TABLE 10: DESCRIPTION OF THE META TYPE BOARD	42
TABLE 11: DESCRIPTION OF THE AUXILIARY META TYPE PROTOTYPICALPOSITION	43
TABLE 12: DESCRIPTION OF THE TYPE POSITIONCATEGORY	45
TABLE 13: DESCRIPTION OF THE META TYPE ORGUNITCATEGORY	46
TABLE 14: DESCRIPTION OF THE TYPE ORGANISATIONCATEGORY	46
TABLE 15: DESCRIPTION OF THE META TYPE ORGANISATIONCATEGORY	47
TABLE 16: AUXILIARY TYPES	54
TABLE 17: SYMBOLS FOR REPRESENTING ORGANISATIONAL UNITS	58
TABLE 18: REPRESENTATION OF RELATIONSHIPS.....	60
TABLE 19: REPRESENTATION OF INTERACTION RELATIONSHIPS	62

Typographical Conventions

If textual elements of meta models (or the meta meta model respectively) are referred to in the standard body text, they are printed in Arial, e.g. OrganisationalUnit.

1 Introduction

Division of labour and coordination are prerequisites of organisational action. Usually, division of labour is not redefined every other day, but is based on an organisational structure that is widely stable for a longer period. Organisational processes – which we will usually refer to as business processes – define division of labour on a finer level of granularity for a certain action context. Graphical visualisations of organisational work have been used for long. Against the background of this report – enterprise modelling – it is remarkable that already Taylor, a pioneer of the so called “scientific management” approach suggested large maps of the factory yard (Taylor 1911). They served as a medium to talk about the current organisation and to discuss ways to improve it: "Once the yard was mapped so that one could see at a glance the relationships in time and sequence between different jobs, it led, naturally enough, to the reorganisation of the yard itself ..." (Ward 1964, p. 65). While organisational structures have a clear impact on coordination, they define only certain static aspects of coordination – especially control aspects that are related to certain (management) positions. Business process descriptions on the other hand address provide a more or less rigid scheme for dynamically coordinating functions and actions. Both, the static aspects represented in an organisational structure, and dynamic aspects represented in business process models, are interdependent. It is not conceivable to design an organisational structure without accounting for functions and processes. At the same time, designing a business process is not possible without considering the responsibilities defined in the organisational structure. Hence, the differentiation into organisational structure and dynamics serves mainly analytical purposes: It reduces complexity and thereby fosters a clearer and more elaborate representation of particular views.

The design of an organisational structure happens on different levels of abstraction. On a high level of abstraction, organisational charts are often used to provide a graphical overview of an organisational structure. On a finer level of granularity, mission statements, job descriptions and labour contracts serve to define further details. In addition to these formal aspects, understanding and managing organisations recommend accounting for informal aspects, too, e.g. for values, norms, sources of informal power, organisation culture etc. The focus of the OrgML is on formal, i.e. explicitly defined aspects, of organisational structure. On the one hand, this is an inherent limitation: The OrgML is a (semi-) formal language and therefore corresponding models are widely restricted to formal aspects because informal aspects – by their nature – resist formalisation. On the other hand, focussing on formal aspects makes sense for various reasons. Enterprise modelling is aimed a co-designing action systems and corresponding information systems. That requires representing parts of the action system in the information system – which can be accomplished only for those concepts of an organisation that allow for formalisation. Furthermore, focussing on formal, i.e. precise

Introduction

concepts of organisational design stresses an analytical, we could also say: rational, perspective on design that emphasises the need for explicit goals and the justification of particular design decision. A formal model of organisational structure should also serve as a common foundation for discursive decisions. Notice, however, that emphasising the need for (semi-) formal descriptions of organisations does not mean to completely exclude other aspects. We recommend regarding organisation models as instruments, not as an expression of an epistemological preference.

This report is aimed at presenting the first part of a major revision of the OrgML. The revision was motivated by various factors. The long-time use of the language has produced new requirements. In addition to that our experience with developing domain-specific modelling languages resulted in a revision of the meta modelling language (Frank 2011a) which enables the specification of more elaborate language concepts. Finally, we decided that a more professional graphical notation would improve the language's usability. The (re-) design of the language is aimed at satisfying requirements presented in a previous report (Frank 2011b). References to specific requirements relate to that report. It follows a method for designing DSML (Frank 2011c). Since the targeted DSML is intended to reconstruct existing technical languages for describing organisational structures, we start with describing and analysing a glossary of respective terms. Subsequently, the focus of the investigation will shift to existing graphical representations that should be replaced by diagrams constructed with the OrgML. These representations comprise organisational charts and organisation interaction diagrams. Against this background, the peculiarities of concepts to describe organisational structures and their graphical representation will be considered in more detail. Then, the corresponding meta model will be presented together with the graphical notation. Finally the use of the MEMO OrgML will be illustrated by exemplary diagrams.

This report is not intended to serve as a mere handbook for guiding the use of the OrgML. Instead, it documents the design of the language including the discussion of related challenges and design problems. Therefore, a preliminary graphical notation is used before the language itself is specified. It should not be mistaken for the final notation. It serves to illustrate concepts and related design issues as a foundation for the subsequent language specification. Those readers who are interested in using the language only may want to skip chapters 2 and 3 and focus on the remaining chapters.

2 Organisational Structure: A Trivial Subject?

There is a plethora of textbooks on management in general, on organising the corporation in particular that provide a technical terminology needed to describe organisational structures. Usually, the technical terminology is supplemented with generic patterns of organisational structures and corresponding organisational charts, graphical diagrams that present those aspects of an organisational structure that are regarded as essential for a certain purpose. Figure 1 and Figure 2 show typical organisational charts.

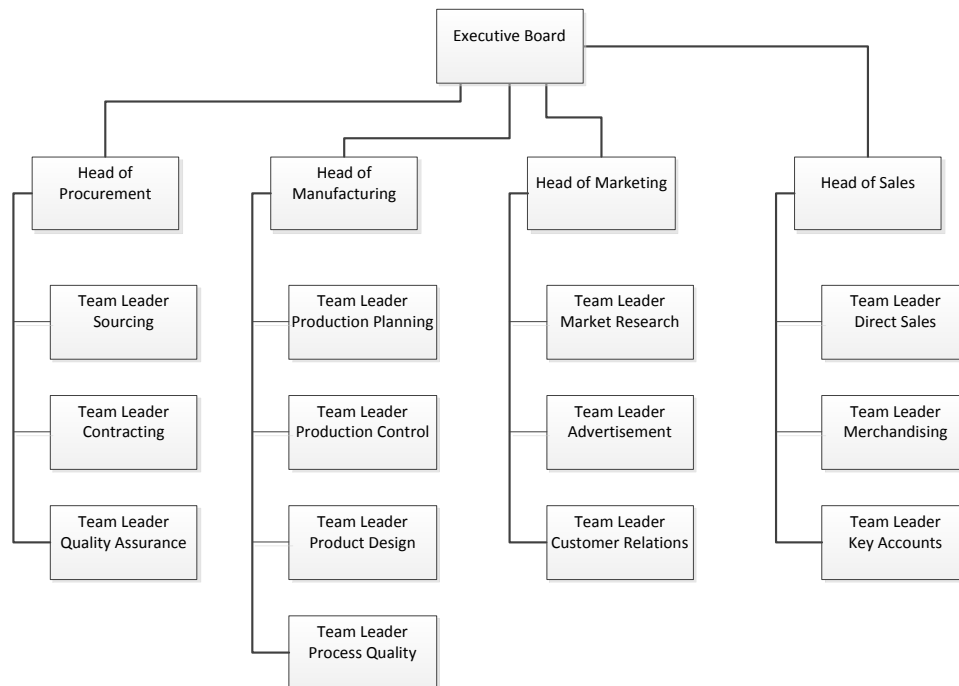


Figure 1: Focus on Line of Command

These two examples indicate that organisational charts often lack a clear definition of the represented concepts: While both examples share the same graphical representation they have apparently different meanings. Nevertheless, from a conceptual modelling point of view, models of organisational structures appear as fairly simple abstractions. However, appearances are deceptive. As we shall see, the design of a DSML for organisation modelling needs to account for the following problems:

Semantic peculiarities: Despite the supposed simplicity of technical terms, it is sometimes challenging to develop a satisfactory definition. This includes the fact that terms cannot always be expected to be used in a uniform way across all organisations.

Different levels of abstraction: Technical terms are often overloaded in the sense that they may represent different levels of abstraction such as type or instance.

Organisational Structure: A Trivial Subject?

Specific requirements for graphical representations: Developing a graphical notation – and guidelines how to apply it – faces various conflicts, e.g. between semantic precision and clarity of presentation.

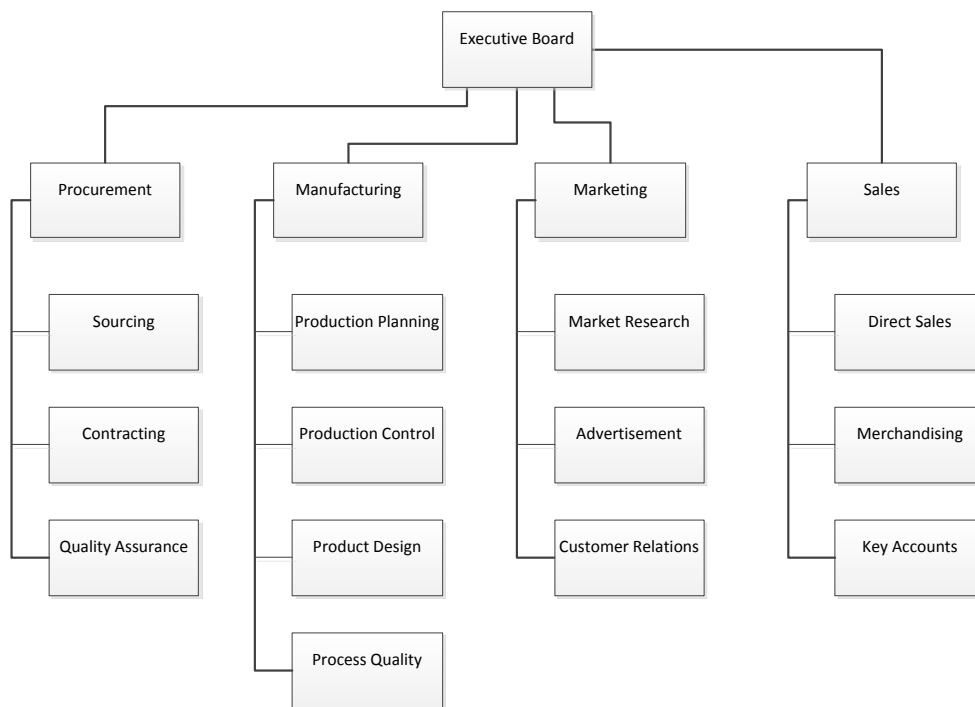


Figure 2: Focus on Composition

To illustrate specific challenges related to modelling organisational structures, we will first look at the corresponding technical terminology.

2.1 Glossary of Technical Terms

Describing organisational structures as they are visualised by the examples in Figure 1 and Figure 2 does not require a complex terminology. A few core terms, such as “organisational unit”, “position” and “is part of” or “reports to” to characterise relationships would be sufficient. A closer look, however, shows that designing organisational structures requires a more elaborate terminology. The following glossary gives an overview of terms as they could be expected from a textbook on organisational design. It builds on work presented in a previous report (Frank 2001a). For each term, it is indicated whether it is directly suited for *modelling* organisational structures (“m”), or rather serves as an *interface* concept to other models (“i”) or addresses *additional* aspects that should be accounted for (“a”), but are not necessarily required as language concepts. Terms as “action”, “activity”, “operation” or “task” are distinguished in a unified way. The terminology suggested in the glossary is inspired by the distinction of “operation”, “action” and “activity” proposed in Leontief 1978).

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

Term	Description	
Action	An action is characterised by an actor who performs it, an object it is related to and an intention that drives the actor. The object can be differentiated into instrument/tool and material. An action may make use of an operation either performed by the corresponding actor himself, by another actor or by a machine.	a
Activity	An activity is composed of one or more actions and/or operations . It serves to fulfil one or more tasks .	m
Actor	A human with respect to his actions .	a
Board	A board is an organisational unit that is composed of superior positions. While each position within a board may be directly superior to other positions , the board itself is superior to other positions .	m
Business Process	A business process is a purposeful organisational construction, which is directed towards the creation of products and/or services for internal or external customers. Executing a business process requires scarce resources . A business process is composed of subprocesses. A subprocess represents a cohesive unit of work, which is triggered by an event and results in one or more further events. The temporal order of subprocesses is subject of a more or less rigid specification, the control structure. Note that “process” is used as a generic term which covers both business processes and subprocesses.	i
Capacity	assigned to an organisational unit , a business process or an actor – describes the amount of output units that can be produced within a given period of time.	a
Centralisation	centralisation is characterised by two interrelated aspects. First, decision making is restricted to a few centres of control. Second, to impose the first rule, the organisation is clearly divided into hierarchical levels of command. Hence, centralisation is characterised by hierarchies whereas decentralisation corresponds to flat organisational structures .	a
Collaboration	If two or more actors work together to accomplish common (or allegedly common) goals , they collaborate. Collaboration implies a (more or less precise) understanding of objects, goals, rules, etc. Therefore, collaboration requires communication and hence a common language.	a
Committee	A committee consists of group of people. Committees can address a wide range of different purposes. Some may make decisions that are mandatory for other organisational units. Others may serve counselling purposes only. In any case, a committee does not include positions, but only roles.	m
Coordination	Division of labour results in a set of activities . It implies the need to combine these activities so that they produce an intended outcome. This combination is called coordination . Coordination can be more or less rigid. It is rigid if it is based on a precise definition of activities and their logical and temporal order - e.g. by specifying business processes that do not allow for individual decisions. On the other hand, coordination may consist of goals only – stressing individual creativity and responsibility .	a
Core Business Process	A business process that is essential for a company's competitiveness, i.e. it represents a competence that allows for effective differentiation against competitors and its outcome is of high relevance for customers.	m
Decision	A decision is a specific kind of activity that is aimed at selecting or creating further activities. It may be performed by one or more actors. It can also be automated based on executing operations on a set of formal decision rules.	m
Decision Making	From a rationalistic, prescriptive point of view decision making consists of four steps: determining goals , detecting alternative options to reach the goals , evaluating the available options, choosing the optimal option. From a more	a

Organisational Structure: A Trivial Subject?

	realistic, descriptive point of view, it is common to take into account human factors that influence the process of decision making – such as fear of failure, avoiding risk, delaying complex decisions etc.	
Decision Scenario Diagram	A decision scenario is a representation of a certain decision type and the context relevant for decision making . A decision scenario diagram represents decision scenarios and relationships between these and relevant context elements such as organisational units, roles or processes . It also includes a structure to describe corresponding decisions in detail.	m
Division of Labour	Division of labour in a corporation describes how the overall work is divided and assigned to organisational units .	a
Employee	A human actor who works for a company on a regular base. An employee holds a position and may fill one to many roles .	m
Formalisation	Formalisation aims at defining organisational rules precisely so that they determine the actions of related actors . While it is similar to the notion of formalisation in mathematics or computer science (avoidance of ambiguity), it is less restrictive because it does not mean to completely specify the semantics of organisational rules. It is sufficient to specify rules to an extent that the likelihood of misinterpretations is relatively small. Bureaucracy takes formalisation to an extreme.	a
Function Diagram	A function diagram is a matrix that is used to assign tasks and responsibilities to organisational units .	m
Goal	Orientation for actions, activities, processes, tasks – process-oriented or state-oriented – serves to guide and control; if two goals support one another, they are complementary; if they hinder one another, they are in conflict; if reaching one goal excludes reaching the other goal , the two goals are contradictory.	i
Incentive	Defining tasks, activities or business processes recommends thinking about the motivation of the actors (employees, customers, suppliers ...) that are involved. Incentives are motivators for actors that are created or identified for this purpose.	a
Line of Command	The line of command defines the superior positions for positions within an organisational structure , hence the positions that are empowered to give instructions. <i>Single line of command</i> means that a position may have no more than one superior position . <i>Multiple line of command</i> allows for more than one superior positions . In the latter case there is need for criteria that describe which superior position is in charge of which type of command – for instance: product-specific, financial etc.	a
Manager	An employee who holds a superior position	a
Medium	A carrier of information used for communication purposes – like text, graphics, voice, audio, video. It can be synchronous or asynchronous.	m
Motivation	The personal reason a human actor feels to perform a task or activity or - in general - to feel committed to goals .	a
Operation	Part of an action that is not further decomposed and follows a certain routine. Can be subject of automation.	m
Organisation Culture	Organisation culture denotes the phenomenon that quality and efficiency of collaboration in organisations depend on attitudes and values the employees share. It is indicated by common ideas of how to solve problems, common perception/conceptualisation of the enterprise, common perception and evaluation of the relevant environment etc. organisation culture is communicated through common practices, rituals and ceremonies.	a
Organisational Structure	Organisational structure is an abstraction of organisation that combines institutional and instrumental aspects. It consists of organisational units (institutional aspect) and their relationships (line of command, responsibilities ...)	a

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

	which stresses an instrumental view.	
Organisational Unit	An organisational unit is a part of an organisation (institutional) that reflects a permanent principle of division of labour within this organisation . An organisational unit may contain other organisational units . The definition of organisational units can be based on functional aspects (e.g. “Finance”, “Production”, “Marketing” ...), product-oriented (e.g. “Trucks”, “Sport Cars” ...), market-oriented (e.g. “North America”, “Europe”, “Consumer”, “Reseller” ...) or combination of these. Usually there is one position that is in charge of an organisational unit .	m
Organisation, institutional	In its institutional sense, the term organisation denotes a social or socio-technical system – like a business firm, a non-profit organisation, public administration etc. An (institutional) organisation has an (instrumental) organisation that comprises an organisational structure and a set of activities and business processes .	m
Organisational Chart	A graphical visualisation of an organisational structure . Organisational charts exist in various flavours – both in terms of their semantics and the symbols used.	m
Organisation Interaction Diagram	A graphical representation of interactions between organisational units. May focus on factual interactions or on supposed.	m
Position	A position is the smallest organisational unit that does not contain any other organisational unit . Usually, a position is assigned to one employee . There are, however, exceptions of this rule (job sharing).	m
Profit Centre	A profit centre is an organisational unit . It emphasises responsibility , independence and creativity – hence, a high degree of decentralisation. Profit (or loss) that is assigned to it is the pivotal instrument to guide and control a profit centre .	m
Project	A project is an initiative that runs for a limited time only. It is characterised by a particular objective and by dedicated planning and management processes. With respect to organisational structure, it is important that a project may require the definition of a corresponding temporary organisational structure .	m
Qualification	Describes the abilities a human actor has or should have in order to perform certain tasks or to fill certain positions or roles . There is a wide range of abilities, some of which can be described rather precisely (e.g. being able to type-write at a certain speed, or being in command of a particular foreign language) where others do not allow for a comprehensive formal description (e.g. social skills).	m
Resource	In general the input that is required to perform a process or task . Resource is an abstraction of human actors , machines, devices and material required to produce products or services. Information is often regarded as a resource , too. However, with respect to the outstanding importance of information for the design of information systems, more specific terms – such as document, object, service etc. – are better suited.	i
Responsibility	An organisational unit , a role or a committee can be assigned a responsibility for a set of tasks or processes – either explicitly or implicitly (for instance by defining responsibility through goals that are to be pursued).	m
Role	A role is defined by a set of responsibilities or tasks . It is usually less formal than a position - and it is orthogonal to position : An employee who holds a position can also fill a set of roles .	m
Service	The term “service” serves as an abstraction that fosters separation of concerns, i.e. division of labour . It focusses on the outcome of activities or operations rather than on the way they are performed. To promote reliability, it emphasises contracts that specify obligations of service providers and clients. A service may represent outcomes of simple or complex activities . The term	m

	also covers services provided by software. However, in the context of organisation modelling, there is emphasis on organisational services, i.e. services provided by organisational units .	
Service Diagram	A diagram that represents organisational services together with respective contracts and related organisational units .	m
Span of Control	Assigned to a position : the number of directly subordinated positions	a
Specialisation	Specialisation aims at dividing labour to a high degree. Specialisation is motivated by the hope for increasing productivity through the development of special skills and routine.	a
Staff Position	A position that is part of a Staff Unit. Usually, holders of respective positions are charged with gathering and summarising information and giving technical assistance to generalist managers who are responsible for making final decisions.	m
Staff Unit	An organisational unit of experts that is directly assigned to one organisational unit (usually a top level unit). It has not superior or subordinated organisational unit (it is not a 'line' unit).	m
Subordinate	Assigned to a position : a subordinated position – sometimes used to denote an employee who holds a subordinated position	m
Substitute	A substitute can be defined on different levels of abstraction. A position may be assigned as substitute to another position meaning that a corresponding employee serves as a substitute . It is also possible to define a role as a substitute of another role . Finally, a substitute can be defined on the level of particular employees , e.g. Sam Smith is the substitute of John Miller – both holding the same position .	m
Superior	Assigned to a position : a superior position – sometimes used to denote an employee who holds a superior position	m
Task	A task is characterised by a non-empty set of goals it is to accomplish. It can be more or less complex. A task is performed by one or more human actors . Performing a task may require to run a process . It requires, however, at least one human actor . The definition of a task is usually related to motivation and responsibility .	m
Job Enlargement	Increasing the amount of work assigned to a position or role	a
Job Enrichment	Increasing the variety of work and the overall responsibility assigned to a position or role	a
Job Sharing	Assigning a position or role to more than one employee ; typically accompanied by a reduction of total working hours of the involved employees .	a

Table 1: Glossary of Key Terms

2.2 Semantic Peculiarities

In addition to the generic terms represented in the glossary, designing organisational structures involves further, more specific terms that can be regarded as part of a corresponding technical language, too, e.g. “team”, “department”, “central department”, “division”, “sales department”, “marketing department”, etc. However, as we shall see, it is not trivial to decide whether terms that are part of a technical language are suited to be reconstructed as concepts of a corresponding domain-specific modelling language. To illustrate this challenge, the terms of the glossary that are regarded as directly suited for modelling organisa-

Organisational Structure: A Trivial Subject?

In addition to the semantic net of generic terms, the semantic nets shown in Figure 3 represent more specific terms that, nevertheless, will be used in many organisations.

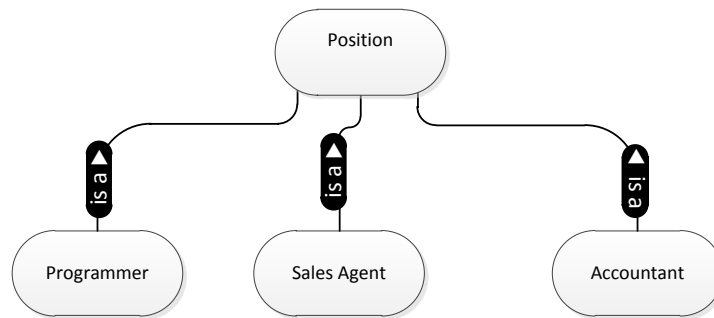


Figure 4: Refinement to Specific Positions

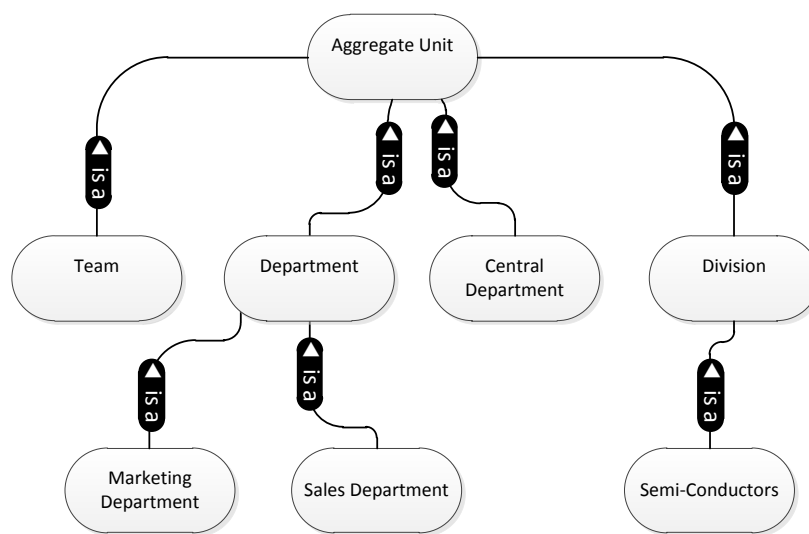


Figure 5: Refinement to Specific Organisational Units

Analysing the semantic nets leads to a number of questions that need to be clarified in order to specify the OrgML:

Incomplete semantics: For instance, collaboration could occur between committees or between organisations. Like a few others, these possibilities are not accounted for. Since multiplicities are not represented at all, the corresponding semantics remains unclear.

