Models, To Model, and Modelling
-
Towards a Theory of Conceptual Models and Modelling
-
Towards a Notion of the Model

Collection of Recent Papers

Bernhard Thalheim

Christian-Albrechts-University Kiel, Department of Computer Science, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Content of this collection

References

[NT15] I. Nissen and B. Thalheim. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*, chapter The Notion of a Model, pages 615–618. De Gryuter, Boston, 2015.

[TD15] B. Thalheim and A. Dahanayake. A conceptual model for services. In *Invited Keynote, CMS 2015, ER 2015 workshop*, LNCS 9382, pages 51–61, Berlin, 2015. Springer.

[Tha09] B. Thalheim. Towards a theory of conceptual modelling. In *ER Workshops*, volume 5833 of *Lecture Notes in Computer Science*, pages 45–54. Springer, 2009.

[Tha10] B. Thalheim. Towards a theory of conceptual modelling. *Journal of Universal Computer Science*, 16(20):3102–3137, 2010. http://www.jucs.org/jucs_16_20/towards_a_theory_of.

[Tha11a] B. Thalheim. The science of conceptual modelling. In *Proc. DEXA 2011*, volume 6860 of *LNCS*, pages 12–26, Berlin, 2011. Springer.

[Tha11b] B. Thalheim. The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 17, pages 547–580. Springer, Berlin, 2011.

[Tha12a] B. Thalheim. The art of conceptual modelling. In *Information Modelling and Knowledge Bases XXII*, volume 237 of *Frontiers in Artificial Intelligence and Applications*, pages 149–168. IOS Press, 2012.

[Tha12b] B. Thalheim. The science and art of conceptual modelling. In A. Hameurlain et al., editor, *TLDKS VI*, LNCS 7600, pages 76–105. Springer, Heidelberg, 2012.

[Tha12c] B. Thalheim. Syntax, semantics and pragmatics of conceptual modelling. In *NLDB*, volume 7337 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2012.

[Tha13a] B. Thalheim. The definition of the (conceptual) model. In *Proc. EJC 2013*, pages 256–269, Nara, Japan, 2013.

[Tha13b] B. Thalheim. Open problems of information systems research and technology. In *Invited Keynote, BIR'2013. LNBIB 158*, pages 10–18. Springer, 2013.

[Tha14] B. Thalheim. The conceptual model $\equiv$ an adequate and dependable artifact enhanced by concepts. In *Information Modelling and Knowledge Bases*, volume XXV of *Frontiers in Artificial Intelligence and Applications, 260*, pages 241–254. IOS Press, 2014.

[Tha15] B. Thalheim. Das Modell des Modelles. *Erwägen-Wissen-Ethik*, EWE-Heft, Heft 3, 2015, 26. Jg.:407–409, 2015.

[TN15] B. Thalheim and I. Nissen, editors. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*. De Gruyter, Boston, 2015.

[TTF15] B. Thalheim and M. Tropmann-Frick. The conception of the conceptual database model. In *ER 2015*, LNCS 9381, pages 603–611, Berlin, 2015. Springer.

[TTF16] B. Thalheim and M. Tropmann-Frick. Models and their capability. In C. Beierle, G. Brewka, and M. Thimm, editors, *Computational Models of Rationality*, volume 29 of *College Publications Series*, pages 34–56. College Publications, 2016.

Towards a Theory of Conceptual Modelling

Bernhard Thalheim
(Christian Albrechts University Kiel, Department of Computer Science
Olshausenstr. 40, D-24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de)

Abstract: Conceptual modelling is a widely applied practice and has led to a large body of knowledge on constructs that might be used for modelling and on methods that might be useful for modelling. It is commonly accepted that database application development is based on conceptual modelling. It is however surprising that only very few publications have been published on a *theory of conceptual modelling*.

Modelling is typically supported by languages that are well-founded and easy to apply for the description of the application domain, the requirements and the system solution. It is thus based on a *theory of modelling constructs*. At the same time, modelling incorporates a description of the application domain and a prescription of requirements for supporting systems. It is thus based on methods of *application domain gathering*. Modelling is also an engineering activity with engineering steps and engineering results. It is thus *engineering*. The first facet of modelling has led to a huge body of knowledge. The second facet is considered from time to time in the scientific literature. The third facet is underexposed in the scientific literature.

This paper aims in developing principles of conceptual modelling. They cover modelling constructs as well as modelling activities as well as modelling properties. We first clarify the notion of conceptual modelling. Principles of modelling may be applied and accepted or not by the modeler. Based on these principles we can derive a theory of conceptual modelling that combines foundations of modelling constructs, application capture and engineering.

A general theory of conceptual modelling is far too comprehensive and far too complex. It is not yet visible how such a theory can be developed. This paper therefore aims in introducing a framework and an approach to a general theory of conceptual modelling. We are however in urgent need of such a theory. We are sure that this theory can be developed and use this paper for the introduction of the main ingredients of this theory.