Ambiguity: For instance, the predicate “is a” is apparently overloaded. In some cases, it seems to represent a specialisation relationship, e.g. “Aggregate Unit -> Organisational Unit”, in other cases it represents rather an instantiation relationship, e.g. “Programmer” -> “Position”. Furthermore, there are circles, e.g. “Role” is associated to “Activity” and to “Task”, where “Activity” and “Task” are associated themselves. This leaves the question whether it is possible that a role can be associated to a task without being associated to the corresponding “Activity”.

Unclear level of abstraction: A further source of ambiguity results from the fact that the concepts are not on the same level of abstraction. For example: “Position” can be interpreted as a meta type which can be instantiated into a set of types (see example in Figure 4). Hence, “Programmer” would represent a type. For “Department”, the level of abstraction is more difficult to decide. It could be regarded as a type with instances such as “Marketing Department”. At the same time, “Marketing Department” could be regarded as a type that represents the set of particular marketing departments.

Concept contingency: This problem is related to the previous one. It addresses the question whether the use of a concept is invariant within the entire range of the modelling language’s intended applications. Only then, it would make sense to include the concept in the language. While it seems obvious that the semantics of a particular position, e.g. “Sales Agent” may vary, this is not clear for a concept such as “Department” at first sight.

Competing abstractions: There are different possibilities to express commonalities between concepts – resulting in different abstractions. For example: “Staff Position” has certainly much in common with “Position”. Therefore, specialising it from “Position” would promise advantages. However, at the same time it would result in the problem that a “Staff Position” must not be assigned to any organisational unit. Further examples are similarities between “Position” and “Role” or between “Organisational Unit” and “Committee”.

Unclear relevance for intended modelling purposes: The semantic net includes concepts that are very similar to others and, hence, may be hard to distinguish from those. Examples would be “Activity”, “Task” and “Goal”. With respect to the requirement A3 that language concepts should be clearly distinguished, this might create a problem.

Unclear scope: While some terms, such as “organisation” or “position” may be used in a widely uniform way across many companies, there are other terms the meaning of which may vary in different organisation. Therefore, it is required to specify the intended scope of applying the language.

2.3 Scope and Purpose

The OrgML is supposed to cover a wide range of existing and possible future organisations. The purposes to be addressed are outlined in Frank 2011b). This intended focus is based on the following assumptions and objectives:

Common foundation: The (re) construction of organisational structures in (post) industrial societies can be adequately achieved by a set of common concepts. Rationale: The conception of organisational structure itself is based on the cultural background characteristic for (post) industrial societies. The diversity found in existing technical languages does not have to reflect necessary conceptual differences, but may also be the result of a partially arbitrary evo-

Organisational Structure: A Trivial Subject?

lution. As a consequence, there is a chance for a unified language to describe/reconstruct existing organisation structures adequately. Note that this does not exclude certain conceptual differences (see below).

Contribution to reuse and integration: Only if the targeted language is suited for a wide range of organisations, wide-range reuse and respective economies of scale are possible. This relates not only to corresponding tools, but also to human expertise, which is easier to accumulate if it is based on a common terminology.

Despite the assumption of and the quest for commonalities, there is terminological diversity. On the one hand, it is related to different types of organisations. In public administration or educational institutions, one will find other types of organisational units than in industrial enterprises. For instance: A “chair” is an organisational unit specific to universities. On the other hand, it is related to the fact that the meaning of terms may vary with the organisational context. For instance: A department may be a top level organisational unit in one organisation, where it is part of a head department or a division in other organisations. Hence, it is required to decide which concepts are invariant throughout the intended scope of the language and which may rather be subject of individual specification. The next section is aimed at analysing these questions.

2.4 Specific Requirements

Our brief analysis of concepts to model organisation structures has – among other things – resulted in the insight that there is a technical terminology that provides a foundation for specifying a corresponding modelling language. However, there are two peculiarities to be accounted for. While a certain term may be used in various organisations, its meaning may be more or less different. The second peculiarity is related to the level of abstraction. Conceptual modelling is usually aimed at representing types. This is for a good reason: A conceptual model is intended to serve as an abstraction that is independent from states or changes of particular instances. Also, it may be intended to serve as a schema that allows to instantiate an entire population of instances of a certain kind. While models of organisation structures are also intended as abstractions of organisational reality, they usually represent instances of organisational units such as a particular marketing department, a particular product division etc. This peculiarity of modelling organisation structures is sobering for two reasons. On the one hand, focussing on instances is a threat to reuse and integrity. On the other hand, it creates a problem for specifying the language. Usually, meta modelling languages are supposed to specify meta types that constitute the concepts of a corresponding modelling language. However, if models are located at the instance level, the corresponding modelling language would require types, i.e. the meta modelling language should allow for specifying types. The latest version of the MEMO MML (Frank 2011a) allows to specify language concepts as types.

3 Analysis of Potential Language Concepts

The subsequent analysis of key terms of the relevant technical language serves two purposes. On the one hand, it is aimed at checking whether the previously considered concepts are sufficient. On the other hand, it serves to assess conceptual variety and to identify the level of abstraction represented by certain model elements.

3.1 Analysis of Key Terms

Organisational structures are a contingent subject: There is a remarkable variety and ambiguity of the corresponding technical language. To illustrate this problem, we will look at a three selected rather generic terms that are likely to be found in most textbooks on organisation studies: *organisational unit*, *department*, and *position*. Hence, they can be regarded as part of the corresponding technical language, which makes them candidates for being incorporated into the MEMO OrgML as well. In order to further analyse this assumption, we will apply the criteria specified in **modelling rule R3**. As we shall see, in the case of modelling organisation structures, a further challenge has to be accounted for: preferences and expectations of prospective users concerning the abstraction of models. Table 2 shows the result of evaluating the three terms against the criteria of modelling rule R3. The description of the terms refers to the glossary presented in Frank 2001b). “+” indicates that the term clearly fulfils the corresponding criterion. “o” expresses that it fulfils the criterion to a satisfactory degree, while “-” indicates that it fails to satisfy the corresponding criterion. “c” means that – with respect to the considered criterion – the use of the term is contingent, i.e. in some cases it fulfils the criterion, in others it does not.

Organisational Unit		
An organisational unit is a part of an organisation (institutional) that reflects a permanent principle of the division of labour. An organisational unit may contain other organisational units. The definition of organisational units can be based on functional aspects (e.g. 'Finance', 'Production', 'Marketing' ...), on objects (e.g. 'trucks', 'sport cars' 'power train'), market-oriented (e.g. 'North America', 'Europe', 'consumer', 'reseller' ...) or combinations of these. Usually there is a position or a board that is in charge of an organisational unit.		
a) invariant semantics	The term is used on a high level of abstraction. The essence of its semantics should not vary substantially.	+
b) number of types	Throughout the intended applications, there is a plethora of different types of organisational units. Also, in most organisations, there will be a noteworthy number of different types. Examples for types could be “Division”, “Department”, “Finance Department” etc.	+
c) variance of types	The semantics of types of organisational units can vary to a large degree.	+
d) instance as type	At least some of the potential instances will be regarded as types almost intuitively, e.g. “Department”, “Division”. Other potential instances, such as “Marketing Department”, “Consumer Electronics Division”, will probably not be regarded as types by many. Hence, the final assessment of this criterion depends on the recommended instantiation.	+ c

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

e) number of instances	Again, for some potential instances, like “Department”, “Division” etc., there will be many instances, while other potential instances will have only one instance themselves.	+ c
f) relevance	This is a key term for describing organisation structures.	+
Department		
A department is an organisational unit, which will usually include further organisational units. It may be part of a superordinate organisational unit.		
a) invariant semantics	As the description of the term indicates, there is not much that is specific to the term department, which would clearly distinguish it from the more general term organisational unit. Even on a high level of abstraction, the meaning of the term varies. In some cases, a department will be part of a superordinated central department or a division, in other cases not. There is, however, one invariant aspect. It is related to the aggregation hierarchy. Certain aggregations are in general not possible, e.g. a department must never include a central department.	-
b) number of types	Throughout the intended applications, there is a plethora of different types of departments. Also, in most organisations, there will be a noteworthy number of different types. Examples for types could be “Finance Department”, “Marketing Department” etc.	+
c) variance of types	The semantics of department types may vary to a noteworthy degree.	+
d) instance as type	In many organisations, a term such as “Marketing Department” will not be regarded as a type, i.e. an abstraction over particular departments. However, it is conceivable that this is different in rare cases. Within a university, for instance, there could be three different types of departments, such as academic, administrative and service.	- c
e) number of instances	In most cases, there will be no more than one instance.	- c
f) relevance	This term is used frequently for describing organisation structures.	+
Position		
A position is the smallest organisational unit . It does not contain any other positions. Usually, a position is assigned to one employee. There are, however, exceptions of this rule (“job sharing”).		
a) invariant semantics	The term is used on a high level of abstraction. The essence of its semantics should not vary substantially.	+
b) number of types	Throughout the intended applications, there is a plethora of different types of positions. Also, in most organisations, there will be a noteworthy number of different types. Types could be, for instance, “Head of Department”, “Sales Representative” etc.	+
c) variance of types	The semantics of position types can vary to a large degree.	+
d) instance as type	Most of the potential instances will be regarded as types almost intuitively, e.g. “Sales Assistant”, “Sales Representative”. It seems that this is the case, too, for those types that usually have only one instance, e.g. “CEO”, because the specification of the position is usually independent from the actual instance.	+
e) number of instances	Again, for some potential instances, like “Department”, “Division” etc., there will be many instances, while other potential instances will have only one instance themselves.	+ c
f) relevance	This is a key term for describing organisation structures.	+

Table 2: Assessment of key terms

The assessment presented in Table 2 suggests including the terms “Organisational Unit” and “Position”. “Department” is more difficult to judge. On the one hand, excluding “Department” from the language has an unpleasant implication: It is possible to design models that obviously contradict commonly agreed on semantics. One could, for instance, express that the type “Department” includes a type “CentralDepartment” without violating the language’s syntax or semantics.

On the one hand, it seems not to be a satisfactory candidate, since its semantics is not sufficiently invariant.

3.2 Levels of Abstraction

Unfortunately, this is not the only problem. Figure 6 illustrates the possible conceptualisation and instantiation of the two potential meta types OrganisationalUnit and Position.

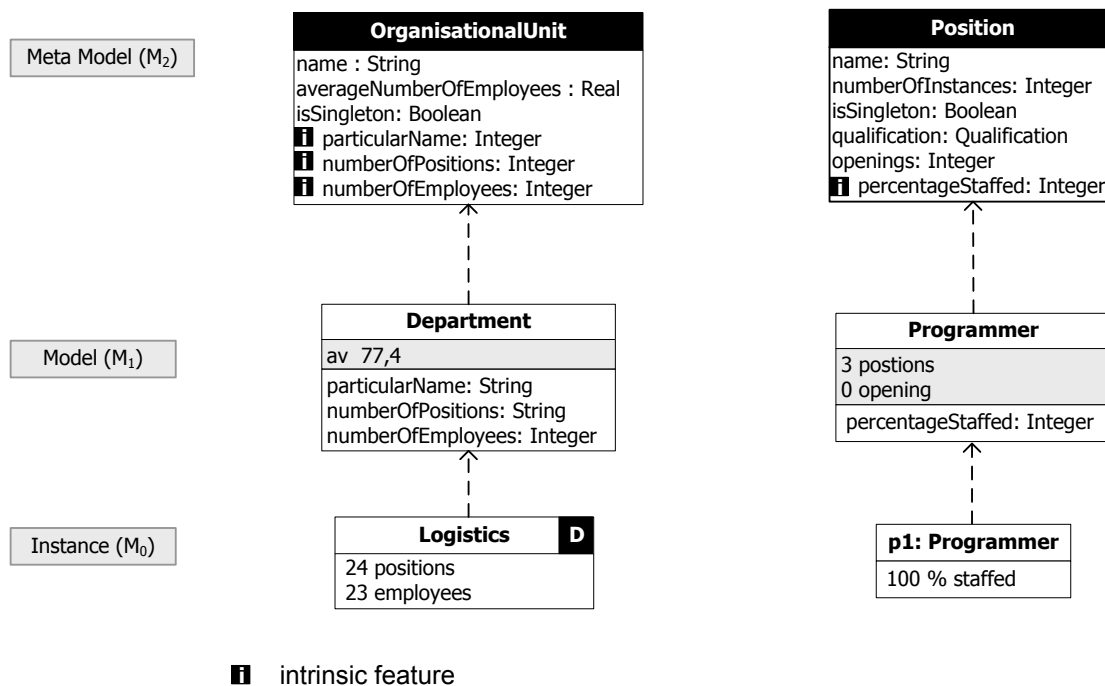


Figure 6: Instantiation of OrganisationalUnit and Position

Both meta types in the example include intrinsic features, which are supposed to be initialised only on the instance level. In conceptual modelling the focus is on types. This is for good reasons. It facilitates models that are fairly stable by abstracting from ever changing instance populations and instance states. Also, representing a plethora of instances compromises the clarity of a model. According to this principle, models of organisational structures would be comprised of types such as “Department” or “Programmer”. Figure 7 shows an example of a corresponding model as well as an excerpt of a potential instance of this model. The model

on level M_1 can be interpreted as a schema for organisational structures within a certain context, e.g. for a certain enterprise. One could also regard this model as a specification of the particular terminology used in this context: It defines the meaning of otherwise contingent terms like “department” or “division”.

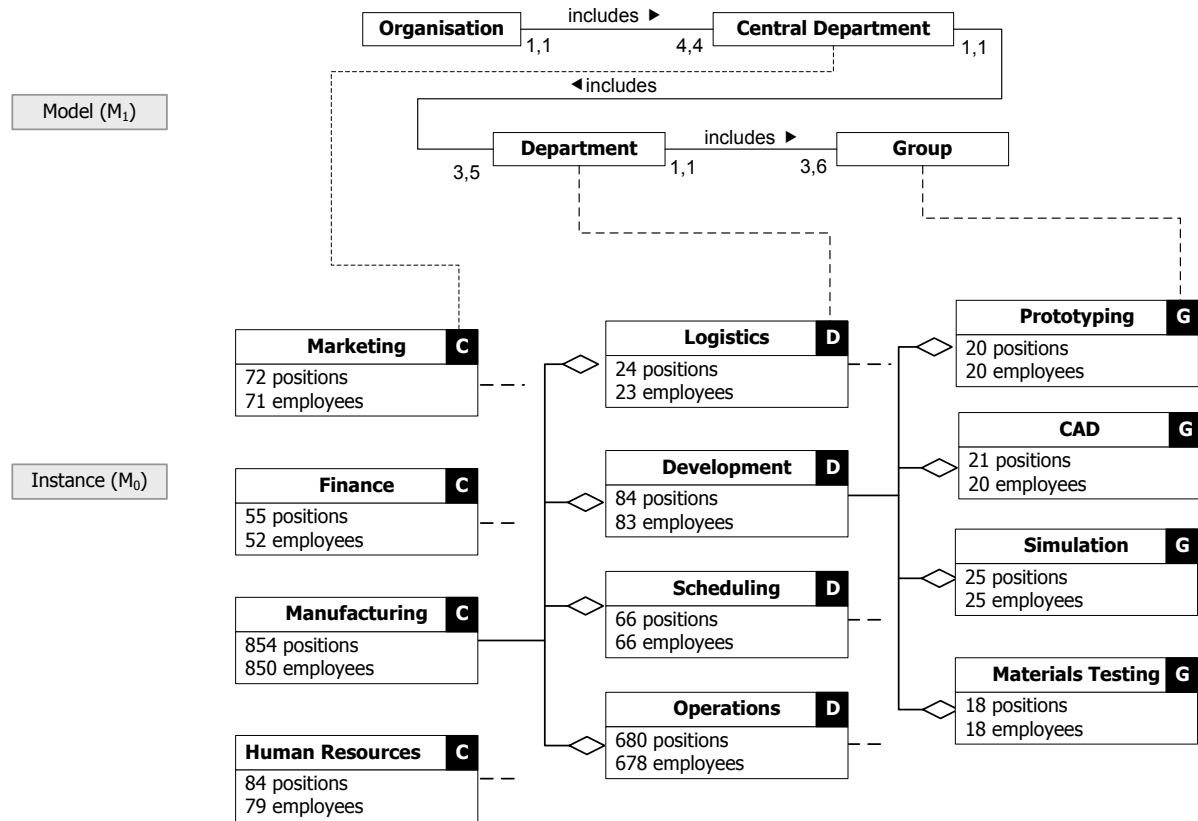


Figure 7: Model of organisation structure on type level and excerpt of corresponding instances

Although this structure is conceptually clean, it is not satisfactory. For many kinds of analysis, the M_1 model remains too abstract. Often, it is important to refer to a particular organisational unit – and not just to a type that represents an entire range of units. Therefore, the M_1 models would need to be supplemented by a valid instance in most cases. However, many users would regard this as hard to understand and – more important – as an unnecessary burden, too. It is probably hard to understand not just because it requires the distinction of two levels of abstraction. It would also demand for a perspective that is clearly different from organisational charts, which many prospective users are familiar with. It will be regarded as redundant by many, because among the many flavours and interpretations of organisational charts, they might usually find what they need. With respect to the acceptance of a language, expectations of prospective users are of major relevance. However, that does not mean that they have to be sacrosanct. If there is a clearly superior approach, users may change their preferences – at least in the long run. However, in many cases the benefit of an additional

schema level will be questionable: Usually, there is only one instance of the schema, which includes relevant semantics the schema does not provide. Therefore, it seems more appropriate to target a level of abstraction as it is represented on the instance level. This requires finding an approach that would allow for instantiating such a model from the meta model. Before we target this problem, we need to account for a further challenge. Figure 8 shows a version of the previous example that is extended by positions.

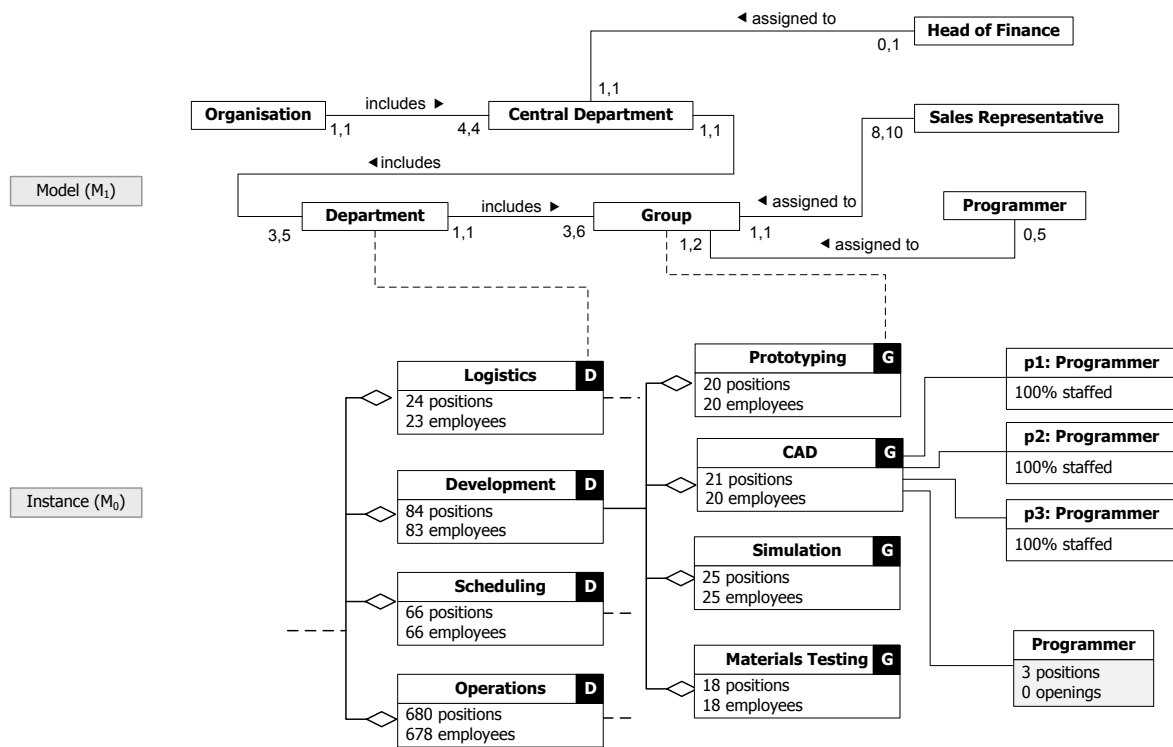


Figure 8: Extending the example by positions

The example supports the assessment that the abstraction shown on level M₁ is of little benefit. The definition of the position types suffers from the fact that the organisational units they are assigned to are not specified. As a consequence, the definitions remain superficial and allow for inconsistent instantiations – e.g. for assigning “Head of Finance” to “Marketing Department”. Furthermore, the example illustrates an additional challenge. While level M₀ implicates to model instances, this can be odd for positions. Especially in those cases, where a position type has a multitude of instances, there will usually be no point in representing all of them – unless they differ in relevant aspects. Using a position type instead – as with “Programmer” in the example – will often seem as the more appropriate choice. However, it would challenge the language architecture: M₀ is intended for representing instances only. Also, sometimes it makes sense to account for position instances. On the one hand, there seems to be a demand for representing exposed positions in an organisational chart on an instance level, e.g. by displaying the names of the corresponding employee. However, for

analytical purposes, which are our main focus, this would not be as important. On the other hand, there are kinds of analysis that suggest accounting for position instances – but not necessarily for specific instances. For example: Within a business process model, there are a few subprocesses which refer to the position “Sales Representative”. The position could be represented as type. But if one would like to express that there need to be two positions of this type involved – in different stages of the process – it must be possible to refer to instances somehow.

So far, our analysis produced a rather sobering result. The semantics of terms denoting organisational units, such as “department”, “central department”, “division” etc. is varying to a remarkable degree, which does not make them a favourable choice for being embedded in the language. Defining them on the M_1 layer, however, results in models that are of little benefit. Furthermore, there will be kinds of analysis that require instance level data – such as the number of employees or positions. Modelling positions faces the challenge that sometimes instances have to be accounted for on the actual M_1 layer. Against this background, it is obvious that the language specification requires trade-offs. Before the suggested solution is presented in the next section, we will give an overview of further key concepts – in addition to “organisational unit” and “position” – that are candidates for becoming part of the language. Note that we do not include concepts that are related to projects.

Description		Rule R2
Organisation	An organisation is a goal-oriented social or socio-technical system – like a business firm, a non-profit organisation, public administration etc. While it could be regarded as the top-level organisational unit, it makes sense to define it as a special concept, because it has features that do not apply to organisational units.	Most criteria are satisfied. However, modelling a specific organisation as a type may not seem intuitive (d). Also, usually, there will be one instance only (e).
Role	A role is defined by a set of responsibilities, the actor who fills the role, has to perform. It is usually less formal than a position - and it is orthogonal to position: An employee who holds a position can also fill one or multiple roles at the same time.	All criteria are satisfied.
Committee	A committee consists of group of people. Committees can address a wide range of different purposes. Some may make decisions that are mandatory for other organisational units. Others may serve counseling purposes only. In any case, a committee does not include positions, but only roles. The challenges that are related to expressing different levels of abstraction are dealt with by using intrinsic features. Note that the graphical notation may be subject of further revisions.	Most criteria are satisfied. Often, there will be only one instance of a certain type of committee. In these cases, criteria d) and e) may not be satisfied.
Board	A board is an organisational unit with command line authority. A board can include positions and roles. Note that within the language the concept is used in a wider sense than in <i>board of directors</i>	similar to “committee”

	(which would be a special type of board).	
--	---	--

Table 3: Assessment of Key Terms

Our analysis shows that the technical terminology used in organisational analysis and design is very challenging because it is overloaded to a large extent. This is the case for the variety of particular interpretations, e.g. of terms such as “department”, “board”, etc. and for using a term on different levels of abstraction, i.e. on an object and a meta level. It seems appropriate to restrict models of organisational structures to concepts and avoid representing instances, e.g. “John Myers, CIO”. One could also speak of an organisational schema, which could serve to enrich instance-level representations with semantics. We will see, however, that for pragmatic reasons, insisting on a clear distinction between types and instances is not always advisable.

4 Language Specification

To prepare for the specification of the OrgML, we will first analyse relevant design challenges and suggest respective decisions. Against this background the conceptualisation of the language's terms are presented, followed by the meta model. Finally, the language's graphical notation is introduced.

4.1 Basic Design Decisions

The previous discussion of core concepts resulted in the frustrating insight that the peculiarities of the targeted domain and the corresponding universes of discourse seem to not allow for a conceptually convincing language design. As a consequence, the suggested solution represents a trade-off. With respect to conceptual clarity and modelling aesthetics, it is certainly not satisfactory. The compromises it makes reflect the ambiguity of the corresponding technical languages, which cannot entirely be resolved without jeopardising the ergonomics of the language. The solution is characterised by seven specific concessions:

Customised local pseudo meta types: Although terms like “department”, “division” or “central department” that denote types of organisational units are not perfectly suited as language concepts (see Table 2), they are nevertheless important language terms in many application areas. Specifying these terms as types on the M_1 level would result in an abstraction (schema of organisational structure) that is of little value only. Alternatively the level of abstraction they represent could be entirely omitted by directly instantiating “Organisational Unit” into types such as “Marketing Department”, “Finance Department” etc. However, this would not be satisfactory either, because it would not be possible to express that a marketing department and a finance department (both defined on the M_1 level) are two types of the same kind. Against this background the following compromise has been made. Since terms such as “department”, “head department” etc. are not used consistently throughout the set of intended applications, they are not incorporated into the language – in order to not violate modelling rule **R2**. However, these terms are an important part of the technical language used by organisers. Also, within a certain domain, their semantics can be reconstructed fairly well. The local “type” of an organisational unit, e.g. “department”, “head department” etc., could be specified as an attribute within the meta type `OrganisationalUnit`. This would violate modelling rule **R3** (no type differentiation through entity states). However, it would still comply with modelling rule **R5**, the relaxation of **R3**. However, using an attribute would imply the redundant specification of local types. In order to protect model integrity, the local type of an organisational unit can be defined by an instance of an additional meta type (`LocalUnitType`), which can be associated with an instance of `OrganisationalUnit`. `LocalUnitType` provides the option to specify the level and a designation that are characteristic for a certain

type of `OrganisationalUnit`. Furthermore, it would be possible to assign a specific graphical notation. A level is defined by a positive Integer. The lowest level, 0, is reserved for those organisational units that do not include further units and/or that do not include any position that is in command of further organisational units. If these levels are specified for all local types of organisational unit types, it is possible to enforce the constraints that they imply, e.g. that no organisational unit on level 2 can be part of a unit on level 1. The definition of a local terminology through instantiations of `LocalUnitType` can be regarded as defining language concepts with a limited scope, or as pseudo meta types. This approach can be further supported by supplementing the language with instantiations of `LocalUnitType` for specific types of organisation, e.g. for industrial enterprises, for public organisations, universities etc. On the one hand, it still satisfies the demand for flexibility: the meta type `OrganisationalUnit` can be instantiated into any particular type of organisational unit. On the other hand, it accounts for model integrity.

Modelling organisational units as types: Although our previous analysis of candidate terms revealed that a term such as “marketing department” will usually not be interpreted as a type, it is regarded as a type in the suggested language architecture. Furthermore, every organisational unit type is regarded as having one instance only (“singleton”). This is for two reasons: Firstly, it reflects the level of abstraction indicated for M1. Secondly, it emphasises that a model of an organisational structure should not only represent the actual situation, but should rather serve as a blueprint, i.e. a schema, for further instantiations to come.

Including instance level specifications into types: Sometimes, there is demand for including data that refer to instance states. For example: Certain kinds of organisational analysis recommend accounting for the actual performance of organisational units. This is hardly a feature of a type. As a response to this demand, the description of types can be enhanced by features that apply to the instance level. This is accomplished through the use of intrinsic features. If it is required, intrinsic features can be initialised in order to show properties of particular instances, e.g. the name of an employee associated with a position. In this case, the conceptual model would be enhanced by corresponding instances.

Partial hiding of type/instance dichotomy: The language is not an end in itself. It is intended to serve its prospective users as an effective instrument. Forcing users to distinguish between type and instance level may be regarded as confusing by many. Therefore, this distinction is widely hidden from users – without compromising its potential.

Disaggregation of position types: Sometimes, a position type is assigned to more than one organisational unit. In this case, using only one representation of a type would not allow for expressing the share of corresponding instances assigned to each organisational unit. Take, for instance, a position type “Personal Assistant”. Describing the type could include the number of current instances. If one assigned this type to more than one organisational unit without further specification, it would not be clear how many instances belong to each of the

corresponding units. Therefore, it is possible to “disaggregate” the representation of a type. This is accomplished through the differentiation of a concept that serves to define the type properties, i.e. the essential characteristics of a position type, and a further concept, called “PositionShare”, that serves the representation of sets of positions. Instances of the latter may be assigned to organisational units. In case one does not want to differentiate the shares of position instances assigned to different organisational units, it is possible to use one instance of PositionShare that represents all positions of a certain type and assign it to the entire organisation.

Prototypical position: For analytical purposes it can be required to distinguish between instances of position types. However, that does not mean that one would refer to a particular instance. Instead, one would rather use an abstraction of any instance that would have an identity of its own in the represented domain. We call this abstraction a prototypical instance or – more specific – a prototypical position. It can also be used for simulation purposes.

The example in Figure 9 gives an impression of the core concepts and the level of abstraction they represent. Note that the example serves illustration purposes only. It does not reflect the final specification, nor does it present the final graphical notation. The instance level is not represented – except for prototypical positions. By definition, organisational unit types have one instance only. Therefore there is not much need for representing instances of organisational unit. Position types, on the other hand, may have multiple instances. But in most cases, representing every single instance would compromise a model’s clarity. Nevertheless, there are cases, when instance level data is relevant, e.g. the number of instances of a position type, or the number of employees of an organisational unit type. Therefore, it is possible to represent instance level data on the type level by using intrinsic features. The number of employees of a type can be interpreted as the sum of all employees of its instances. In the case of one instance, it would be exactly the number of employees of this instance.

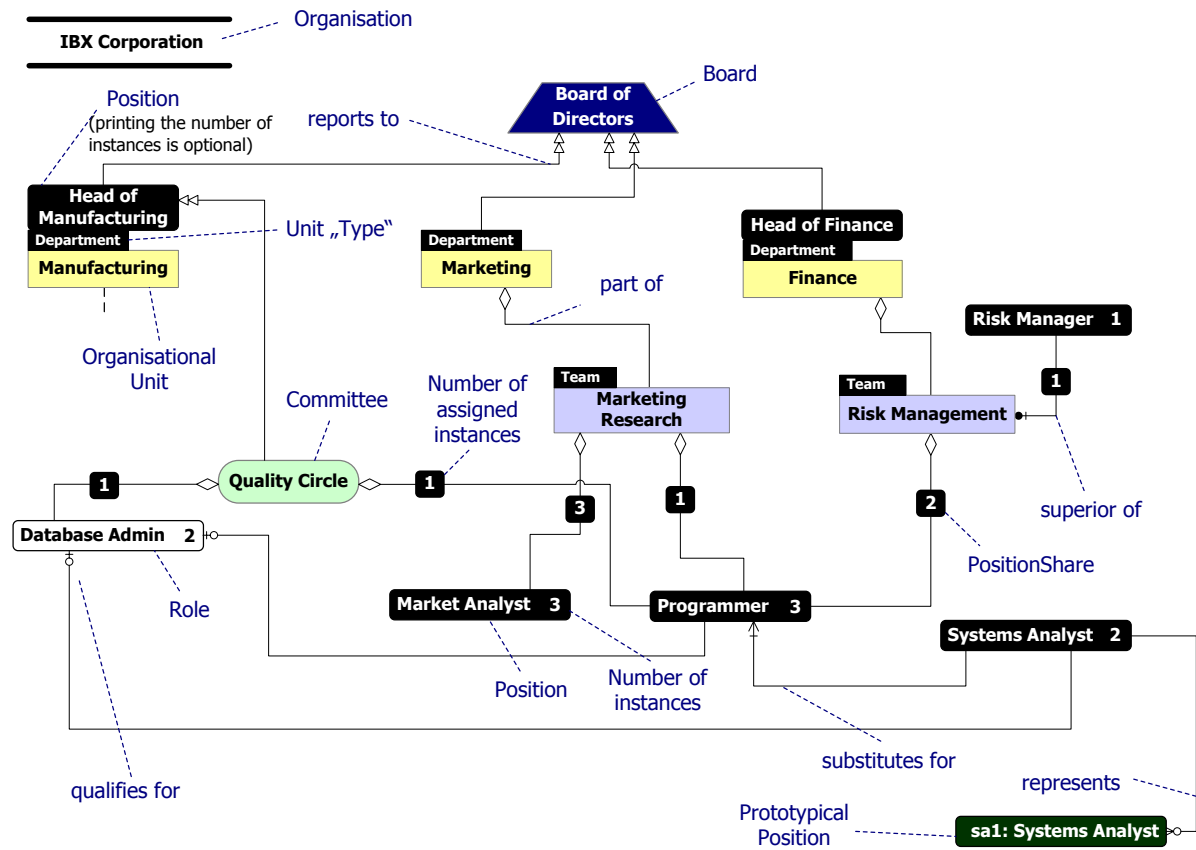


Figure 9: Illustration of core concepts

Relationships between organisational units are essential for models of organisational structures. They can be divided into *aggregations*, which serve to indicate that one organisational unit is part of another, and *interaction relationships*. An interaction relationship in its most general form indicates that there is a regular interchange between organisational units. This interchange can be further characterised by its intensity, the media that are used and its quality. A *command relationship* is a special case of an interaction relationship. In its generic form, it indicates that an organisational unit may issue instructions towards other units. More specific types of command relationships restrict instructions to objects, e.g. products or product lines, to disciplinary issues or to functions, e.g. marketing research. Note that the concepts considered here may also be involved in other types of relationships that can be established with concepts of other models, e.g. business process models.

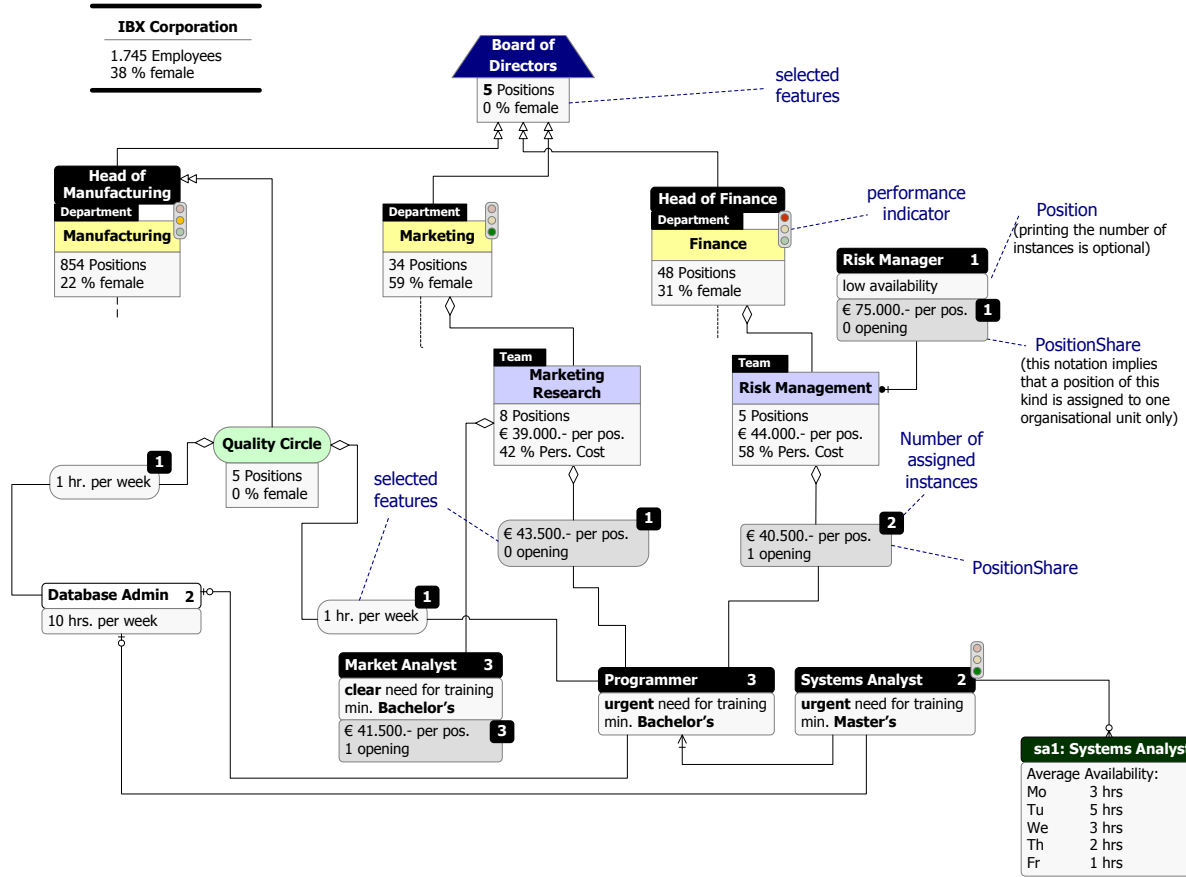


Figure 10: Representation of selected features

4.2 Language Concepts

The example model in Figure 9 and Figure 10 illustrate the concepts of the language and the corresponding graphical notation. It should give an impression of their core meaning. Against this background, Table 4 to Table 11 present the language concepts in more detail. The concepts include a relatively large number of features. In order to avoid a confusing representation it will usually be appropriate to select the set of features that is relevant for a particular perspective or a certain kind of analysis. The distinction between instance and type level may be regarded as confusing (and redundant) by some. However, it is important to make this distinction in the language specification. This is for two reasons. Firstly, it is imperative with respect to conceptual clarity. Secondly, the language may include instance level features that are not intended for designing models on the M₁ layer. Instead, they serve as a conceptual foundation for managing corresponding instance populations. Though it may seem unusual to some, the distinction is yet straightforward: Instance level features describe those characteristics that are related to a specific occurrence, e.g. the actual performance of an organisational unit. Type level features serve two functions. On the one hand, they provide a blueprint for defining a specific type regardless of a particular instance. This could be,

Analysis of Potential Language Concepts

e.g. the mission of the organisational unit type such as “R&D Department”. On the other hand, they allow for describing properties of a type that are aggregated from the set of its instances. For example: The number of existing positions of the type “Research Assistant”. Nevertheless, it is conceivable to hide most of the distinction between instance and type level from users.

Some attributes are introduced only to support simulations. They are marked with <<simulation>>, which corresponds to the attribute simulation of MetaAttribute in the MEMO meta meta model (Frank 2011a). If an attribute on the instance level could be initialised through data provided by an object in an information system, it is advised to establish a corresponding interface to obtain it from an external source (in case of using the language within a modelling tool). Such an interface could either allow for establishing a reference or for copying a set of values. These attributes are marked with <<obtainable>>, which corresponds to the attribute obtainable in the MEMO meta meta model. If it is conceivable that the value of a feature could be derived from other parts of the model, such a feature is marked as <<derivable>>. Hence, a feature that is derivable indicates a possible case of redundancy. While redundancy should definitely be avoided in a conceptual model, accounting for requirement U3 recommends not forcing users to apply a certain level of detail: Sometimes it may be regarded as appropriate to assign the value of a feature like “number of positions” to each organisational unit – which would make the corresponding value for the entire organisation redundant. In other cases, however, it may be sufficient to assign this value to the entire organisation only. Note that a feature value could be both obtained from an external source and from other parts of the corresponding model. A precise specification of the features described below is given with the meta model. Note that the constraints correspond to OCL constraints defined with the meta model. The number of the corresponding constraint is printed in white in a read rectangle.

Organisation	This concept serves two key purposes. Firstly, it allows for modelling interactions between different organisations. Secondly, it provides an abstraction over organisational units and positions – hence, over its parts. Thereby it is possible to define invariants that apply to all parts of an organisation. Regarding an organisation that represents e.g. a specific firm is somewhat artificial. Nevertheless it is not absurd. The essential, invariant features of an organisation can be interpreted as constituting its type, which could be used as a blueprint for creating further instances.
Example Instantiations	
“IBM Corporation”, “BMW AG”, “University Duisburg-Essen”	
Attributes on type level	
name	The name of the organisation.
mission	Serves to describe the essential mission and the responsibilities on a high level

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

orgDimension	In case one dimension for defining organisational units – such as “functional” (e.g. “Finance Department”, “Accounting Department”), “object-oriented” (e.g. “Division Storage Technology”, “Division Consumer Electronics”) is obligatory for building organisational units, it could be defined here – as a global invariant.
maxLineOfCommand	The maximum number of superiors, a particular organisational unit or position may have.
<<derivable>> levels	The number of management levels in the organisational hierarchy.
Attributes with reference to instance level	
<<derivable>> averageSpan	Each position within an organisation that is superior to other positions is characterised by a specific span of control, i.e. the number of directly subordinated positions. This attribute serves to represent the average span of control across all positions in an organisation. It is conceivable to regard this feature as a type level attribute. However, since the number of positions and as a consequence the average span of control may change, it seems more appropriate to place it on the instance level.
<<obtainable>> <<derivable>> numberOfEmployees	The actual number of employees working for the organisational. This value can be obtained from a corresponding information system, assigned manually, or it can be calculated from the corresponding values of the organisational units that are part of the organisation.
<<obtainable>> <<derivable>> genderRatio	The percentage of females among the employees. Again, this value can be obtained from a corresponding information system, assigned manually, or it can be calculated from the corresponding values of the organisational units that are part of the organisation.
<<obtainable>> <<derivable>> numberOfPositions	The actual number of positions assigned to the organisation. Often, this number will be the same as for the attribute employees. But it does not have to be, since positions may be vacant or filled by more than one employee. It can be obtained from an external system, assigned manually or calculated from the positions assigned to this organisational unit and all its subunits.
<<obtainable>> <<derivable>> averageCostPerPosition	The average cost of a filled position. This value can be provided by an external system. It can also be calculated from the values assigned to the positions of the included organisational units (provided the costs are specified for all positions).
<<obtainable>> shareOfPersonnelCost	The share of costs for personnel from the overall costs – to be obtained from an external system or calculated from corresponding values of the organisation’s organisational units.
<<obtainable>> <<derivable>> shareOfGraduates	This value can be provided by an external system. It can also be calculated from the descriptions of the positions assigned to the included organisational units (provided the formal qualification is specified for all positions).
shareOfUnderPerformers <<derivable>>	The share of positions that are assessed as not performing satisfactorily. This value can be aggregated from corresponding values of the included organisational units, for which in turn the value would be calculated from corre-

Analysis of Potential Language Concepts

		spending values of the assigned position types or positions.
Relationships on type level		
Composed_Of (corresponds to part_Of for OrganisationalUnit)		With OrganisationalUnit . An organisation can be composed of one or more (0,*) organisational unit types. On the other hand, an organisational unit type can be part of one (this is the default) Organisation .
includes (corresponds to part_of for PositionShare)		With PositionShare . An Organisation can include zero or many PositionShare . A PositionShare is assigned to zero or one Organisation .
hosts (corresponds to hosted_by for Committee)		With Committee . A type of organisational can host zero or more (0,*) committee types. On the other hand, a committee unit type can be hosted by one (default) or more than one organisational unit type.
part of (corresponds to includes for OrganisationCategory)		Allows for specifying a category, e.g. “multi-national corporation” or “educational institution”. To foster a consistent use of categories, it is recommended referring to a corresponding, central dictionary. In cases where only one organisation is represented, this attribute is of limited value.
subordinatedTo (corresponds to superior_of for Position, Role or Board)		With Position, Role or Board . An Organisation can have zero to many superiors.
supervisedBy (corresponds to superior_Of for Position, Role or Board)		With Position, Role or Board . An Organisation can have zero to many supervisors.
Constituted_as (corresponds to constitutes with LegalForm)		With LegalForm . An Organisation has zero to one LegalForm .
customer_of		With Organisation . Zero to many organisations can be customer of zero to many other organisations.
supplier_of		With Organisation . Zero to many organisations can be supplier of zero to many other organisations.
cooperates_with		With Organisation . Zero to many organisations can cooperate with zero to many other organisations
competes_with		With Organisation . Zero to many organisations can compete with zero to many other organisations.
Constraints		
C2	consistent use of orgDimension	If the attribute orgDimension is specified, then the corresponding attribute of OrganisationalUnit must have a compliant value. If, e.g. orgDimension is specified as #functional , it is not possible for an OrganisationalUnit to use #matrix (also defined with OrganisationalUnit).

Table 4: Description of the meta type Organisation

Organisational units may include other organisational units. They cannot be superior to other organisational units (this is reserved to the specific organisational unit **Board**). One could

argue that a position (or a role, or a board) is, in a strict sense, not superior of an entire organisational unit, but only of the corresponding positions. However, often, representing all corresponding positions will be regarded as too much effort. Therefore, it is possible to assign a position, role or board as superior to an entire organisational unit. The multiplicities used for characterising these relationships are somewhat corrupted by the delicate abstractions related to `OrganisationalUnit`, `Position` and `Role`. `OrganisationalUnit` will – in most cases – be a singleton, which implies that a maximum multiplicity of one on the type level means that also not more than one instance can be involved. This is different with `Position` and `Role`, which both can have multiple instances. However, we assume that modelling particular positions and roles is not the appropriate level of abstraction. The concept `PositionShare` serves to assign a certain number of position instances to an organisational unit – if that is required. We assume that there is no need for modelling roles at such a level of detail. With respect to superior relationships, we assume that each position type that is in charge of some unit of work is a singleton. Furthermore, we assume that there is no more than one superior involved in a particular superior relationship. The concept of a `Board` serves to express that a group of people is in charge of an organisational unit or superior to other positions, roles, or boards. A `Board` is also assumed to be a singleton.