Key Words: modelling, conceptual modelling, modelling act(ivity), principles of models and modelling, general theory of models

Category: H.2.1, H.2.2, H.1.0, M.4, I.6.5, I.6.4, H.0, L.1.0

1 Introduction

The main purpose of conceptual modelling is classically understood as the elicitation [Chen et al.(1998); Olivé(2007); Thalheim(2000)] of a high-quality conceptual schema of a (software, information, workflow, ...) system. This understanding mainly concentrates on the result of conceptual modelling and hinders the development of a general theory of conceptual modelling. Modelling is based on languages which might be sophisticated [Chen et al.(1998)] and well understood [Thalheim(2000)] like the ER modelling language or might be fuzzy with

lazy semantics development like the UML. Let us analyse the complexity of the modelling task and then let us draw some conclusions for the modelling "act".

1.1 The Three Dimensions of Conceptual Modelling

Conceptual modelling is often only discussed on the basis of modelling constructs and illustrated by some small examples. It has however three fundamental dimensions:

1. *Modelling language constructs* are applied during conceptual modelling. Their syntactics, semantics and pragmatics must be well understood.
2. *Application domain gathering* allows to understand the problems to be solved, the opportunities of solutions for a system, and the requirements and architecture that might be prescribed for the solution that has been chosen.
3. *Engineering* is oriented towards encapsulation of experiences with design problems pared down to a manageable scale.

The first dimension is handled and well understood in the literature. Except few publications, e.g. [Bjørner(2009)], the second dimension has not yet got a sophisticated and well understood support. The third dimension has received much attention by data modelers [Simsion(2007)] but did not get through to research literature. It must therefore be our goal to combine the three dimensions into a holistic framework.

1.2 Alternatives for a Notion of a Theory

The notion of a theory itself is be a matter of intensive research [Deppert(2009); Mittelstraß(2004)]. We base our understanding of the notion of a theory on the three dimensions and the main goal of conceptual modelling. This understanding is covered by the understanding of the notions of a theory[1].

The classical treatment of the notion of a theory is based on mathematical and philosophical logics and is far too strict. We may however inherit certain elements of such logics. Already the notion of semantics provides a larger number of choices [Schewe and Thalheim(2008)] beyond those that are taken for granted

[1] Websters dictionary [Web(1991)] distinguishes several understandings of the notion of theory: 1: the analysis of a set of facts in their relation to one another;
2: abstract thought; speculation;
3: the general or abstract principles of a body of fact, a science, or an art;
4a: a belief, policy, or procedure proposed or followed as the basis of action;
4b: an ideal or hypothetical set of facts, principles, or circumstances;
5: a plausible or scientifically acceptable general principle or body of principles offered to explain phenomena;
6a: a hypothesis assumed for the sake of argument or investigation
6b: an unproved assumption; conjecture
6c: a body of theorems presenting a concise systematic view of a subject; hypothesis

in Computer Science. The notion of a theory is based on a theory of truth that is based on a notion of truth and on a number of supporting theories such as a correspondence theory for truth, a coherence theory for truth, and a consensus theory for truth.

Theories of conceptual modelling must step beyond axiomatic and analytical theories. They must also be operational and 'genetic'. Theories of conceptual modelling can be developed in the frameworks of logical empiricism, of context theories ('context of use', 'language game'), and of constructivism. The first framework supports to define purposes of conceptual modelling, to emphasise threats that should be handled with the help of models, to select appropriate modelling languages and methods, to reason on the quality of the model depending on the purpose of the model, to select measures for the quality of models, and to guide the process of modelling. It may use development experiments, case studies, evaluation surveys, and implementation studies. The second framework relates models to the application domain, to the stakeholders participating in the development process, to the aspects reflected within a model, and to the resources provided either by the system and by the knowledge from the application domain. It requires to base conceptual modelling on application domain theories. The last framework provides a basis for a general structure by a *language of constructs* that can be applied for the development of a model, a *set of constructors* that can be applied to combine models into a new model, and a *number of quality properties for characterisation of usage* of certain constructs.

1.3 Implications for a Theory of Conceptual Modelling

The three dimensions of conceptual modelling must be integrated into a framework that supports the relevant dimension depending on the modelling work progress. The currently most difficult dimension is the engineering dimension. Engineering is inherently concerned with failures of construction, with incompleteness both in specification and in coverage of the application domain, with compromises for all quality dimensions, and with problems of technologies currently at hand.