OrganisationalUnit	This is a key abstraction of the language. It can be instantiated in various ways. On the highest level of aggregation, it can be instantiated into the type of the entire organisation. To differentiate types according to the technical language used in an application domain, it is possible to refer to a corresponding type description by initialising the attribute <code>orgLevel</code> . Often, there will be only one instance of a type that is instantiated from this meta type.
Example Instantiations	
"Division Electronic Devices", "Marketing Department", "Car Manufacturing Plant", "Human Resources Department"	
Attributes on type level	
name	Allows for assigning a type name. Note that the name of this type will usually be the same as the corresponding instance name since there will often be only one instance.
mission	Serves to describe the mission and the responsibilities.
orgDimension	Serves to specify the dimension the definition of the type of organisational unit is based on. If no value is specified, the value of the <code>OrganisationalUnit</code> , this <code>OrganisationalUnit</code> is part of, applies. The value specified for <code>OrganisationalUnit</code> must not violate the value specified for the corresponding <code>Organisation</code> .
staffUnit	If this attribute is set to <code>#true</code> , the corresponding organisational unit type is regarded as staff unit. The semantics of staff units varies to a remarkable degree. In general, a staff unit is regarded to serve coun-

Analysis of Potential Language Concepts

	selling purposes. Hence, the staff experts are charged with gathering and summarising information and giving technical assistance to generalist managers who are responsible for making final decisions. Staff units are not included in the regular line of command. Hence, a position that is in command of a staff unit cannot be in command of another organisational unit that is not a staff unit.
corporateRelevance	Allows for expressing the relevance the organisational unit has for an organisation's competitiveness. The assigned value represents the degree of relevance, e.g.: 0: no need; 1: could do without; 2: needed; 3: essential.
subjectOfOutsourcing	Allows for expressing whether the corresponding type of organisational unit should be outsourced. The assigned value expresses the degree of urgency, e.g.: 0: no need; 2: should be considered; 3: seems reasonable; 4: urgent need.
subjectOfReorganisation	Allows for expressing whether the corresponding type of organisational unit should be reorganised. The assigned value represents the degree of urgency, e.g.: 0: no need; 2: should be considered; 3: seems reasonable; 4: urgent need. While both attributes, subjectOfOutsourcing and subjectOfReorganisation , may reflect the actual performance of the OrganisationalUnit , hence, an instance level feature, they are still assigned to the type level, because they are related to a potential change of the organisational structure.
Attributes with reference to instance level	
<<derivable>> averageSpan	This attribute serves to represent the average span of control across all positions of the OrganisationalUnit that fulfil managerial functions, i.e. that are superior to other positions.
<<obtainable>> <<derivable>> numberOfEmployees	The actual number of employees working for the organisational unit.
<<obtainable>> <<derivable>> genderRatio	The percentage of females among the employees.
<<obtainable>> <<derivable>> numberOfPositions	The actual number of positions assigned to the organisational unit. Often, this number will be the same as for the attribute employees. But it does not have to be, since positions may be vacant or filled by more than one employee. It can be obtained from an external system, assigned manually or calculated from the positions assigned to this organisational unit and all its subunits.
<<obtainable>> <<derivable>> fluctuation	The fluctuation of all positions assigned to this organisational unit within a certain period (e.g. a year). There are various formula for defining fluctuation. At best, this feature allows for selecting from a list of corresponding concepts. The actual value could be provided by a corresponding information system, e.g. a HRM system.
<<obtainable>>	The value of a performance indicator on an ordinary scale, e.g.: e.g.:

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

performance	0: critical; 1: satisfactory; 2: outstanding. The value can either be calculated from data provided by e.g. an ERP system or assigned manually.
<<obtainable>> <<derivable>> averageCostPerPosition	The average cost of a filled position assigned to this OrganisationalUnit . This value can be provided by an external system. It can also be calculated from the values assigned to the positions of the organisational unit (provided the costs are specified for all positions).
<<obtainable>> shareOfPersonnelCost	The share of costs for personnel from the entire costs assigned to an organisational unit – to be obtained from an external system.
<<obtainable>> <<derivable>> averageAge	The average age of all employees of this organisational unit– also to be obtained from an external system.
<<obtainable>> <<derivable>> shareOfUnderPerformers	The share of positions that are assessed as not performing satisfactorily. This value can be aggregated from corresponding values of the included organisational units, for which in turn the value would be calculated from corresponding values of the assigned position types or positions.
Relationships on type level	
composed_of	With OrganisationalUnit . A type of organisational unit can be composed of zero or more (0,*) other organisational unit types. On the other hand, an organisational unit type can be part of zero or one other organisational unit type. If the attribute orgLevel is specified for the types of organisational unit that are involved in this relationship, the level of the composed type must be higher than the levels of its parts.
hosts (corresponds to hosted_by for Committee)	With Committee . A type of organisational unit can host zero or more (0,*) committee types. On the other hand, a committee unit type can be hosted by one (default) or more than one organisational unit type.
includes (corresponds to part_of for PositionShare)	With PositionShare . A type of organisational unit can include zero or more (0,*) PositionShare . On the other hand, a PositionShare can be part of one or more organisational unit types.
assigns (corresponds to assigned_to for Role)	With Role . A type of organisational unit can assign zero or more (0,*) role types. On the other hand, a role type can be part of zero or more organisational unit types.
characterisedAs (corresponds to characterises with LocalUnitType)	With LocalUnitType . An OrganisationalUnit has zero to one LocalUnitType .
supervisedBy (corresponds to superior_of for Position , Role or Board)	With Position , Role or Board . Multiplicity is zero to many on both sides.
subordinated_to (corresponds to superior_of for Position ,	With Position , Role or Board . This relationship is used in cases where no particular definition of superiority exists or matters. Multiplicity is zero to many on both sides. The number of superiors must not exceed

Analysis of Potential Language Concepts

Role or Board)		the value of <code>maxLineOfCommand</code> within the corresponding <code>Organisation</code> .
<code>functionalSubordinated_to</code> (corresponds to <code>functionalSuperior_of</code> for <code>Position</code> , <code>Role</code> or <code>Board</code>)		With <code>Position</code> , <code>Role</code> or <code>Board</code> . Multiplicity is zero to many on both sides.
<code>objectSubordinated_to</code> (corresponds to <code>objectSuperior_of</code> for <code>Position</code> , <code>Role</code> or <code>Board</code>)		With <code>Position</code> , <code>Role</code> or <code>Board</code> . Multiplicity is zero to many on both sides.
<code>disciplinarySubordinated_to</code> (corresponds to <code>disciplinarySuperior_of</code> for <code>Position</code> , <code>Role</code> or <code>Board</code>)		With <code>Position</code> , <code>Role</code> or <code>Board</code> . Multiplicity is zero to many on both sides.
<code>part_of</code>		With <code>Organisation</code> . An <code>OrganisationalUnit</code> can be assigned to zero or one <code>Organisation</code> .
Constraints		
C8	Exclude staff units from regular line of command	If the attribute <code>staffUnit</code> is specified as <code>#true</code> , it is not possible that the <code>OrganisationalUnit</code> includes any other <code>OrganisationalUnit</code> that has this attribute set to <code>#false</code> . Furthermore, it is not possible that a <code>Position</code> that is in command of this <code>OrganisationalUnit</code> is in command of any <code>OrganisationalUnit</code> other than an included staff unit.
C3	No cyclic <code>composed_of</code> relationships	An organisational unit type A must not be composed of a further organisational unit type B, which in turn is composed of A (Constraint C3).
C1	Not part of unit with lower or equal level	An <code>OrganisationalUnit</code> must not be part of another <code>OrganisationalUnit</code> that has a level assigned (through an associated <code>LocalUnitType</code>), which is lower than or equal to its own level (Constraint C1). Note that this constraint applies only, if the <code>OrganisationalUnit</code> , it refers to are associated with an initialised instance of <code>LocalUnitType</code> .
C2	consistent use of <code>orgDimension</code>	If the attribute <code>orgDimension</code> within <code>Organisation</code> is specified, then the corresponding attribute of <code>OrganisationalUnit</code> must have a compliant value. If, e.g. <code>orgDimension</code> is specified as <code>#functional</code> , it is not possible for an <code>OrganisationalUnit</code> to use <code>#matrix</code> .
C7	No more superiors than <code>maxLineOfCommand</code>	The sum of superiors (functional, object, disciplinary) must not exceed the value of <code>maxLineOfCommand</code> specified for the corresponding <code>Organisation</code> .
C5	No joint use of generic and specific superiority	If a <code>subordinated_to</code> relationship is specified for an <code>OrganisationalUnit</code> , there must be not further special subordinate relationship for this <code>OrganisationalUnit</code> .

Table 5: Description of the meta type `OrganisationalUnit`

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

Position	This concept serves to describe the essential characteristics of a position type. It is supplemented by PositionShare to express features of corresponding instance populations. It is conceivable that certain characteristics of a position type vary with its assignment to different organisational units, e.g. the required qualification. However, to foster a consistent use of position types throughout an organisation, this possibility is excluded. If there is need for this kind of differentiation, it would be necessary to define specific position types.
Example Instantiations	
"Head of Marketing Department", "Sales Representative", "Systems Analyst"	
Attributes on type level	
name	The name of the position type.
responsibility	Serves to describe the tasks and the responsibilities. Note that a more detailed description of responsibilities is possible by associating a position type with business processes or decision scenarios.
qualification	The qualification that is required for this position type. It includes the professional qualification and the formal qualification – e.g. a certain degree. It may also include specific experiences.
availability	Serves to express the availability of adequately qualified personnel on the job market.
staff	Serves to express whether this position type is a staff position.
needForOnTheJobTraining	Allows for expressing the need for further training that is required for this position type, e.g.: 0: no need; 1: little; 2: clear; 3: urgent.
averageSpan	This attribute serves to represent the average span of control across all instances of this Position – only if the Position has subordinated positions. If the Position does not have subordinated positions (or organisational units), this attribute must not be initialised.
corporateRelevance	Allows for expressing the relevance the position type has for an organisation's competitiveness. The assigned value represents the degree of relevance, e.g.: 0: no need; 1: could do without; 2: needed; 3: essential.
Relationships on type level	
represented_by	With PositionShare . A Position can be represented by zero

Analysis of Potential Language Concepts

(corresponds to represents for PositionShare)	to many PositionShare. Each PositionShare is assigned to exactly one Position.	
qualifies_for (corresponds to requires for Role or Committee)	With Role and Committee – allows for expressing that a Position qualifies for filling a Role or joining a Committee.	
supervises (corresponds to supervised_by for Position, Role, Committee, Board, or Organisation)	With Position, Role or Board. Multiplicity is zero to many on both sides.	
superior_of (corresponds to subordinatedTo for OrganisationalUnit Position, Role or Board)	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.	
functionalSuperior_of (corresponds to functionalsubordinated_to for OrganisationalUnit Position, Role or Board)	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.	
objectSuperior_of (corresponds to objectSubordinated_to for OrganisationalUnit Position, Role or Board)	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.	
disciplinarySuperior_of (corresponds to disciplinarySubordinated_to for OrganisationalUnit, Position, Role or Board)	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.	
responsible_for (corresponds to assigned_to for Task)	With Task. Allows for assigning task types, a position type is responsible for.	
Constraints		
C9	exclude staff positions from regular line of command	If the attribute staff is set to #true, it is not possible that the Position can be superior to any other OrganisationalUnit than the one, it is part of.
C7	No more superiors than maxLineOfCommand.	The sum of superiors (generic, functional, object, disciplinary) must not exceed the value of maxLineOfCommand specified for the corresponding Organisation.
C4	No joint use of generic and specific superiority	If a Position is superior to any other Position, Board or Role, there must be not further special superior relationship with that.
C6	no cyclic superior relationships	There must be no Position or Role or Board A that is superior of a further Position or Role or Board B while at the same time B is superior of A.
C12	No more than one time superior to units	A Position can be assigned to an OrganisationalUnit or an Organisation only once as superior.
C13	No more than one time superior to Position, Role or Board	A Position can be assigned to another Position, a Role or a Board only once as superior.

Table 6: Description of the meta type Position

PositionShare is an abstraction that allows for associating position types in a differentiated way to organisational units – or to the entire organisation. Instances of this type may be used in a redundant way. If, for example, the shares of a position were assigned to a set of organisational units and at the same time to that organisational unit those are part of, the feature values of the aggregate organisational unit would be redundant.

PositionShare	This concept is used to assign parts of the extension of a position type to organisational units. It allows for expressing features of these sets of positions. In those cases where all instances of a position type are assigned to one organisational unit only, the concept would not be required. However, for integrity reasons, it is used in these cases to.
Example Instantiations	
"Sales Representatives working for Division C", "Team Assistants working for Research&Development"	
Attributes with reference to instance level	
<<obtainable>> <<derivable>> openings	The number of currently open positions of this share of a type. The value can either be entered manually or obtained from a corresponding information system.
<<obtainable>> <<derivable>> averagePerformance	Allows for expressing how well the positions within this share of a type perform on average. The assigned value expresses the level of performance, e.g.: 0: critical; 1: satisfactory; 2: outstanding. The value could be calculated from data provided by an external source, e.g. an HRM system. Often, corresponding data will not be available. Then the value has to be assigned manually. In case a PositionShare has one instance only, this value serves to characterise this instance.
<<obtainable>> <<derivable>> fluctuation	The fluctuation of all positions of this share of a type within a certain period (e.g. a year). There are various formula for defining the fluctuation. The actual value could be provided by a corresponding information system, e.g. a HRM system.
<<obtainable>> <<derivable>> numberOfInstances	The actual number of positions of this share of a type. The value can be obtained from a corresponding information system.
<<obtainable>> <<derivable>> genderRatio	The percentage of females among the employees that fill positions of this share of a type.
<<obtainable>> <<derivable>> averageAge	The average age of all employees that fill positions of this share of a type – to be obtained from an external system.
<<obtainable>> <<derivable>> shareOfUnderPerformers	The share of positions that are assessed as not performing satisfactorily. This value can be aggregated from performance assessments of the corresponding instances. Alternatively, it can be based on estimations.
<<obtainable>> <<derivable>> averageCost	The average full cost per position within this share of a position type. This value should be provided by a corresponding information system, e.g. an accounting or an ERP system.

Analysis of Potential Language Concepts

Relationships on type level		
part_of (corresponds to includes for OrganisationalUnit)		With OrganisationalUnit. A PositionShare must be assigned to exactly one OrganisationalUnit.
represents (corresponds to represented_by for Position)		With Position. A PositionShare is assigned to exactly one Position.
qualifies_for (corresponds to requires for Role or Committee)		With Role and Committee – allows for expressing a PositionShare qualifies for filling a Role or a Committee. This relationship is needed only, if this statement cannot be made for the entire corresponding Position. Multiplicity is zero to many on both sides.
Represented_by (corresponds to represents for PrototypicalPosition)		With PrototypicalPosition. A PositionShare can be represented by zero to many PrototypicalPosition.
Relationships with reference to instance level		
<<obtainable>> filled_by		This relationship allows for establishing an association to a representation of the employee that fills this position, e.g. for including the name of this employee in the organisation model. For this purpose, one could use an interface object that contains the relevant attributes of the employee's representation (in most cases, the name will be sufficient). Note that referring to particular instances is certainly not the level of abstraction focussed by conceptual modelling.
Constraints		
C10	Not more than one PositionShare of a certain Position type can be assigned to a PermanentUnitOfWork	If a PositionShare of a certain Position is associated to a set of OrganisationalUnit and to the OrganisationalUnit, they are part of, then the sum of their numberOfInstances must not exceed the numberOfInstances of the OrganisationalUnit they are part of. Furthermore, they must not exceed the numberOfInstances of a corresponding PositionShare associated with Organisation.
C11	Aggregation of position shares must not exceed extension of corresponding position share in aggregate organisational unit.	If a PositionShare of a certain Position is associated to a set of OrganisationalUnit and to the OrganisationalUnit, they are part of, then the sum of their numberOfInstances must not exceed the numberOfInstances of the OrganisationalUnit they are part of. Furthermore, they must not exceed the numberOfInstances of a corresponding PositionShare associated with Organisation.

Table 7: Description of the meta type PositionShare

Note that PositionShare is rather an auxiliary abstraction than a core language concept. It is, however, an essential supplement to Position.

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

Role	<p>This concept is similar to Position. In some organisations a responsibility that is assigned to a position in other organisations may be assigned to a role. While many of the features defined for Position are applied for Role as well, it will often not be required to model roles in much detail. Different from Position, it is not regarded as necessary to supplement Role with a concept to express the share of role fillers assigned to an OrganisationalUnit: It is assumed that in the majority of cases there is no demand for such a concept.</p>
Example Instantiations	
"Quality Agent" , "Standardisation Delegate" , "Project Manager" , "Dean"	
Attributes on type level	
name	The name of the role type.
responsibility	Serves to describe the tasks and the responsibilities. Note that a more detailed description of responsibilities is possible by associating a role type with business processes or decision scenarios.
procedure	Description of the procedure that serves to fill a role of this type.
internal	Indicates whether a role of this type has to be filled with an internal employee.
period	The default time period for filling a role of this type.
qualification	The qualification that is required for this role type. It includes the professional qualification and the formal qualification – e.g. a certain degree. It may also include specific experiences. Further preconditions concerning qualification can be defined through the relationship <i>requires</i> (with Position).
corporateRelevance	Allows for expressing the relevance the role type has for an organisation's competitiveness. The assigned value represents the degree of relevance, e.g.: 0: no need; 1: could do without; 2: needed; 3: essential.
availability	Serves to express the availability of adequately qualified personnel within an organisation and/or on the job market.
Attributes with reference to instance level	
<<obtainable>> numberOfInstances	The actual number of roles of this type.
<<obtainable>> notFilled	The number of roles of this type that are currently not filled. The value can either be entered manually or obtained from a corresponding information system.
<<obtainable>> averagePerformance	Allows for expressing how well the roles of this type (i.e. the corresponding role fillers) perform on average. The assigned value expresses the degree of urgency, e.g.: 0: critical; 1: satisfactory; 2: outstanding. The value can be obtained from external sources – if available. Alternatively it can be assigned manually.
<<obtainable>>	The fluctuation of all roles of this type within a certain period (e.g. a year).

Analysis of Potential Language Concepts

fluctuation	There are various formula for defining the fluctuation. At best, it is possible to select one. The actual value could be provided by a corresponding information system, e.g. a HRM system.
needForOnTheJobTraining	Allows for expressing the need for further training that is required for this role type, e.g.: 0: no need; 1: little; 2: clear; 3: urgent.
<<obtainable>> genderRatio	The percentage of females among the employees that fill roles of this type.
<<obtainable>> averageAge	The average age of all employees that fill roles of this type – to be obtained from an external system.
<<obtainable>> shareOfUnderPerformers	The share of roles that are assessed as not performing satisfactorily. This value can be aggregated from performance assessments of the corresponding instances. Alternatively, it can be based on estimations.
<<obtainable>> averageTimePerMonth	The average time per month, an employee spends for filling a role of this type. This is an indicator of costs caused by this role type.
Relationships on type level	
part_of (corresponds to includes for OrganisationalUnit)	With OrganisationalUnit. A Role may be assigned to zero or more OrganisationalUnit. Different from Position, it is not possible to define shares of a role type, because it is assumed that in most cases this is not required. If this is regarded as a restriction, it is possible to differentiate the corresponding role types. If, e.g., one wants to assign the role type “Dean” to the various departments of a university, one would specify different role types instead, e.g. “Dean of the
requires (corresponds to qualifies_for for Position, PositionShare or PositionCategory)	With Position, PositionShare and PositionCategory. Allows for specifying the position types that are a required for filling roles of this type. Multiplicity is zero to many on both sides.
qualifies_for (corresponds to requires for Committee)	With Committee – allows for expressing that a Role qualifies for participating in a Committee. Multiplicity is zero to many on both sides.
supervises (corresponds to supervised_by for Position, Role, Committee, Board, or Organisation)	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.
superior_of (corresponds to subordinated_to for OrganisationalUnit Position, Role or Board)	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.
functionalSuperior_of (corresponds to functionalSubordinated_to for OrganisationalUnit, Position, Role or Board)	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.
objectSuperior_of (corresponds to objectSubordi-	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

nated_to for OrganisationalUnit, Position, Role or Board)		
disciplinarySuperior_of (corresponds to disciplinarySubordinatedTo for OrganisationalUnit, Position, Role or Board)	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.	
assigned_to (corresponds to assigns for OrganisationalUnit)	A Role can be assigned to zero or more OrganisationalUnit. An OrganisationalUnit can assign zero or more Role.	
Relationships with reference to instance level		
<<obtainable>> filled_by	This relationship allows for establishing an association to a representation of the employee that fills this role. For this purpose, one could use an interface object that contains the relevant attributes of the employee's representation (in most cases, the name will be sufficient). Note that referring to particular instances is certainly not the level of abstraction focussed by conceptual modelling.	
Constraints		
C7	No more superiors than maxLineOfCommand.	The sum of superiors (functional, object, disciplinary) must not exceed the value of maxLineOfCommand specified for the corresponding Organisation.
C4	No joint use of generic and specific superiority	If a Role is superior to any other Position, Board or Role, there must be not further special superior relationship with that.
C6	no cyclic superior relationships	There must be no Position or Role or Board A that is superior of a further Position or Role or Board B while at the same time B is superior of A.
C12	No more than one time superior to units	A Role can be assigned to an OrganisationalUnit or an Organisation only once as superior.
C13	No more than one time superior to Position, Role or Board	A Role can be assigned to another Position, a Role or a Board only once as superior.

Table 8: Description of the meta type Role

Committee	The relevance of committees depends on the type of organisation. In many cases, most of the features specified for this meta type will not be used. Usually, a committee type has one instance only. Like organisational units, committees are supposed to be singletons.
Example Instantiations	
"Quality Circle", "Diversity Management Committee", "Steering Committee"	
Attributes on type level	
name	The name of the committee type.
mission	Serves to describe the mission and the responsibilities.

Analysis of Potential Language Concepts

internal	Specifies whether all members of committees of this type have to be internal employees.
corporateRelevance	Allows for expressing the relevance the committee type has for an organisation's competitiveness. The assigned value represents the degree of relevance, e.g.: 0: no need; 1: could do without; 2: needed; 3: essential.
meetingsPerYear	The regular number of meetings per year. It is assumed that this number is part of a committee's constitution and does not vary to a noteworthy degree from year to year.
Attributes with reference to instance level	
strengths	The strengths of a committee with respect to the organisational goals. In a simple case, this attribute can be described by a string. Alternatively it is conceivable to use a more elaborate structure.
weaknesses	The strengths of a committee with respect to the organisational goals. In a simple case, this attribute can be described by a string. Alternatively it is conceivable to use a more elaborate structure.
<<obtainable>> genderRatio	The percentage of females among the members
<<obtainable>> numberOfPositions	The actual number of positions within the committee.
<<obtainable>> performance	The value of a performance indicator on an ordinary scale, e.g.: e.g.: 0: critical; 1: satisfactory; 2: outstanding.
averageDuration	The average duration of a session (usually in hours).
<<obtainable>> averageAge	The average age of all employees of this organisational unit – also to be obtained from an external system.
Relationships on type level	
requires	With Role, Position, PositionShare or PositionCategory. This relationship allows for specifying the role and/or position types that are required for becoming a committee member.
supervised_by (corresponds to supervises for Position, Role or Board)	With Position, Role or Board. Multiplicity is zero to many on both sides.
subordinated_to (corresponds to superior_of for Position, Role or Board)	With Position, Role or Board. This relationship is used in cases where no particular definition of superiority exists or matters. Multiplicity is zero to many on both sides. The number of superiors must not exceed the value of maxLineOfCommand within the corresponding Organisation.
functionalSubordinated_to (corresponds to functionalSuperi- or_of for Position, Role or Board)	With Position, Role or Board. Multiplicity is zero to many on both sides.
objectSubordinated_to (corresponds to objectSuperi- or_of for Position, Role or Board)	With Position, Role or Board. Multiplicity is zero to many on both sides.

disciplinarySubordinated_to (corresponds to disciplinarySuperior_of for Position, Role or Board)	With Position, Role or Board. Multiplicity is zero to many on both sides.
hosted_by (corresponds to hosted for OrganisationalUnit)	With OrganisationalUnit. A type of organisational unit can host zero or more (0,*) committee types. On the other hand, a committee unit type can be hosted by one (default) or more than one organisational unit type.

Table 9: Description of the meta type Committee

Board	Different from other organisational units, a board has executive authority – similar to certain positions. Usually, there will be only one instance of a board type.
Example Instantiations	
"Board of Directors", "Executive Board", "Supervisory Board"	
Attributes on type level	
name	Allows for assigning a type name.
mission	Serves to describe the mission and the responsibilities.
corporateRelevance	Allows for expressing the relevance the board type has for an organisation's competitiveness. The assigned value represents the degree of relevance, e.g.: 0: no need; 1: could do without; 2: needed; 3: essential.
internal	Specifies whether all members of boards of this type have to be internal employees.
Attributes on instance level	
<<obtainable>> genderRatio	The percentage of females among the members
<<obtainable>> numberOfMembers	The actual number of members
performance	The value of a performance indicator on an ordinary scale, e.g.: 0: critical; 1: satisfactory; 2: outstanding.
<<obtainable>> averageAge	The average age of all employees of the board – can be obtained from an external system.
Relationships on type level	
qualifiesFor (corresponds to requires for Committee)	With Committee – allows for expressing that membership in the Board qualifies for participating in a Committee. Multiplicity is zero to many on both sides.
supervises (corresponds to supervisedBy for Position, Role, Committee, Board,	With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.

Analysis of Potential Language Concepts

or Organisation)		
superior_of (corresponds to subordinated_to for OrganisationalUnit, Position, Role or Board)		With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.
functionalSuperior_of (corresponds to functionalSubordinated_to for OrganisationalUnit, Position, Role or Board)		With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.
objectSuperior_of (corresponds to objectSubordinated_to for OrganisationalUnit, Position, Role or Board)		With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.
disciplinarySuperior_of (corresponds to disciplinarySubordinated_to for OrganisationalUnit, Position, Role or Board)		With OrganisationalUnit, Position, Role, Organisation, Committee or Board. Multiplicity is zero to many on both sides.
Constraints		
C7	No more superiors than maxLineOfCommand.	The sum of superiors (functional, object, disciplinary) must not exceed the value of maxLineOfCommand specified for the corresponding Organisation.
C4	No joint use of generic and specific superiority	If a Board is superior to any other Position, Board or Role, there must be not further special superior relationship with that.
C6	no cyclic superior relationships	There must be no Position or Role or Board A that is superior of a further Position or Role or Board B while at the same time B is superior of A.
C12	No more than one time superior to units	A Board can be assigned to an OrganisationalUnit or an Organisation only once as superior.
C13	No more than one time superior to Position, Role or Board	A Board can be assigned to another Position, a Role or a Board only once as superior.

Table 10: Description of the meta type Board

Finally, there are a few auxiliary concepts that are not perceived as original concepts, but rather serve to express particular viewpoints/abstractions of existing concepts. PrototypicalPosition can be used to represent single prototypical positions, but also an entire type of prototypical positions. In case the features serve to describe particular instances of instances, they are characterised as <<intrinsic>>.

PrototypicalPosition	An auxiliary concept that serves to represent a position of a certain type that has with prototypical features. It allows for expressing constraints that require accounting for instances (e.g.: “The entire process must be attended by the same sales assistant.”). Furthermore, it can be used to create instances for simulation purposes.
Example Instantiations	
“Sales Assistant that is in charge of a particular instance of an order management process”; “One of two insurance clerks that are involved in an application process.”	
Attributes on type level	
<<simulation>> numberOfInstances	Normally, the maximum number of instances of a PrototypicalPosition must not exceed the number of instances of the Position or the SplitPosition it is associated to. However, this feature serves simulation purposes only. Therefore, any value can be assigned that might be useful for running a simulation.
<<simulation>> timeAverageAvailability	Serves to express the availability of positions during a certain period, e.g. a week. For this purpose, corresponding distribution functions can be used.
<<simulation>> averagePerformance	This attribute allows to define a distribution function for characterising how the performance indicator (e.g.: 0: critical; 1: satisfactory; 2: outstanding) is distributed across the population of prototypical instances that is used for a simulation.
Attributes on instance level	
<<intrinsic>> identity	A string that provides a unique identifier for an instance of PrototypicalPosition.
<<intrinsic>> <<simulation>> timeAvailability	The availability of a particular instance of PrototypicalPosition during a certain period – could be expressed by a distribution function.
<<intrinsic>> <<simulation>> performance	Serves to assign a value of a performance indicator to a particular instance of PrototypicalPosition.
Relationships on type level	
represents (corresponds to represented_by for Position or PositionsShare)	With Position or PositionShare. This relationship can be further characterised by a description of the function it is related to, e.g.: “database design”.

Table 11: Description of the auxiliary meta type PrototypicalPosition

Categories of concepts provide the language user with an instrument for defining his own abstractions that serve particular analysis purposes. It is conceivable to define categories for all essential language concepts. However, the current version of the MEMO OrgML is restricted to three categories, which allow for grouping organisations, e.g. to industries, and position types. This restriction is based on the assumption – which is still to be evaluated –

Analysis of Potential Language Concepts

that other categories are usually not required. A category is regarded as an auxiliary language concept. It is certainly not a clear meta type, because it is difficult to imagine that an instance of the concept is a type. However, it is conceivable to think of a category type, e.g. “Managerial Positions” that can be “instantiated” into concrete categories for a particular organisation. In an ideal case, a category would be defined as a view on existing data. However, it is possible that one wants to do without representing the elements of a category. Therefore, all features of category concepts are marked as “derivable”, which means that they should be aggregated from existing values, if these are available. Figure 11 shows two examples, a category of organisational units and a category of positions.

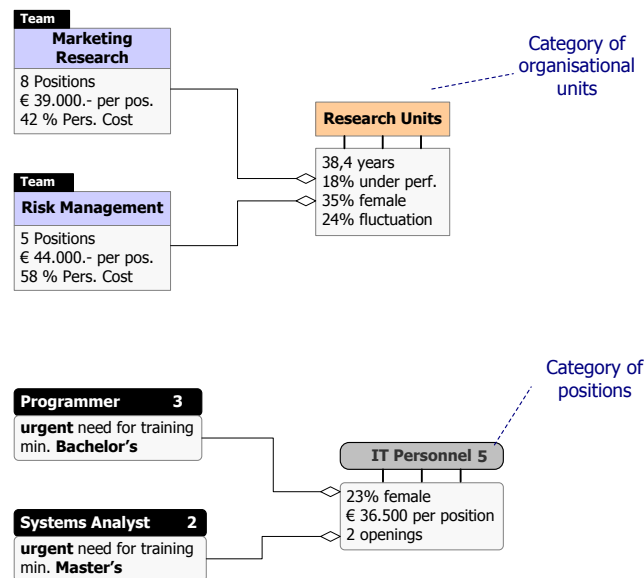


Figure 11: Example of categories

PositionCategory	A category of position types is defined with respect to a certain perspective that may, e.g., reflect a particular kind of analysis.
Example Instantiations	
“Core Units”, “Visible Units”, “Critical Units”	
Attributes	
name	Name of the category.
purpose	Serves to describe the purpose, the category is supposed to serve.
<<derivable>> numberOfElements	Number of Position types, the category is comprised of.
<<derivable>> numberOfInstances	The total number of instances of all included position types.
<<derivable>>	The number of currently open positions within this category. The value can

The MEMO Organisation Modelling Language (1): Focus on Organisational Structure

openings	either be entered manually or obtained from a corresponding information system.
<<derivable>> averageCost	The average full cost per position in this category.
<<derivable>> genderRatio	The percentage of females among the employees of the corresponding positions.
<<derivable>> averagePerformance	The value of a performance indicator on an ordinary scale, e.g.: 0: critical; 1: satisfactory; 2: outstanding. The indicator serves to express the average performance of the corresponding positions.
<<derivable>> averageAge	The average age of all employees of the corresponding positions.
Relationships	
qualifies_for (corresponds to requires for Committee)	With Committee – allows for expressing that any of the included Position qualifies for participating in this Committee. Multiplicity is zero to many on both sides.
includes (corresponds to part_of for Position)	With Position. Allows for assigning the corresponding position types (zero to many).

Table 12: Description of the type PositionCategory

OrgUnitCategory	A category of organisational unit types is defined with respect to a certain perspective that may, e.g., reflect a particular kind of analysis.
Example Instantiations	
"Core Units", "Visible Units", "Critical Units"	
Attributes on type level	
name	Name of the category.
purpose	Serves to describe the purpose, the category is supposed to serve.
Attributes with reference to the instance level	
<<derivable>> numberOfPositions	The total number of positions of all includes organisational units.
<<derivable>> numberOfEmployees	The total number of employees of all includes organisational units.
<<derivable>> aveNumberOfPositions	The average number of positions of the corresponding organisational units.
<<derivable>> aveNumberOfEmployees	The average number of employees of the corresponding organisational units.
<<derivable>> genderRatio	The percentage of females among the employees of the corresponding organisational units.

Analysis of Potential Language Concepts

<<derivable>> averagePerformance	The value of a performance indicator on an ordinary scale, e.g.: 0: critical; 1: satisfactory; 2: outstanding. The indicator serves to express the average performance of the corresponding organisational units.
<<derivable>> averageAge	The average age of all employees of the corresponding organisational units.
Relationships on type level	
includes (corresponds to <code>part_of</code> for <code>OrganisationalUnit</code>)	With <code>OrganisationalUnit</code> . Allows for assigning the corresponding organisational units.

Table 13: Description of the meta type `OrgUnitCategory`

OrganisationCategory	A category of organisation types can be used to group organisations of a certain kind.
Example Instantiations	
"Industrial Enterprise", "Software Vendor", "PublicAgency", "University"	
Attributes	
name	Name of the category.
purpose	Serves to describe the purpose, the category is supposed to serve.
Relationships	
qualifies_for (corresponds to <code>requires</code> for <code>Committee</code>)	With <code>Committee</code> – allows for expressing that all positions assigned to this <code>PositionCategory</code> qualify for participating in the associated <code>Committee</code> . A <code>PositionCategory</code> may qualify for zero to many <code>Committee</code> . On the other hand, zero to many <code>PositionCategory</code> may qualify for a <code>Committee</code> .
includes (corresponds to <code>part_of</code> for <code>Organisation</code>)	With <code>Organisation</code> . Allows for assigning the corresponding organisations. An <code>Organisation</code> can be assigned to zero or more <code>OrganisationCategory</code> .

Table 14: Description of the type `OrganisationCategory`

Interaction	Two <code>PermanentUnitOfWork</code> , i.e. organisational units and organisations, may interact/communicate via various media. This meta type serves to represent types of interaction which are aggregates of particular interactions. The attributes relate to different media and to average numbers of contact and duration within a certain reference period, e.g. a month. Hence, the attributes are not type level attributes in a strict sense. An instance of <code>Interaction</code> cannot be further instantiated.
Example Instantiations	

“Interaction between Marketing and Manufacturing”, “Interaction between Market Research and Sales”	
Attributes on type level	
<<derivable>> contacts	The total number of interaction contacts within the reference period. This number can be calculated from the contact numbers assigned to the various media – if those are available.
<<derivable>> duration	The average duration of an interaction within the reference period. This number can be calculated from the average durations assigned to the various media – if those are available.
<<derivable>> intensity	Serves to indicate the intensity of an interaction between two PermanentUnitOfWork. A possible specification could be: intensity (o1, o2) := (noEmpl(o1) + noEmpl(o2))/contacts.
phone	The average number of phone contacts within the reference period.
phoneDuration	The average duration of phone interactions within the reference period.
e-mail	The average number of e-mail contacts within the reference period. This time would be calculated from the times required to prepare and read a message.
e-Duration	The average duration of e-mail interactions within the reference period.
fax	The average number of fax contacts within the reference period.
faxDuration	The average duration of fax interactions within the reference period. This time would be calculated from the times required to prepare and read a fax.
face	The average number of face to face contacts within the reference period.
faceDuration	The average duration of face to face interactions within the reference period.
refPeriod	A String that describes a reference period such as “Month”, “Year” etc. or “2012-May” or “2012”. To facilitate machine interpretation a more elaborate type may be advisable.
Relationships on type level	
connects	With PermanentUnitOfWork. Serves to assign the two units that form the corresponding interaction (type).

Table 15: Description of the meta type OrganisationCategory

Further auxiliary concepts comprise types for defining the legal form of an organisation, local organisational unit types, the availability of prototypical positions or the qualification required for a position. A detailed description of these types is provided with the meta model below.

4.3 Meta Model

The graphical illustration of modelling concepts and their natural language description may be regarded as sufficient – if not as too comprehensive – by many prospective users. Those may skip this section. The meta model is intended to provide a more precise specification of the language's syntax and semantics. For this purpose, it is required to specify those parts that remain vague in the above description, e.g. the types of the features. Furthermore, there is need to analyse commonalities of language concepts in order to define abstractions such as generalisations. This may include the introduction of additional meta types. Unfortunately, the semantically overloaded terms commonly used for describing organisations, do not remain without effect: They compromise the meta model's conceptual clarity and its aesthetics. A few, not to say most of the concepts of the meta model are not meta types in the sense that they would allow for instantiating their instances. However, the core concepts of the language, those that reflect the technical language of organisers, can be regarded as meta types. `OrganisationalUnit` for instance can be instantiated to a particular type of organisational unit that will usually be a singleton. `Position` qualifies as a meta type without any doubts. Its instances are position types, which could be instantiated into many particular positions. Other concepts, such as `LegalForm` or `LocalUnitType` are clearly no meta types. They are part of the meta model – and represented as meta types, because they are required for defining the semantics of the core concepts of the language.

The varying level of abstraction emphasised by the concepts in the meta model does not only affect their interpretation as meta types or types. Furthermore, it has an impact on the semantics of multiplicities. As a default, in a meta model, the multiplicities of associations apply to the number of corresponding types. Take, for instance, the association `qualifies_for` between `Position` and `Role`. A multiplicity of `0,*` with `Position` indicates that zero to many position *types* qualify for corresponding role *types*. The type level multiplicity does not, however, restrict the number of corresponding instances. To give a further example: Exactly one position type is involved in the association between `Position` (represented by its supertype `PotentialSuperior`) and `OrganisationalUnit` (represented by the supertype `UnitOfWork`). That does not exclude that more than one position of this type act as superior. To express a specific multiplicity on this level in a model of an organisation structure, the concept `OrgSuperior` allows for specifying cardinalities. They apply to instances of concrete subtypes of `PotentialSuperior`. The multiplicity of `1,1` with `UnitOfWork` applies to the instance level as well because instances of corresponding subtypes are singletons.

The meta types in the model correspond to the meta types described above. To contribute to the meta model's flexibility, generalisation/specialisation relationships have been introduced. For the same reason, some attributes are specified with customised types. Customised types are printed in boldface. The MEMO meta modelling language does not allow for differentiating type level features and those which refer to the instance level. However, it allows for

specifying that a feature can be obtained from an external source (<<obtainable>>) or that it can be derived from another part of a model (<<derived>>). Since `OrganisationalUnit`, `Organisation` and `Committee` share the attributes `averageAge` and `genderRatio`, it may seem that these common attributes should be placed in the common supertype `UnitOfWork`. However, while both attributes can be obtained from external sources in all cases, they are derivable only for `OrganisationalUnit` and `Organisation`. Therefore, the two attributes are specified separately for `Committee` and `PermanentUnitOfWork`, the generalisation of `OrganisationalUnit` and `Organisation`. The model contains a number of features that are related to performance issues, e.g. `averagePerformance` or `performance`. They could be used for specific control purposes, e.g. by comparing them against benchmark values. In addition to that, they could be part of corresponding indicators. To contribute to a more consistent design of indicators and to counter dysfunctional effects caused by focussing on particular indicators only, it is a promising approach to create models of indicators systems as a separate abstraction. Each indicator type associated to other indicator types in order to represent the effect its instances may have on other indicators. It can also be associated to corresponding goals to enrich it with context. In addition to that, an indicator type can be assigned to a reference object such as an organisational unit, a business process type or a product type. A first version of a MEMO modelling language for designing indicator systems is presented in Frank et al. 2008). Indicators that apply to organisational units could include aspects such as revenues per employee, cost per employee but also share of graduates etc. In addition to features included in the meta model in Figure 12, an indicator system would include corresponding benchmarks and relationships between indicators. For instance: Increasing the share of graduates (which could be regarded as an indicator for workforce qualification) will usually have a negative impact on average cost per position. Note that the integration of organisational models with corresponding indicator systems is not further regarded in this report. It will, however, be subject of future extensions.



Figure 12: Meta model of concepts to model organisation structures

The meta model rendered in Figure 12 is supplemented by a set of constraints. Due to space limitations, they are represented by their identifiers only. In the following section these constraints are specified in OCL. Note that the syntax of the constraints has not yet been checked with a tool. Furthermore, the semantics relies on the interpretation of the specification in (OMG 2006), which does not seem to be complete. A natural language description of the constraints can be found in the description of the concepts above (Table 4 – Table 14). Note that the meta model includes additional abstractions. Therefore some constraints may include concepts that are not included in the above description.

C1 *context* **OrganisationalUnit**
def:
 let ownLevel : self.characterizedBy.level
inv:
 self.composedOf->forAll (o |
 o.characterizedBy.level < self.ownLevel)

C2 *context* **OrganisationalUnit**
def:
 let ownDim : self.orgDimension
inv:
 self.ownDim = self.organisation.orgDimension or
 self.organisation.orgDimension = #tensor or
 (self.organisation.orgDimension = #matrix and
 self.ownDim <> #tensor)

C3 *context* **OrganisationalUnit**
def:
 let allIncluded: self.composedOf->union(self.composedOf->collect(ou | ou.allIncluded))
inv:
 not self.allIncluded->includes (self)

C4 *context* **PotentialSuperior**
def:
 let allGeneric: self.acts->select(mode = #generic)
 let allGenericAss: allGeneric.aims
 let allNonGeneric: self.acts->select(mode <> #generic)
 let allNonGenericAss: allNonGeneric.aims
inv:
 allGenericAss->forAll (i | allNonGenericAss->excludes (i) = true)

C5 *context* **PotentialSuperior**
def:
 allGeneric: self.actsAs->select(mode = #generic)
 allGenericAss: allGeneric->collect (p | p.aimedAt)
 allNonGeneric: self.actsAs->select(mode <> #generic)
 allNonGenericAss: allNonGeneric-> collect (p | p.aimedAt)
inv:
 allGenericAss->forAll (i | allNonGenericAss->excludes (i) = true)

C6 *context* **PotentialSuperior**
def:
 let allDirectSubordinates: self.actsAs->collect (s | s.aimedAt)
 let allSubordinates: self.allDirectSubordinates->union(self.allDirectSubordinates->collect(ps | ps.allSubordinates))
inv:
 not self.allSubordinates->includes(self)

Analysis of Potential Language Concepts

C7	<i>context OrganisationalUnit</i> inv: self->orSuperiors->size <= self.organisation.maxLineOfCommand
C8	<i>context OrganisationalUnit</i> inv: self.staff=true implies not (self.composed_of->exists(p p.staff=false))
C9	<i>context Position</i> inv: self.staff=true implies not (self.acts.aims->exists(o o.staff=false))
C10	<i>context PositionShare</i> def: let allUnits: self.permanentUnitOfWork->collect (o o.positionShare.position = self.position) inv: allUnits->size() < 2
C11	<i>context PositionShare</i> def: let unit: self.permanentUnitOfWork let selPos: self.position let subShares: unit.composed_of->collect (p p.positionShares) let relSubShares: subShares->collect (p p.position = selPos) let numOfInst: 0 let relSubShares->forAll (ps numOfInst = numOfInst + ps.numberOfInstances) inv: numOfInst <= self.numberOfInstances
C12	<i>context PotentialSuperior</i> def: let allDirectSubordinates: self.actsAs->collect (s s.aimedAt) inv: allDirectSubordinates->size() = allDirectSubordinates->asSet()->size()
C13	<i>context PotentialSuperior</i> def: let allDirectSubordinates: self.acts->collect (s s.aimedAt) inv: allDirectSubordinates->size() = allDirectSubordinates->asSet()->size()
C14	<i>context Interaction</i> def: let os: self.connects.asOrderedSet() inv: os.first() <> os.last()

Figure 13: Constraints

The specification of auxiliary entity types is shown in Table 16. Note that they are represented in a different notation than the meta model because they do not represent meta types, but types. Some of these types are in a preliminary or simplified state. With more information about particular use cases available, they can easily be refined without affecting the meta model. The specification of the type *TimeAvailability*, which is used within *PrototypicalPosition* depends largely on the simulation model it is used for. Therefore, its specification serves as a

simplified example only. Some types could be further refined by the introduction of specific types for defining their attributes. For instance: The attribute field within Qualification refers to the field of education or expertise. A corresponding type such as “Field” could make sure that the instances have more semantics than a plain string. Alternatively, a set of reference initialisations could be provided to contribute to a coherent and consistent use of the attribute. The more specific types are supplemented by the generic types Money, Duration and TimeUnit.

Specification	Comment				
<table border="1"> <tr> <td>Duration</td> </tr> <tr> <td>unit: TimeUnit</td> </tr> <tr> <td>dur: Float</td> </tr> </table>	Duration	unit: TimeUnit	dur: Float	Duration allows to define a time period using a selected unit of time.	
Duration					
unit: TimeUnit					
dur: Float					
<table border="1"> <tr> <td>TimeUnit</td> </tr> <tr> <td>unit: {#second, #minute, #hour ...}</td> </tr> </table>	TimeUnit	unit: {#second, #minute, #hour ...}	TimeUnit allows to define the relevant unit of time.		
TimeUnit					
unit: {#second, #minute, #hour ...}					
<table border="1"> <tr> <td>Money</td> </tr> <tr> <td>currency: String</td> </tr> <tr> <td>amount: Float</td> </tr> </table>	Money	currency: String	amount: Float	Money serves to specify financial amounts together with the respective currency.	
Money					
currency: String					
amount: Float					
<table border="1"> <tr> <td>Affirmation</td> </tr> <tr> <td>level: {#high, #medium, #low}</td> </tr> </table>	Affirmation	level: {#high, #medium, #low}	Affirmation serves expressing assessments.		
Affirmation					
level: {#high, #medium, #low}					
<table border="1"> <tr> <td>Availability</td> </tr> <tr> <td>description: String</td> </tr> <tr> <td>level: Level</td> </tr> </table>	Availability	description: String	level: Level	Availability is used to specify the availability of a resource.	
Availability					
description: String					
level: Level					
<table border="1"> <tr> <td>Fluctuation</td> </tr> <tr> <td>description: String</td> </tr> <tr> <td>numberOfMonths: Integer</td> </tr> <tr> <td>percentage: Real</td> </tr> </table>	Fluctuation	description: String	numberOfMonths: Integer	percentage: Real	Fluctuation allows for specifying features that are characteristic for fluctuation: average number of months an employee stays in the organisation, and the percentage of the employees that leave within a certain time period, e.g. one year.
Fluctuation					
description: String					
numberOfMonths: Integer					
percentage: Real					
<table border="1"> <tr> <td>LeadAspect</td> </tr> <tr> <td>aspect: {#generic, #function, #object}</td> </tr> <tr> <td>description: String</td> </tr> </table>	LeadAspect	aspect: {#generic, #function, #object}	description: String	LeadAspect serves the specification of the primary aspect of a superior relationship. E.g., if aspect is set to #object, this would indicate that the corresponding superior is superior with respect to an object such as a product or product group.	
LeadAspect					
aspect: {#generic, #function, #object}					
description: String					
<table border="1"> <tr> <td>Level</td> </tr> <tr> <td>level: {#critical, #satisfactory, #high}</td> </tr> </table>	Level	level: {#critical, #satisfactory, #high}	Level serves expressing quality levels.		
Level					
level: {#critical, #satisfactory, #high}					
<table border="1"> <tr> <td>Mission</td> </tr> <tr> <td>description: String</td> </tr> </table>	Mission	description: String	Mission serves describing the mission of an organisational unit, a role etc. Currently, the specification is not differentiated. This		
Mission					
description: String					

	can, however, be changes any time.		
<table border="1"> <tr> <td>OrgDimension</td> </tr> <tr> <td>dim: {#object, #function, #matrix, #tensor}</td> </tr> </table>	OrgDimension	dim: {#object, #function, #matrix, #tensor}	OrgDimension can be used to describe the primary principle of division of labour within an organisation.
OrgDimension			
dim: {#object, #function, #matrix, #tensor}			
<table border="1"> <tr> <td>Performance</td> </tr> <tr> <td>strengths: String weaknesses: String potential: String perfLevel: Level</td> </tr> </table>	Performance	strengths: String weaknesses: String potential: String perfLevel: Level	Performance serves the differentiated specification of the performance related to an organisational unit, a role etc.
Performance			
strengths: String weaknesses: String potential: String perfLevel: Level			
<table border="1"> <tr> <td>Qualification</td> </tr> <tr> <td>description: String field: String level: String experience: {#no, #little, #satisfactory, #outstanding}</td> </tr> </table>	Qualification	description: String field: String level: String experience: {#no, #little, #satisfactory, #outstanding}	Qualification allows defining the actual or demanded qualification of a position, role, etc. In the case of business process modelling, it is used to describe qualification relevant for performing processes.
Qualification			
description: String field: String level: String experience: {#no, #little, #satisfactory, #outstanding}			
<table border="1"> <tr> <td>RelAspect</td> </tr> <tr> <td>aspect: {#customer, #competitor, #partner}</td> </tr> </table>	RelAspect	aspect: {#customer, #competitor, #partner}	RelAspect serves the classification of relationships with external organisations.
RelAspect			
aspect: {#customer, #competitor, #partner}			
<table border="1"> <tr> <td>Responsibility</td> </tr> <tr> <td>description: String level: {#little, #substantial, #outstanding}</td> </tr> </table>	Responsibility	description: String level: {#little, #substantial, #outstanding}	Responsibility allows for specifying the responsibility related to an organisational unit, a role etc.
Responsibility			
description: String level: {#little, #substantial, #outstanding}			
<table border="1"> <tr> <td>TimeAvailability</td> </tr> <tr> <td>dayDist [0..24]: Real weekDist [0..7]: Real monthDist [0..12]: Real yearDist [0..12]: Real</td> </tr> </table>	TimeAvailability	dayDist [0..24]: Real weekDist [0..7]: Real monthDist [0..12]: Real yearDist [0..12]: Real	TimeAvailability is a preliminary concept that allows for describing the availability of prototypical instances for the purpose of simulation.
TimeAvailability			
dayDist [0..24]: Real weekDist [0..7]: Real monthDist [0..12]: Real yearDist [0..12]: Real			

Table 16: Auxiliary Types

4.4 Excursus: Support for Managing Multi-Lingual Model Systems

Sometimes, it may be required to manage two or more country-specific versions of the same model. In the easiest case, the model semantics does not vary, while the designators do. The resulting demand for supporting the management of multi-lingual model systems is not a direct requirement for a modelling language. It is a requirement for a corresponding modelling tool only. Nevertheless it makes sense to account for this requirement already at the time of designing a language, because the language design may affect the effort it takes to implement a tool with support for multi-lingual designators. There is a straightforward approach to satisfy this request: overloading the semantics of attributes that may store country-specific designators. In the case of a single-language system, the attribute would serve to store a string that represents actual designator (as it is intended on the level of the language specification). To support a multi-lingual system, the corresponding string would serve as a unique

key to access a repository of multi-lingual designators. This would allow for retrieving the entry that corresponds to the chosen language.

4.5 Graphical Notation

Regarded as “syntactical sugar” by some, the graphical notation of a modelling language may have a remarkable impact on the usability and acceptance, hence: on the productivity enabled by a language. The graphical notation has been developed with respect to the guidelines in Frank 2011c). A professional graphic designer revised the original version that was used to illustrate language concepts (e.g. in Figure 9) and created two variants (“glossy” and “matt”) to account for different aesthetic preferences of prospective users. The elements of the graphical notation are listed in Table 17. In addition to that they are subsequently illustrated in example diagrams.

While regarded as irrelevant “syntactic sugar” by some, the graphical notation featured by a DSML will often be of pivotal relevance for its acceptance and usability. The graphical symbols that form the concrete syntax of the OrgML were created by a graphic artist. They are aimed at both, promoting readability and appealing, aesthetic diagrams. Currently, the graphical notation exists in two variants, “matt” and “glossy”. There are a few core symbols. They serve to represent core concepts that have an identity on their own, like organisational units, roles etc. Additional symbols can be attached to core symbols to allow for further differentiating the corresponding concepts. The wide and unforeseeable range of possible organisational unit types does not allow for providing a comprehensive set of symbols. To support a visual differentiation of organisational unit types nevertheless, the notation includes two symbols. One symbol serves to mark larger organisational units such as departments, head departments etc., while the second symbol is supposed to mark smaller units such as teams. Note that these symbols do not represent a specific concept. They only serve to allow for some degree of visual discrimination. This is also the case for the symbols that are provided for distinguishing different types of positions. As a consequence, the set of these symbols may be modified or extended independent from the specification of the language’s semantics and abstract syntax.

There are three performance symbols (“critical”, “satisfactory”, “high”) that serve to assign performance indicators to organisational entities. Note that a finer differentiation is possible. Especially with the use of a modelling tool, it may be possible to calculate the particular graphical indicator during run-time. Finally, the notation offers a set of symbols to characterise association types.

Table 17 shows a complete dictionary of all notation elements in both variants.

OrgStructure Symbols			
	matt	glossy	Description
Organisational Units and Committees			<p>Generic organisational entity: This symbol is the basic element for representing organisational units, positions, boards and committees. It needs to be supplemented with an additional symbol represents a certain kind of organisational entity (see below).</p> <p>Colours can be used to distinguish different kinds of entities.</p>
			Larger organisational unit that typically includes further organisational units, e.g. a department.
			Smaller organisational unit that typically does not include further organisational units, e.g. team. Note the both current symbols to represent organisational units can be reserved for specific "local types".
		Staff unit	
		Board, e.g. board of directors	

			Committee – discriminated against organisational units by a different shape (ellipse).
Positions and Roles			Generic position. The number in the specific position symbol is optional and serves to represent the number of instances that exist for the corresponding position type. This number can be assigned to any position type.
			Position that requires a technical background, e.g. technician, engineer ...
			Position for sales persons.
			Position for purchasing, procurement
			Programmer
			Management position
			Staff position
			Role – currently, there is only one role symbol. The ellipse is available in various colours.
	Perfo		

			the above elements.
			Satisfactory performance
			Critical performance
Textbox	 open.: 1 ave.: € 78.000 Total: € 39.000	 open.: 1 ave.: € 78.000 total: € 39.000	Textbox, can be attached to all elements shown above.

Table 17: Symbols for Representing Organisational Units

Organisational charts include two major kinds of relationships between organisational units: aggregation and superior relationships. Superior relationships are differentiated into various forms in order to allow for an elaborate representation of management aspects. The following table serves to illustrate the graphical notation. It does not cover all possible superior relationships.

Aggregation (is part of)		
		Can be used to associate two organisational units. The “+” symbol is placed next to the aggregate unit, i.e. the unit that includes the other one. Hence, the association is interpreted as “is part of”, read in the direction towards the “+” symbol.
Superior, generic		
		A position or a board can be marked as superior of a position, board or organisational unit.

	<p>“Generic” refers to the fact that it is not further specified whether the superior relationship is restricted to certain aspects.</p>
<p>Superior, functional</p>	
<p style="text-align: center;">-< functionally superior of</p>	<p>Superior with respect to a functional aspect, e.g. “procurement”.</p>
<p>Superior, object-oriented</p>	
<p style="text-align: center;">-< oo superior of</p>	<p>Superior with respect to certain objects, e.g. a certain product or product category.</p>
<p>Substitutes for</p>	
	<p>This relationship type serves to indicate that a position type serves as a substitute for another position type. The substitution symbol is placed at that end of the edge that is attached to the symbol that represents the substitutable position type.</p>
<p>Qualifies for</p>	
	<p>Sometimes, an organisational role requires that only employees which hold certain positions may</p>

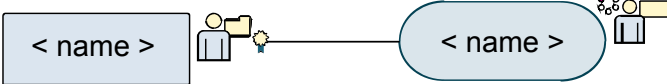
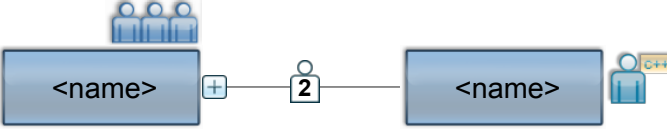
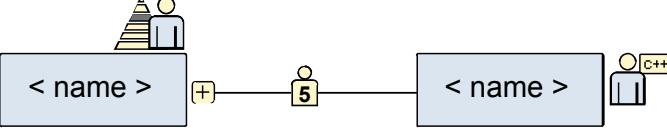
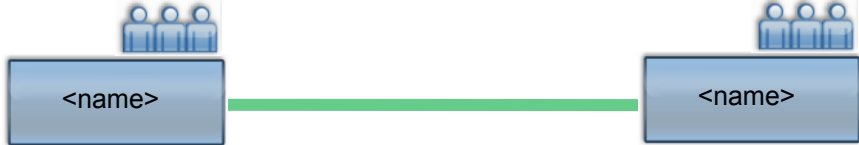

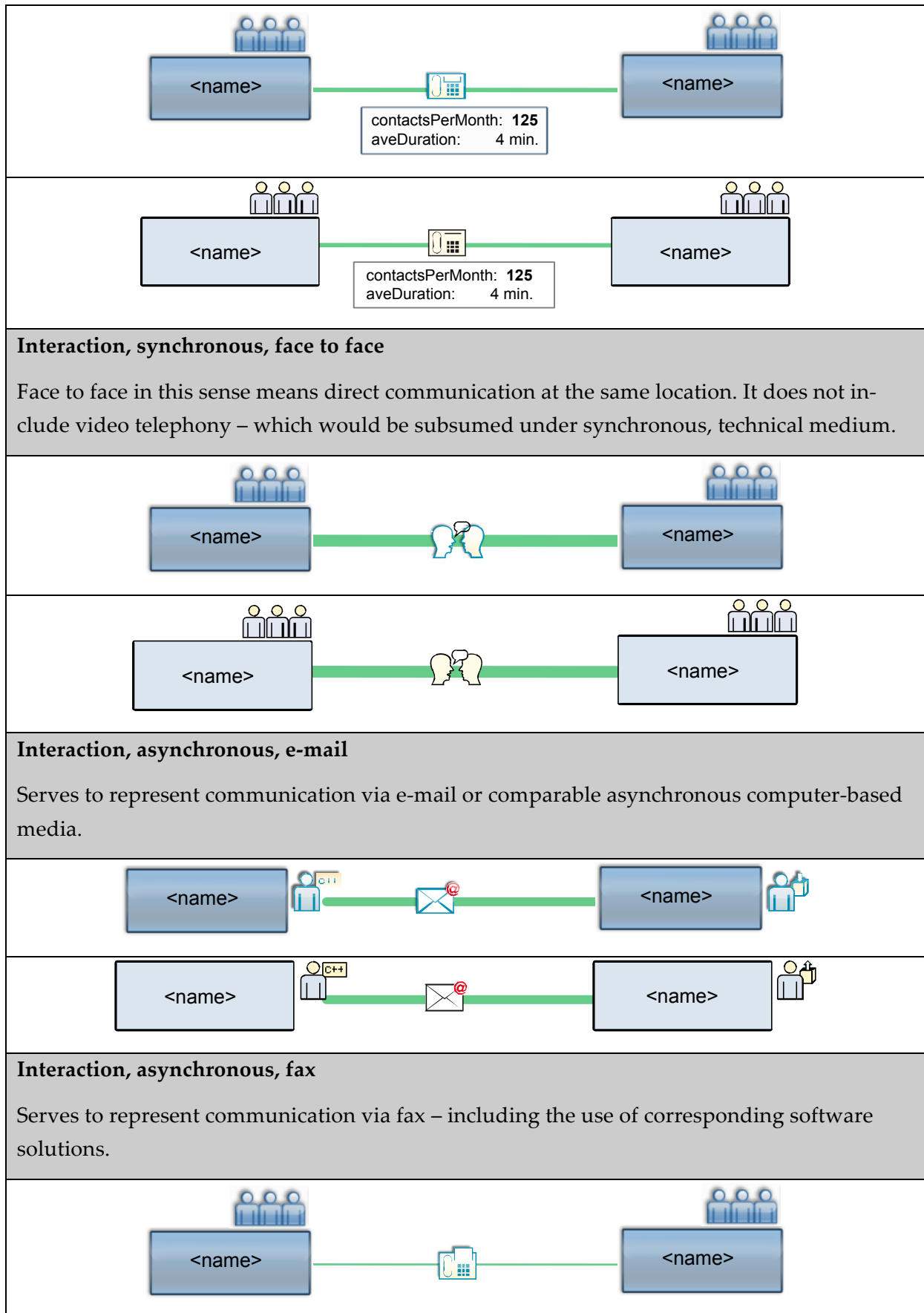
	<p>fill it. This can be expressed by attaching the qualification symbol to a position symbol.</p>
<p>Number of Positions</p>	
	<p>A position type may be assigned to more than one organisational unit. In this case, the number of respective instances can be represented using a person icon that is attached to a corresponding "is part of" relationship.</p>
	<p>A position type may be assigned to more than one organisational unit. In this case, the number of respective instances can be represented using a person icon that is attached to a corresponding "is part of" relationship.</p>

Table 18: Representation of Relationships

<p>Interaction, generic</p> <p>The interaction between two organisational units can be represented by connecting them with a continuous line, the thickness of which is an indicator of the interaction frequency.</p>


<p>Interaction, synchronous, technical medium</p> <p>The telephone symbol can be used to characterise communication as synchronous with the use of a technical medium. Note that the medium does not have to be a telephone in the original sense. It may also be realised with other technologies, e.g. through Voice over IP. The optional textbox can be used to define additional features of an interaction relationship. This is the case for all other kinds of interaction, too.</p>



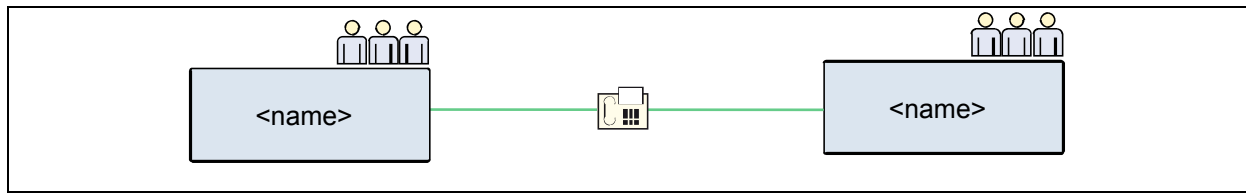


Table 19: Representation of Interaction Relationships

Organisational units can be grouped into – possibly overlapping – categories. A category is represented as a rounded down rectangle that encloses the symbols representing the respective organisational units. The colour of the rectangle and the surrounding line is subject to individual choice. The line may be dotted as in the following example and in Figure 17 or continuous as in Figure 19. Positions can be grouped into categories accordingly. Text boxes can be attached to categories of organisational units or positions to represent particular features (see Figure 20). Category names can be placed in the same boxes used for organisational units – and supplemented with performance symbols and text boxes (see Figure 20).

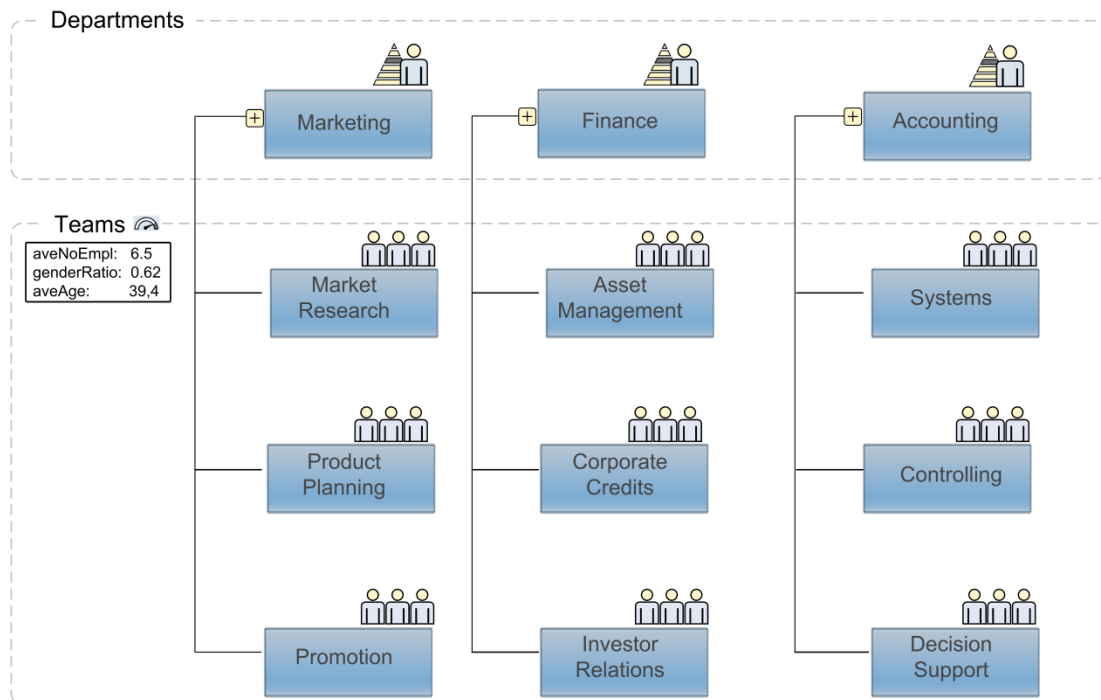


Figure 14: Representation of Categories

Comments and Constraints		
matt	<p>The diagram shows two small boxes, each labeled 'C 1', connected by lines to a larger, light gray box labeled '<Comment>'.</p>	<p>A comment can be assigned to any part of a diagram. It serves to provide a description/explanation to foster an adequate understanding of a model. It can be connected to the respective part of a dia-</p>


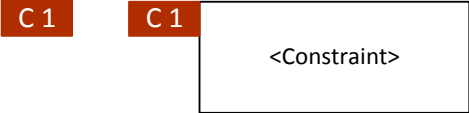
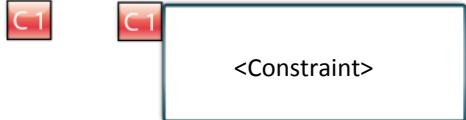
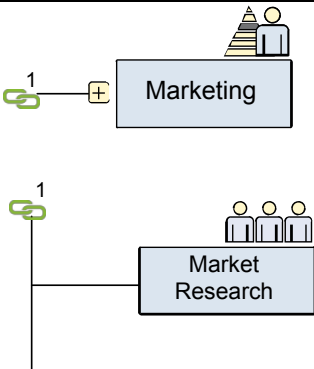
glossy		<p>gram through a dotted line. If there is space enough, the box that includes the comment can be attached directly. Otherwise, one may attach the key only and refer to a separate representation of the comment. The key comprises of a “C” for “Comment” and an additional integer.</p>
matt		<p>A constraint serves to reduce ambiguities within a model, i.e. it reduces the range of permissible interpretations. In an ideal case, constraints should be specified using the OCL or some other formal language. However, if at a certain point in time a formal specification is not an option, a constraint can be defined using a natural language expression as well.</p>
glossy		<p>If there is space enough, the box that includes the constraint can be linked to the part of the diagram it applies to directly – again through a dotted line. Otherwise, one may attach the key only and refer to a separate representation of the comment. The key comprises of a “C” for “Constraint” and an additional integer.</p>

Figure 15: Comments and Constraints

Connectors		
matt		<p>Often, diagrams reach a size that does not fit onto a page of a certain medium anymore. In this case, a connector can be used. A connector can replace any element of a diagram. Its number serves to identify the related part of the diagram that is shown on a further page.</p>

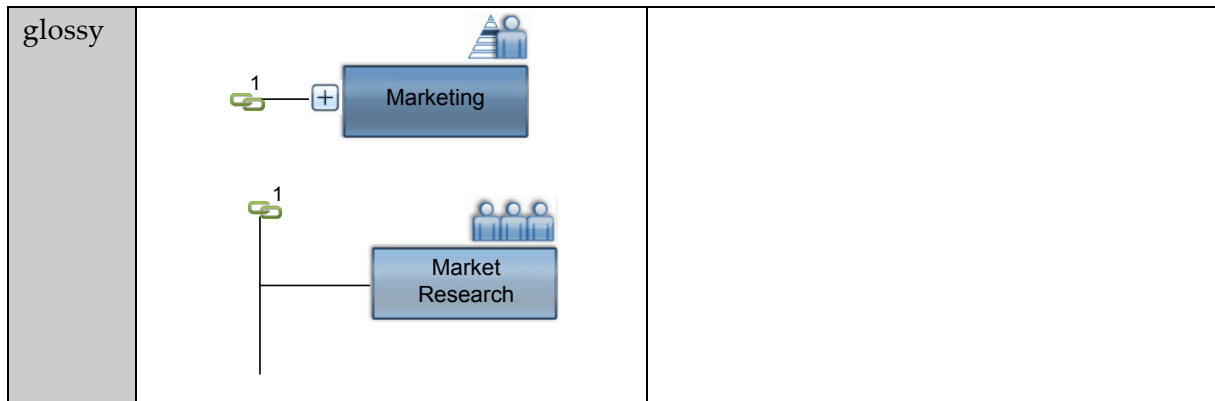


Figure 16: Connectors

4.6 Example Diagrams

The following examples serve to illustrate the use of the OgML for modelling organisation structures. The diagram in Figure 17 shows an example of a functional organisation. It makes use of the “glossy” notation variant. To support distinguishing between different types of organisational units (in this case: board, departments and teams), different colours are used. In addition to that the dotted lines mark respective categories. The relationships between organisational units reflect both, aggregation (teams as part of departments) and superior (board superior to departments).

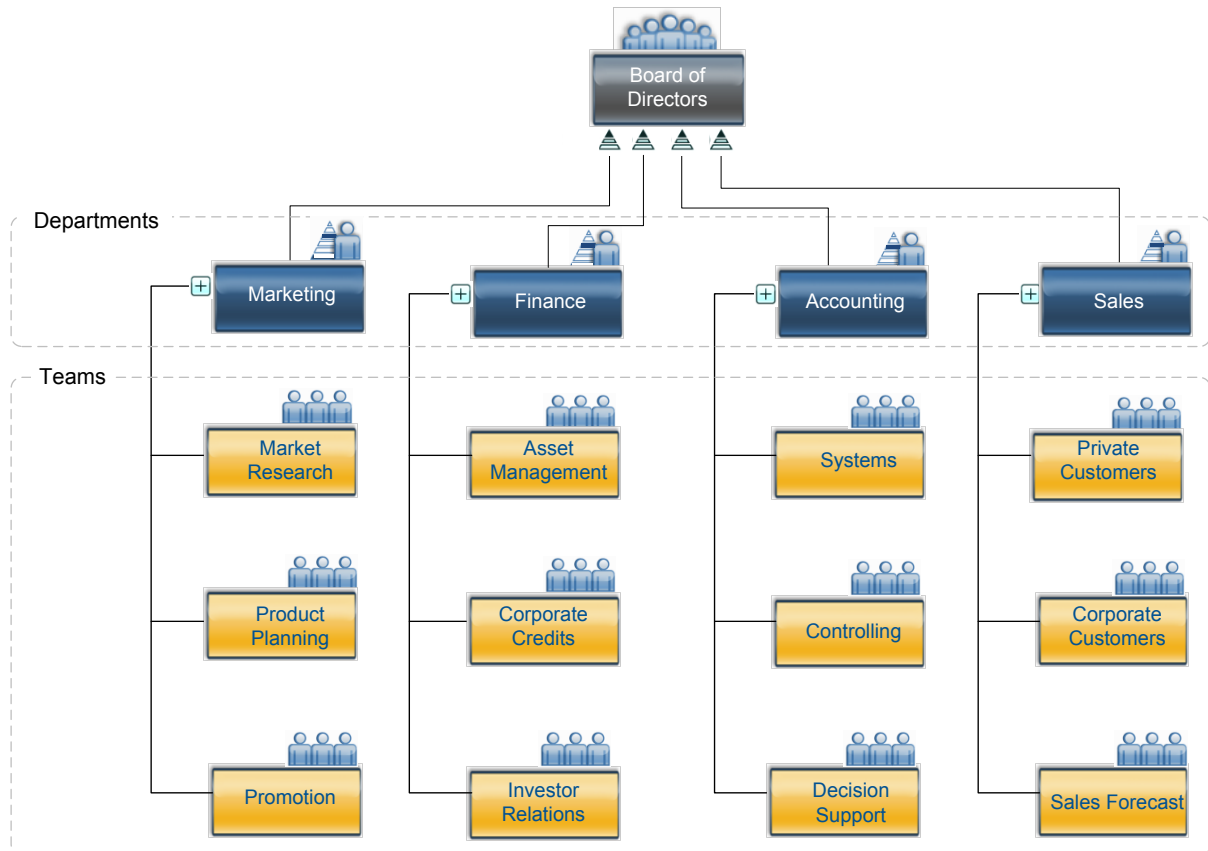


Figure 17: Functional Organisation without Detail

Figure 18 shows a slightly more differentiated diagram of a similar organisation in the “matt” notation. It includes the department managers’ positions as well as a detailed representation of the management board.

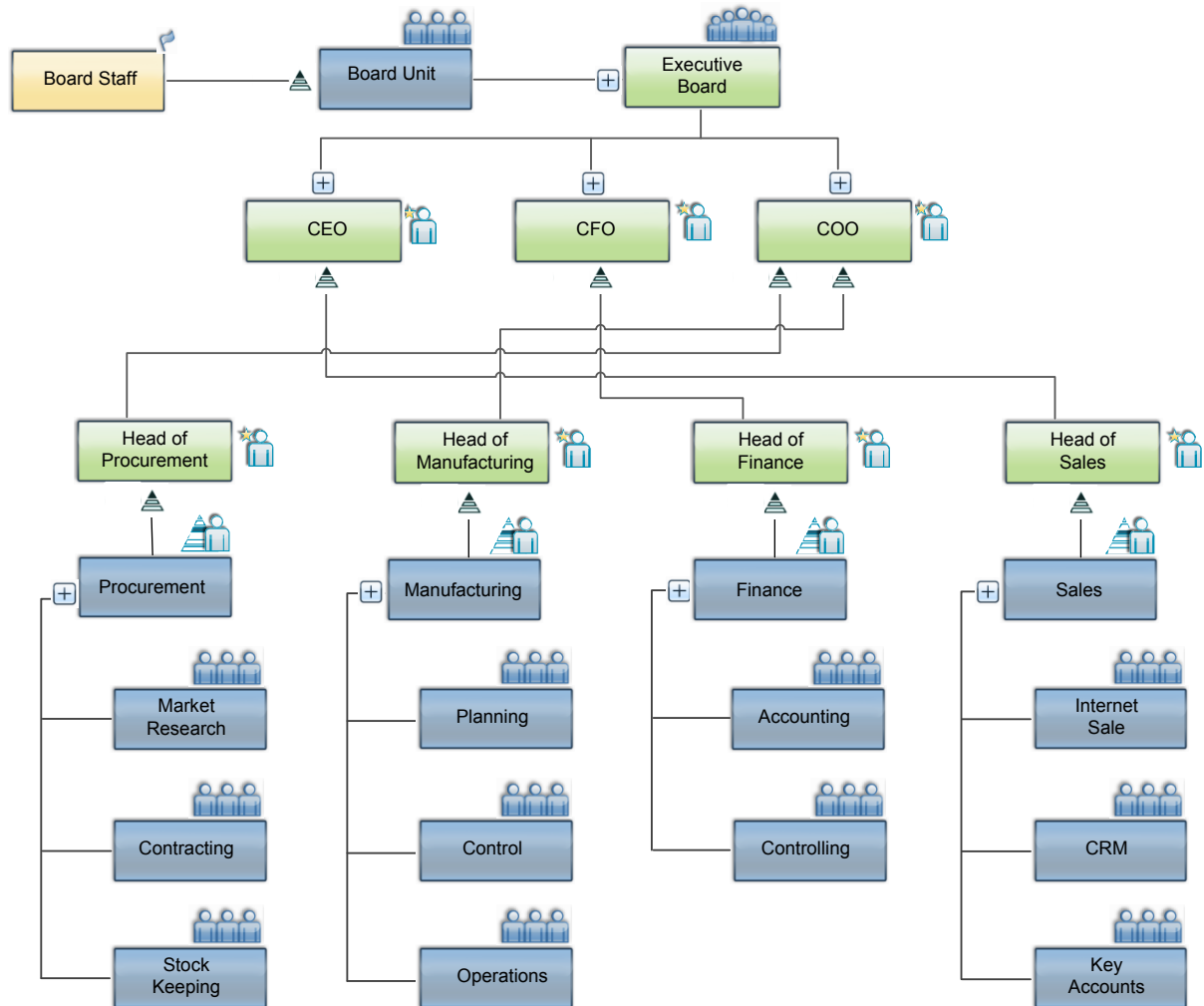


Figure 18: Example of Functional Organisation Structure Diagram

Matrix organisations combine functional and object-oriented aspects for building organisational units. Figure 19 shows a corresponding diagram, with one dimension that is related to products and a further dimension that comprises functions. To facilitate a clear optical distinction between these two dimensions corresponding categories are represented as coloured rectangles.

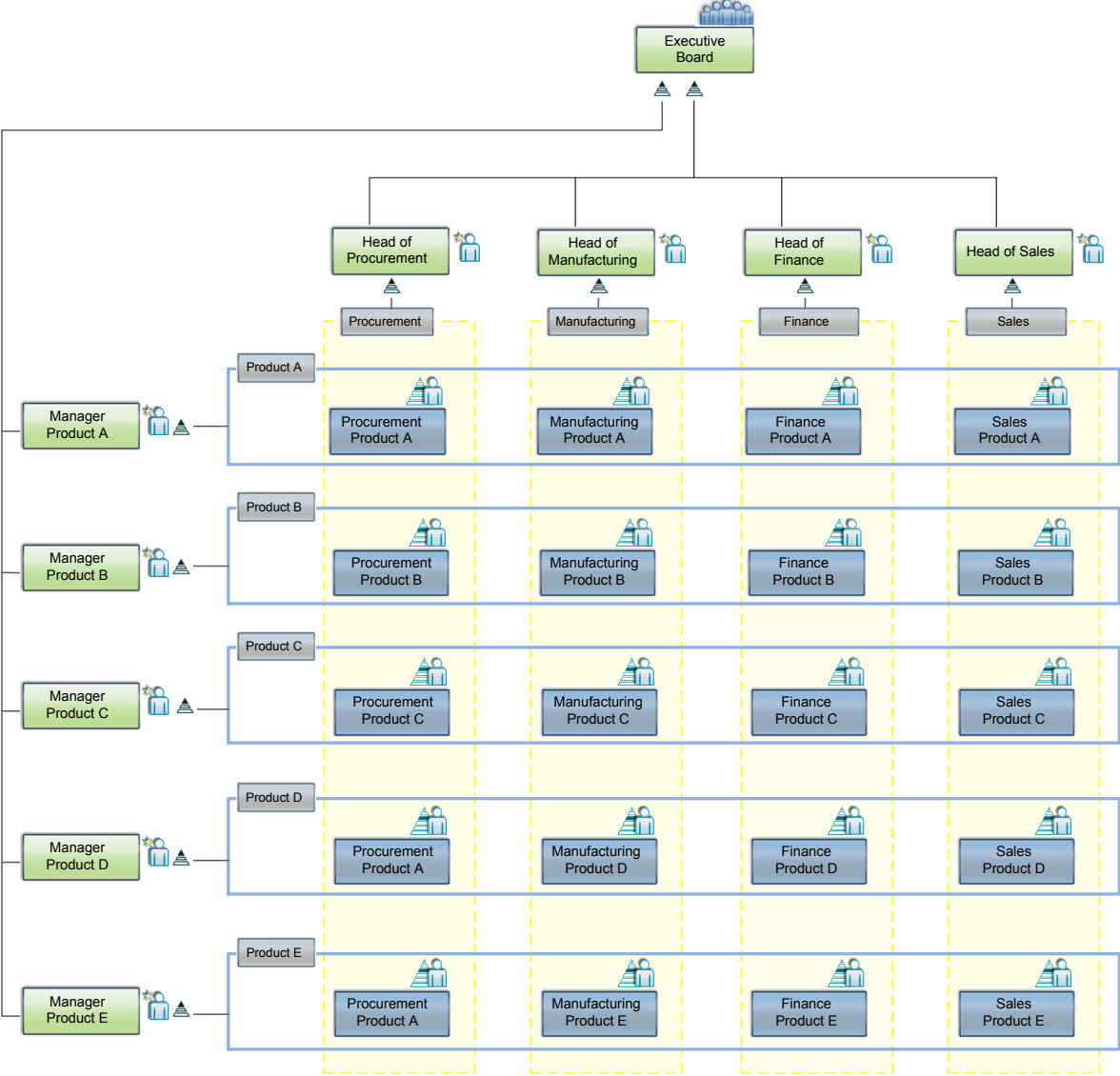


Figure 19: Representation of Matrix Organisation

The diagram in Figure 20 puts emphasis on additional information such as number of employees (positions) assigned to an organisational unit, and performance indicators. Note that for simplification reasons, the same indicator symbol was used for all organisational units.

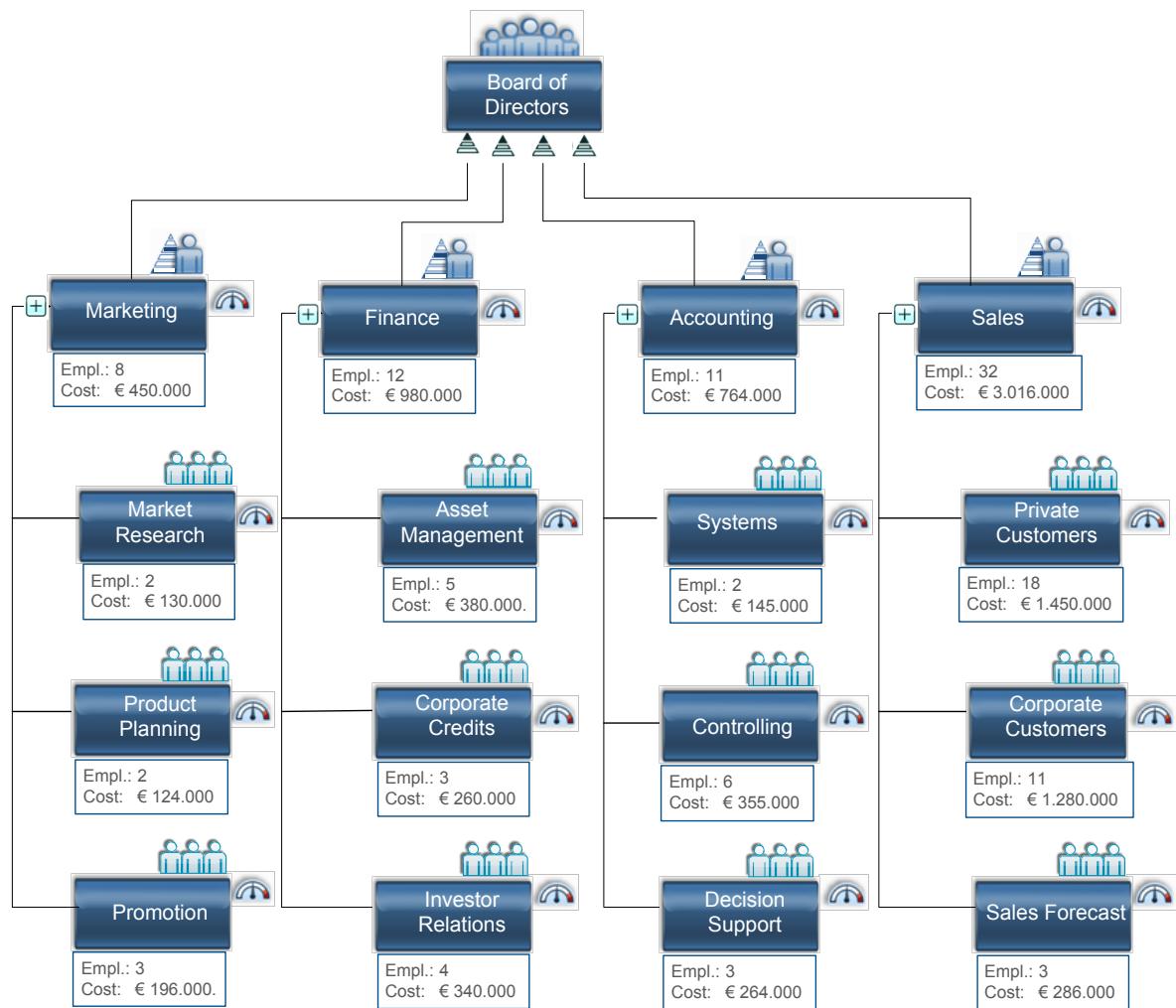


Figure 20: Organisational Chart with Additional Information

The previous examples did not include the explicit representation of positions. The following example (Figure 21) includes management positions only to illustrate the management structure of a company.

Analysis of Potential Language Concepts

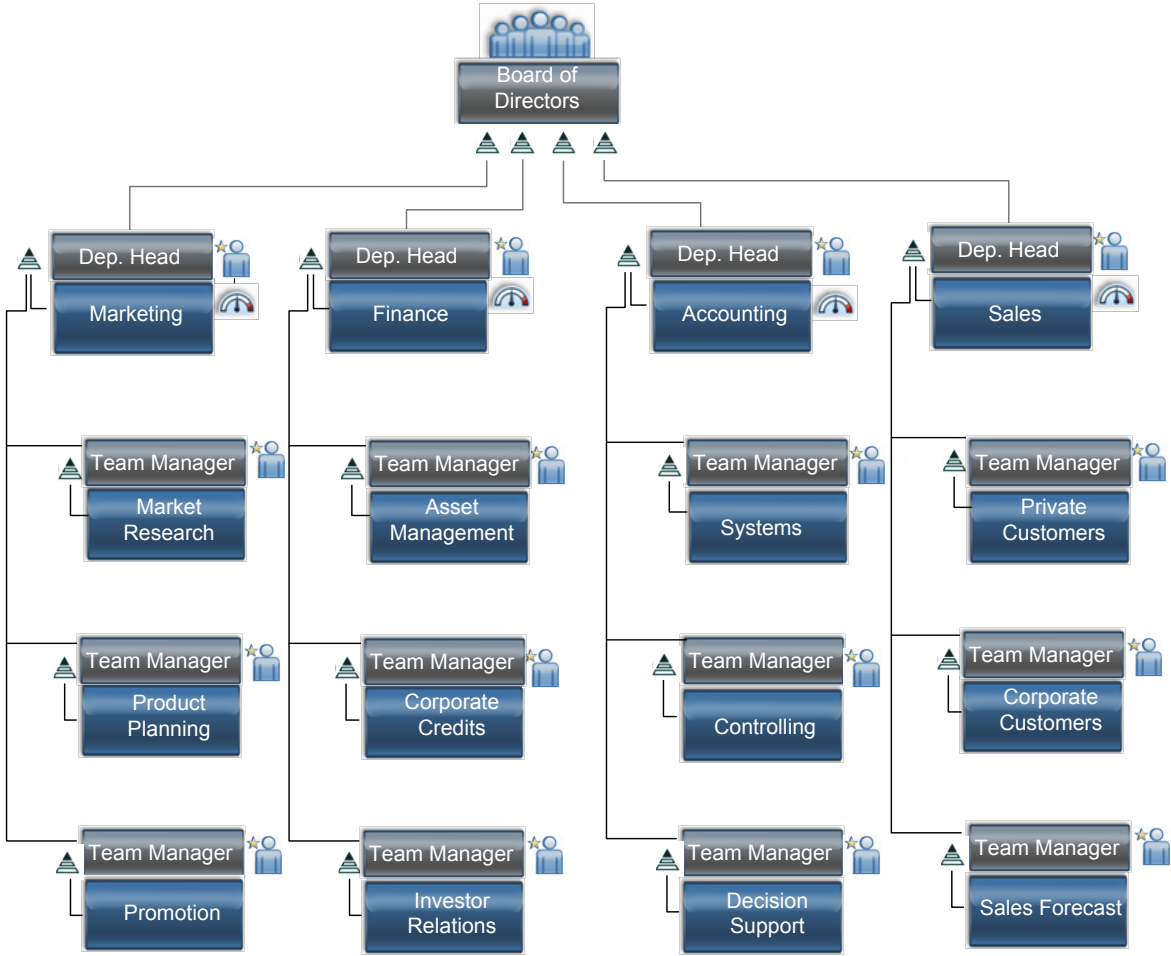


Figure 21: Focus on Management Structure

The diagram shown in Figure 22 illustrates how position types can be assigned to organisational units. Each assignment can be supplemented with a symbol that carries the number of positions of the respective type. In addition to that, each position type is characterised by the number of instances, the number of open positions and average and total personnel costs.

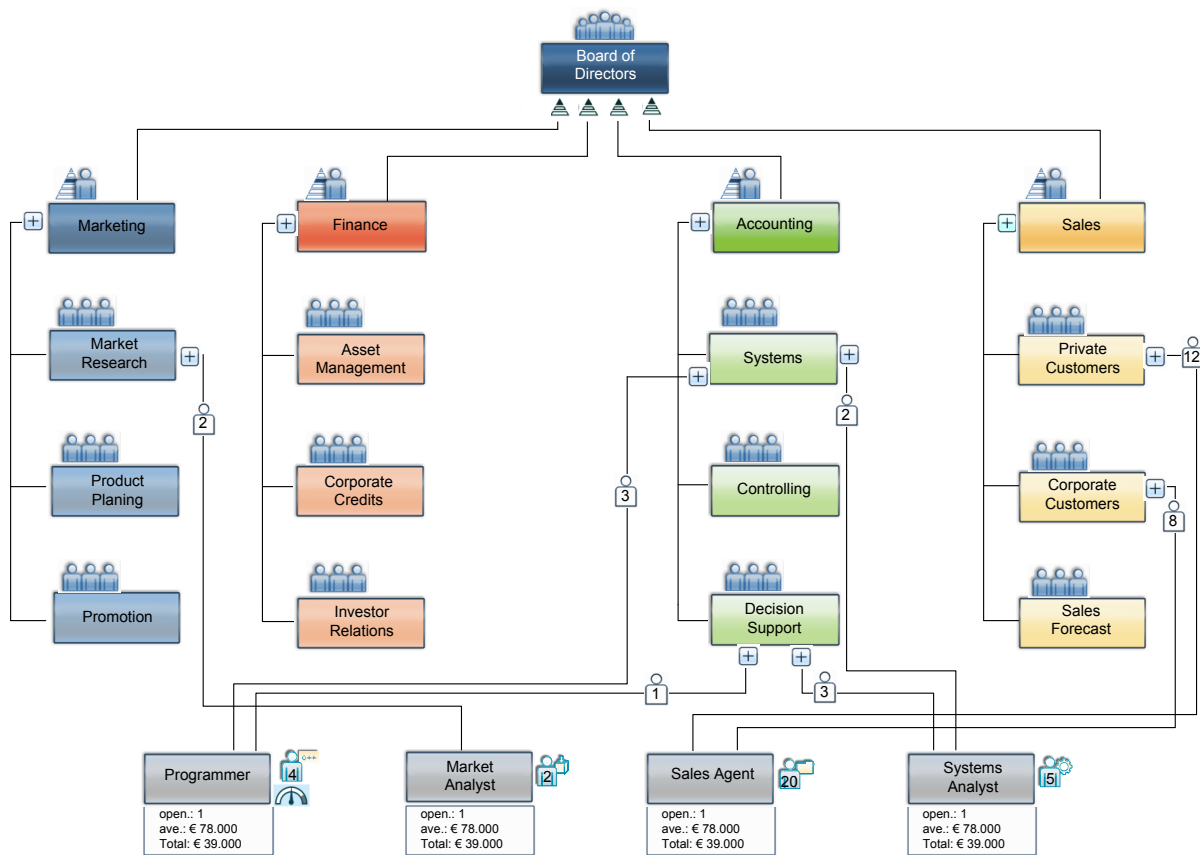


Figure 22: Assigning Positions to Organisational Units

The example in Figure 23 illustrates the representation of committees (“Quality Circle”, “Information Security” etc.) and roles (“Quality Commissioner”).

For reasons outlined above the OrgML does not include specific meta types of organisational units, e.g. “Department” or “Division”. As a consequence, types of organisational units can be used in a way that is not consistent with the prevalent or local terminology. For instance: A team could be modelled as being composed of departments. The concept of a LocalType allows for extending the OrgML with concepts that are characteristic for a certain domain. The example in Figure 24 illustrates the definition and use of two local types, “Department” and “Division”. As a consequence of defining “Division” on a higher level than “Department”, modelling a particular division as part of a particular department would result in an error. Note that it is possible to assign specific graphical symbols to each local type to support differentiating them.

Analysis of Potential Language Concepts

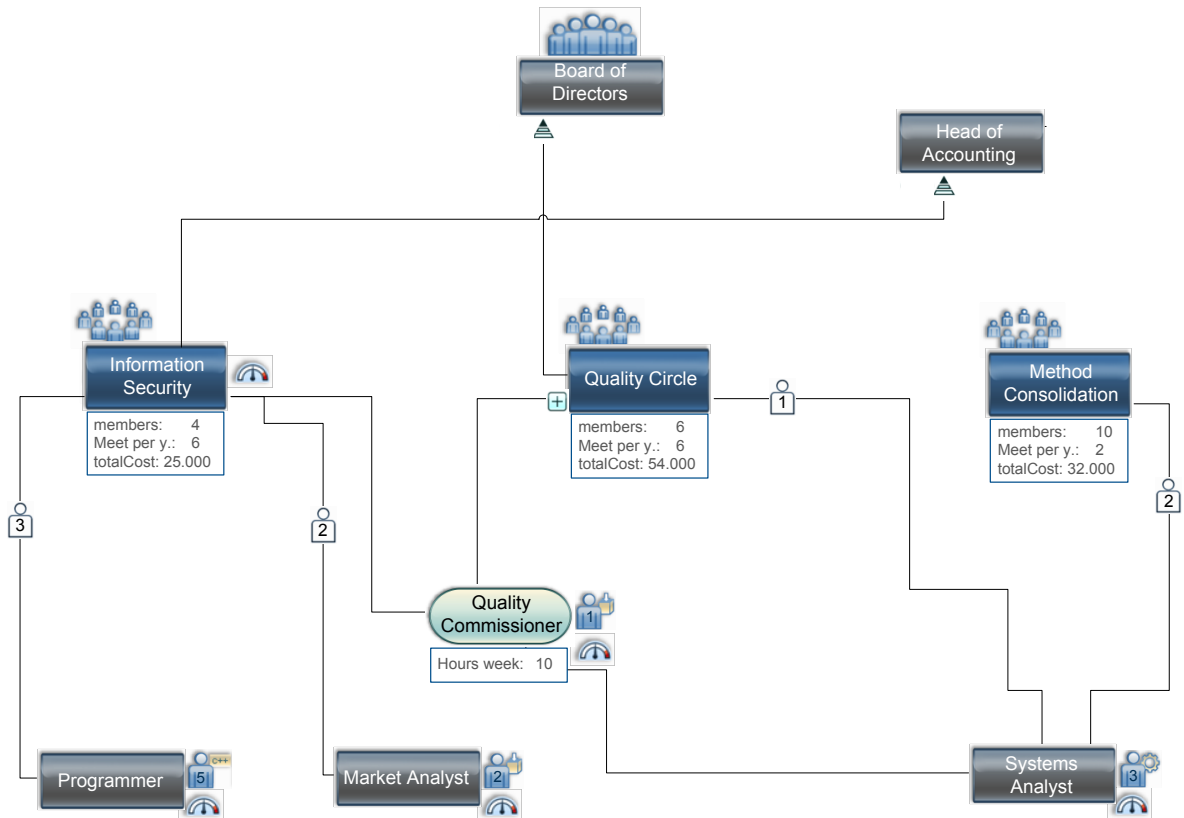


Figure 23: Representation of Committees and Roles

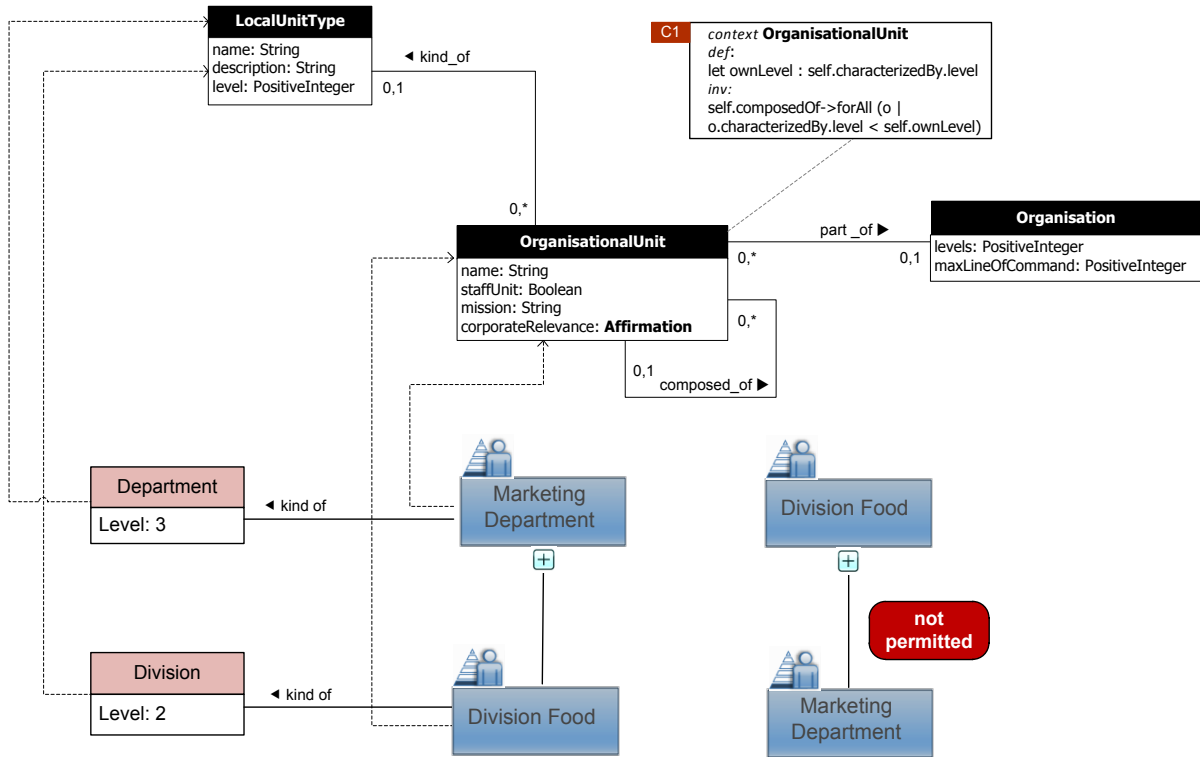


Figure 24: Defining and Using Local Types

While the main focus of the OrgML is on static relationships between organisational units, it also allows for representing interaction/communication relationships. An interaction diagram represents binary interaction relationships within a set of organisational units and/or organisations. Each interaction can be described on different levels of abstraction and detail. Figure 25 shows an interaction diagram that visualises the interaction frequency between organisational units within a certain time period. Two interaction relationships are augmented with further details using a textbox.

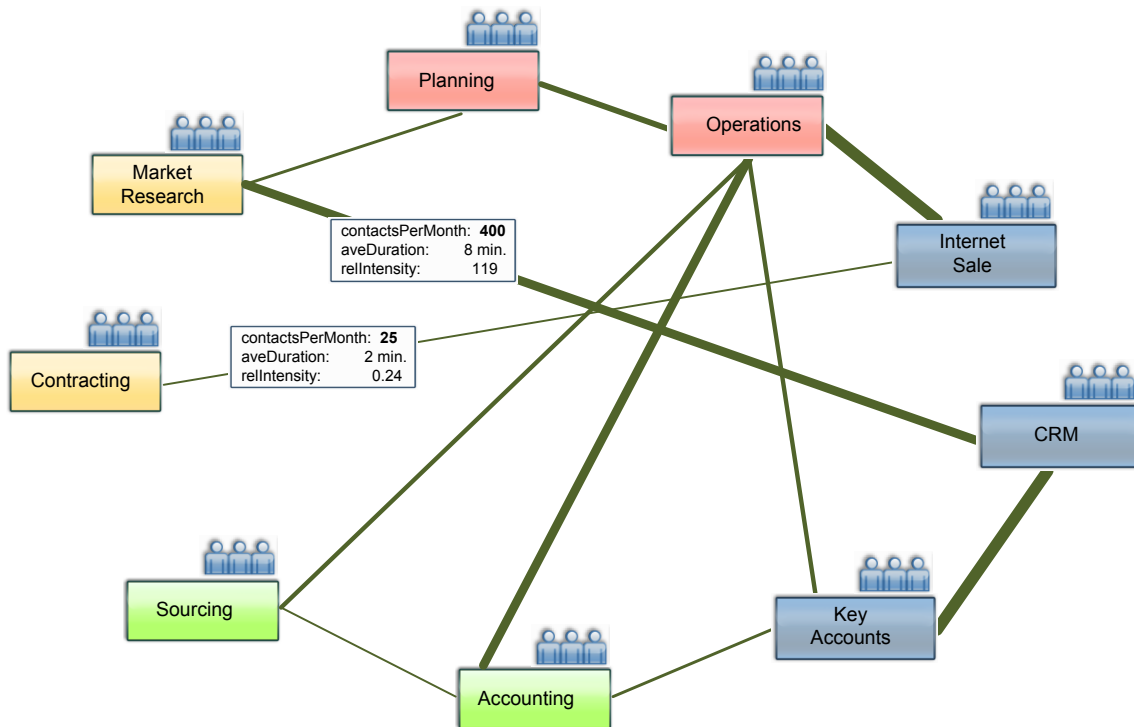


Figure 25: Example of Interaction Diagram

The example in Figure 26 illustrates how interaction relationships can be differentiated with respect to media.

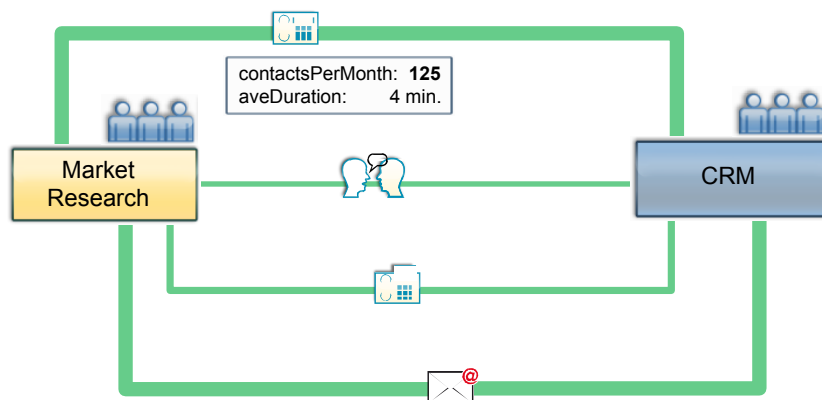


Figure 26: Interaction via Different Media

Analysis of Potential Language Concepts

The above examples do not only illustrate the concrete syntax of the language, they also demonstrate that the comprehensibility of a diagram depends on the layout. There are no explicit rules for creating the layout of a diagram. In general, it is a good idea to choose a layout, prospective users are familiar with. In most cases, this will suggest to represent aggregate organisational units above their respective parts. Also, it will often foster a better comprehensibility to place positions or boards that act as superiors above the respective subordinate units.

5 Conclusions

This report presents a specification of that part of the MEMO OrgML that serves modelling organisational structures. An evaluation will be performed on the entire language – including the process modelling part – in an upcoming report. In many organisations, organisational charts are used as *drawings* of the organisational structure. While these drawings may help with illustrating organisational structures, they are of limited use for elaborate analyses. They provide a particular level of detail only that does not fit every analysis. Also, they usually lack the precision and consistency that would be required for thoroughly analysing an organisation. A DSML that provides concepts for modelling organisational structures allows for overcoming the deficiencies of organisational charts. Its syntax and semantics allow for building model editors that prevent formally inconsistent models and that allow for selecting between different diagrams of one model. Furthermore, a tool would allow for performing machine analysis and could be used to generate code of corresponding software systems, e.g. a schema of an organisational information system. Last but not least, the language specification allows for integrating the OrgML with other MEMO languages that are based on the same meta meta model. Integrating models of organisational structures with other models, e.g. business process models, strategy models, IT infrastructure models etc. enriches them with semantics and allows for further, more elaborate analyses. But even though a model of organisation structures that is embedded in an enterprise models supports analysing an organisation from different perspectives, it would be too naïve to assume this would enable a comprehensive description of an organisation.

"Stated in more conventional terms, there is a difference between the full and rich reality of an organization, and the knowledge that we are able to gain about that organization. We can know organizations only through our experience of them. We can use metaphors and theories to grasp and express this knowledge and experience, and to share our understandings, but we can never be sure that we are absolutely right. I believe we must always recognise this basic uncertainty." (Morgan 1986, p. 341)

Therefore, models of organisational structures – as well as enterprise models in general – should always be regarded as constructions. These constructions serve certain purposes. They hopefully foster elaborate analyses and more rational, cross-perspective discourses. However, they should always be supplemented by accounting for aspects of social reality that bulk against formalisation, e.g. the effect of (mutual) expectations, symbolic action, power, personal (hidden) interests etc.

Future work on the OrgML will focus on concepts that enable the representation of “non standard” types of organisation structure, e.g. for project organisation, virtual enterprises or for the integration of external “nomads”.

References

- FRANK, U. 2011a. The MEMO Meta Modelling Language (MML) and Language Architecture. ICB Research Report University Duisburg-Essen. 2nd Edition.
- FRANK, U. 2011b. MEMO Organisation Modelling Language: Requirements and Core Diagram Types. *ICB Research Report No 47*.
- FRANK, U. 2001a. Organising the Corporation: Research Perspectives, Concepts and Diagrams. *Research Report, University Koblenz-Landau*.
- FRANK, U. 2001b. Organising the Corporation: Research Perspectives, Concepts and Diagrams. Institut für Wirtschaftsinformatik, Universität Koblenz-Landau
- FRANK, U. 2011c. Some Guidelines for the Conception of Domain-Specific Modelling Languages. In: NÜTTGENS, M., THOMAS, O. & WEBER, B. (eds.) *Enterprise Modelling and Information Systems Architectures*. Hamburg: GI.
- FRANK, U., HEISE, D., KATTENSTROTH, H. & SCHAUER, H. 2008. Designing and Utilizing Business Indicator Systems within Enterprise Models – Outline of a Method. *Modeling Business Information Systems (MoBIS)*.
- LEONTIEF, A. N. 1978. *Activity, Consciousness and Personality*, New York, Prentice Hall.
- MORGAN, G. 1986. *Images of Organization*, London, Thousand Oaks.
- OMG. 2006. OMG: Object Constraint Language. Available: <http://www.omg.org/spec/OCL/2.0/PDF>.
- TAYLOR, F. W. 1911. *The Principles of Scientific Management*, New York, Harper.
- WARD, J. W. 1964. The Idea of Individualism and the Reality of Organization. In: CHEIT & F., E. (eds.) *The Business Establishment*. New York: Wiley.

Previously published ICB - Research Reports

2011

No 47 (December 2011)

Frank, Ulrich: "MEMO Organisation Modelling Language (OrgML): Requirements and Core Diagram Types"

No 46 (December 2011)

Frank, Ulrich: "Multi-Perspective Enterprise Modelling: Background and Terminological Foundation"

No 45 (November 2011)

Frank, Ulrich; Strecker, Stefan; Heise, David; Kattenstroth, Heiko; Schauer, Carola: "Leitfaden zur Erstellung wissenschaftlicher Arbeiten in der Wirtschaftsinformatik"

No 44 (September 2010)

Berenbach, Brian; Daneva, Maya; Dörr, Jörg; Frickler, Samuel; Gervasi, Vincenzo; Glinz, Martin; Herrmann, Andrea; Krams, Benedikt; Madhavji, Nazim H.; Paech, Barbara; Schockert, Sixten; Seyff, Norbert (Eds): "17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2011). Proceedings of the REFSQ 2011 Workshops REEW, EPICAL and RePriCo, the REFSQ 2011 Empirical Track (Empirical Live Experiment and Empirical Research Fair), and the REFSQ 2011 Doctoral Symposium"

No 43 (February 2011)

Frank, Ulrich: "The MEMO Meta Modelling Language (MML) and Language Architecture – 2nd Edition"

2010

No 42 (December 2010)

Frank, Ulrich: "Outline of a Method for Designing Domain-Specific Modelling Languages"

No 41 (December 2010)

Adelsberger, Heimo; Drechsler, Andreas (Eds): "Ausgewählte Aspekte des Cloud-Computing aus einer IT-Management-Perspektive – Cloud Governance, Cloud Security und Einsatz von Cloud Computing in jungen Unternehmen"

No 40 (October 2010)

Bürsner, Simone; Dörr, Jörg; Gehlert, Andreas; Herrmann, Andrea; Herzwurm, Georg; Janzen, Dirk; Merten, Thorsten; Pietsch, Wolfram; Schmid, Klaus; Schneider, Kurt; Thurimella, Anil Kumar (Eds): "16th International Working Conference on Requirements Engineering: Foundation for Software Quality. Proceedings of the Workshops CreaRE, PLREQ, RePriCo and RESC"

No 39 (May 2010)

Strecker, Stefan; Heise, David; Frank, Ulrich: "Entwurf einer Mentoring-Konzeption für den Studiengang M.Sc. Wirtschaftsinformatik an der Fakultät für Wirtschaftswissenschaften der Universität Duisburg-Essen"

No 38 (February 2010)

Schauer, Carola: "Wie praxisorientiert ist die Wirtschaftsinformatik? Einschätzungen von CIOs und WI-Professoren"

No 37 (January 2010)

Benavides, David; Batory, Don; Grunbacher, Paul (Eds.): "Fourth International Workshop on Variability Modelling of Software-intensive Systems"

2009

No 36 (December 2009)

Strecker, Stefan: "Ein Kommentar zur Diskussion um Begriff und Verständnis der IT-Governance - Anregungen zu einer kritischen Reflexion"

No 35 (August 2009)

Rüngeler, Irene; Tüxen, Michael; Rathgeb, Erwin P.: "Considerations on Handling Link Errors in STCP"

No 34 (June 2009)

Karastoyanova, Dimka; Kazhamiakan, Raman; Metzger, Andreas; Pistore, Marco (Eds.): "Workshop on Service Monitoring, Adaption and Beyond"

No 33 (May 2009)

Adelsberger, Heimo; Drechsler, Andreas; Bruckmann, Tobias; Kalvelage, Peter; Kinne, Sophia; Pellingner, Jan; Rosenberger, Marcel; Trepper, Tobias: „Einsatz von Social Software in Unternehmen – Studie über Umfang und Zweck der Nutzung“

No 32 (April 2009)

Barth, Manfred; Gadatsch, Andreas; Kütz, Martin; Rüdiger, Otto; Schauer, Hanno; Strecker, Stefan: „Leitbild IT-Controller/-in – Beitrag der Fachgruppe IT-Controlling der Gesellschaft für Informatik e. V.“

No 31 (April 2009)

Frank, Ulrich; Strecker, Stefan: "Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems – Requirements, Conceptual Foundation and Design Options"

No 30 (February 2009)

Schauer, Hanno; Wolff, Frank: „Kriterien guter Wissensarbeit – Ein Vorschlag aus dem Blickwinkel der Wissenschaftstheorie (Langfassung)“

No 29 (January 2009)

Benavides, David; Metzger, Andreas; Eisenecker, Ulrich (Eds.): "Third International Workshop on Variability Modelling of Software-intensive Systems"

2008

No 28 (December 2008)

Goedicke, Michael; Striewe, Michael; Balz, Moritz: „Computer Aided Assessments and Programming Exercises with JACK“

No 27 (December 2008)

Schauer, Carola: "Größe und Ausrichtung der Disziplin Wirtschaftsinformatik an Universitäten im deutschsprachigen Raum - Aktueller Status und Entwicklung seit 1992"

No 26 (September 2008)

Milen, Tilev; Bruno Müller-Clostermann: " CapSys: A Tool for Macroscopic Capacity Planning"

No 25 (August 2008)

Eicker, Stefan; Spies, Thorsten; Tschersich, Markus: "Einsatz von Multi-Touch beim Softwaredesign am Beispiel der CRC Card-Methode"

No 24 (August 2008)

Frank, Ulrich: "The MEMO Meta Modelling Language (MML) and Language Architecture – Revised Version"

No 23 (January 2008)

Sprenger, Jonas; Jung, Jürgen: "Enterprise Modelling in the Context of Manufacturing – Outline of an Approach Supporting Production Planning"

No 22 (January 2008)

Heymans, Patrick; Kang, Kyo-Chul; Metzger, Andreas, Pohl, Klaus (Eds.): "Second International Workshop on Variability Modelling of Software-intensive Systems"

2007

No 21 (September 2007)

Eicker, Stefan; Annett Nagel; Peter M. Schuler: "Flexibilität im Geschäftsprozess-management-Kreislauf"

No 20 (August 2007)

Blau, Holger; Eicker, Stefan; Spies, Thorsten: "Reifegradüberwachung von Software"

No 19 (June 2007)

Schauer, Carola: "Relevance and Success of IS Teaching and Research: An Analysis of the 'Relevance Debate'"

No 18 (May 2007)

Schauer, Carola: "Rekonstruktion der historischen Entwicklung der Wirtschaftsinformatik: Schritte der Institutionalisierung, Diskussion zum Status, Rahmenempfehlungen für die Lehre"

No 17 (May 2007)

Schauer, Carola; Schmeing, Tobias: "Development of IS Teaching in North-America: An Analysis of Model Curricula"

No 16 (May 2007)

Müller-Clostermann, Bruno; Tilev, Milen: "Using G/G/m-Models for Multi-Server and Mainframe Capacity Planning"

No 15 (April 2007)

Heise, David; Schauer, Carola; Strecker, Stefan: "Informationsquellen für IT-Professionals – Analyse und Bewertung der Fachpresse aus Sicht der Wirtschaftsinformatik"

No 14 (March 2007)

Eicker, Stefan; Hegmanns, Christian; Malich, Stefan: "Auswahl von Bewertungsmethoden für Softwarearchitekturen"

No 13 (February 2007)

Eicker, Stefan; Spies, Thorsten; Kahl, Christian: "Softwarevisualisierung im Kontext serviceorientierter Architekturen"

No 12 (February 2007)

Brenner, Freimut: "Cumulative Measures of Absorbing Joint Markov Chains and an Application to Markovian Process Algebras"

No 11 (February 2007)

Kirchner, Lutz: "Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements – Grundlagen, Anforderungen und Metamodell"

No 10 (February 2007)

Schauer, Carola; Strecker, Stefan: "Vergleichende Literaturstudie aktueller einführender Lehrbücher der Wirtschaftsinformatik: Bezugsrahmen und Auswertung"

No 9 (February 2007)

Strecker, Stefan; Kuckertz, Andreas; Pawlowski, Jan M.: "Überlegungen zur Qualifizierung des wissenschaftlichen Nachwuchses: Ein Diskussionsbeitrag zur (kumulativen) Habilitation"

No 8 (February 2007)

Frank, Ulrich; Strecker, Stefan; Koch, Stefan: "Open Model - Ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik (Langfassung)"

2006

No 7 (December 2006)

Frank, Ulrich: "Towards a Pluralistic Conception of Research Methods in Information Systems Research"

No 6 (April 2006)

Frank, Ulrich: "Evaluation von Forschung und Lehre an Universitäten – Ein Diskussionsbeitrag"

No 5 (April 2006)

Jung, Jürgen: "Supply Chains in the Context of Resource Modelling"

No 4 (February 2006)

Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part III – Results Wirtschaftsinformatik Discipline"

2005

No 3 (December 2005)

Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part II – Results Information Systems Discipline"

No 2 (December 2005)

Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part I – Research Objectives and Method"

No 1 (August 2005)

Lange, Carola: „Ein Bezugsrahmen zur Beschreibung von Forschungsgegenständen und -methoden in Wirtschaftsinformatik und Information Systems“

Research Group	Core Research Topics
Prof. Dr. H. H. Adelsberger Information Systems for Production and Operations Management	E-Learning, Knowledge Management, Skill-Management, Simulation, Artificial Intelligence
Prof. Dr. P. Chamoni MIS and Management Science / Operations Research	Information Systems and Operations Research, Business Intelligence, Data Warehousing
Prof. Dr. F.-D. Dorloff Procurement, Logistics and Information Management	E-Business, E-Procurement, E-Government
Prof. Dr. K. Echtele Dependability of Computing Systems	Dependability of Computing Systems
Prof. Dr. S. Eicker Information Systems and Software Engineering	Process Models, Software-Architectures
Prof. Dr. U. Frank Information Systems and Enterprise Modelling	Enterprise Modelling, Enterprise Application Integration, IT Management, Knowledge Management
Prof. Dr. M. Goedicke Specification of Software Systems	Distributed Systems, Software Components, CSCW
Prof. Dr. V. Gruhn Software Engineering	Design of Software Processes, Software Architecture, Usability, Mobile Applications, Component-based and Generative Software Development
PD Dr. C. Klüver Computer Based Analysis of Social Complexity	Soft Computing, Modeling of Social, Cognitive, and Economic Processes, Development of Algorithms
Prof. Dr. T. Kollmann E-Business and E-Entrepreneurship	E-Business and Information Management, E-Entrepreneurship/E-Venture, Virtual Marketplaces and Mobile Commerce, Online-Marketing
Prof. Dr. B. Müller-Clostermann Systems Modelling	Performance Evaluation of Computer and Communication Systems, Modelling and Simulation
Prof. Dr. K. Pohl Software Systems Engineering	Requirements Engineering, Software Quality Assurance, Software-Architectures, Evaluation of COTS/Open Source-Components
Prof. Dr.-Ing. E. Rathgeb Computer Networking Technology	Computer Networking Technology
Prof. Dr. E. Rukzio Mobile Mensch Computer Interaktion mit Software Services	Novel Interaction Technologies, Personal Projectors, Pervasive User Interfaces, Ubiquitous Computing
Prof. Dr. R. Unland Data Management Systems and Knowledge Representation	Data Management, Artificial Intelligence, Software Engineering, Internet Based Teaching
Prof. Dr. S. Zelewski Institute of Production and Industrial Information Management	Industrial Business Processes, Innovation Management, Information Management, Economic Analyses