At the same time, there is no universal approach and no universal language that cover all *aspects of an application*, that have a well-founded *semantics* for all constructions, that reflect any *relevant facet in applications*, and that *support engineering*. The choice of modelling languages is often a matter of preferences and case, empirical usage, evolution history, and supporting technology. Models are at different levels of abstraction and *particularisation*. We therefore are going to develop *a number of different models* that reflect different aspects of the system that is under development. [Thalheim(2009)] introduces *model suites* as a set of models with explicit *associations* among the models, with explicit *controllers* for maintenance of coherence of the models, with application schemata for their

explicit *maintenance and evolution*, and tracers for establishment of their *coherence*. Model suites and multi-model specification increases the complexity of modelling. Interdependencies among models must be given in an explicit form. Models that are used to specify different views of the same problem or application must be used consistently in an integrated form. Changes within one model must be propagated to all dependent models. Each singleton model must have a well-defined semantics as well as a number of representations for display of model content. The representation and the model must be tightly coupled. Changes within any model must either be refinements of previous models or explicit revisions of such models. The change management must support rollback to earlier versions of the model suite. Model suites may have different versions.

1.4 Quality Assessment, Control and Improvement

According to [Kangassalo(2007)] the result of conceptual modelling depends on *information available* about the UoD; on information about the UoD, regarded as *not relevant* for the concept or conceptual model at hands, and therefore abandoned or renounced; on the *philosophical background* to be applied in the modelling work; on the *additional knowledge* included by the modeler, e.g. knowledge primitives, conceptual 'components', selected logical or mathematical presuppositions, mathematical structures, etc.; on the *collection of problems* that may be investigated in this environment; on the *ontology* (or better language with its lexical semantics [Schewe and Thalheim(2008)]) used as a basis of the conceptualization process; on the *epistemological theory*, which directs how ontology should be applied in recognizing and formulating concepts, conceptual models or theories, and in constructing information, data, and knowledge, on different levels of abstraction; on the *the purpose and goal of the conceptual modelling work*; on the collection of *methods for conceptual modelling*; on the *process of the practical concept formation and modelling work*; and finally on the *knowledge and skill of the person making modelling*, as well as those of the people giving information for the modelling work.

Quality properties are

- *static qualities* of models such as the development quality (pervasiveness, analysability, changeability, stability, testability, privacy of the models, ubiquity, development parsimony), internal quality (accuracy, suitability, interoperability, robustness, self-contained, independence, internal parsimony), and quality of use (understandability, learnability, operability, attractiveness, appropriateness, user parsimony), and
- *dynamic qualities* within a selected development approach (executability, refinement quality, scope restriction, effect preservation, context explicitness, completion tracking, resource parsimony).

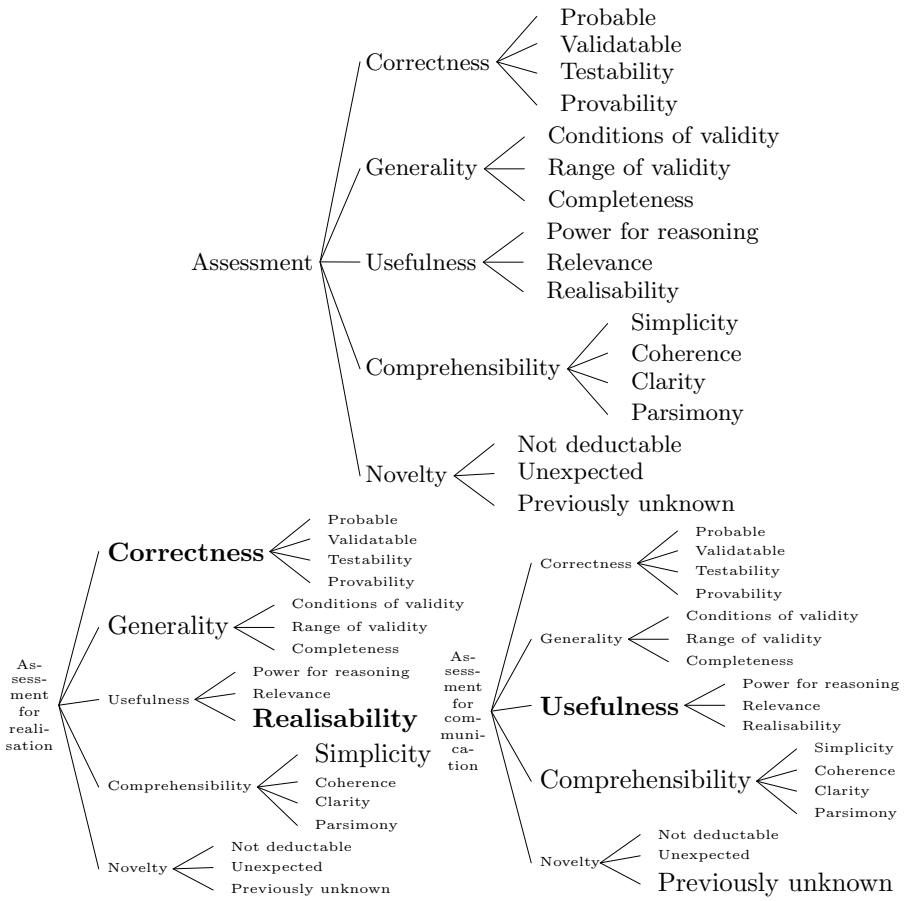


Figure 1: Assessment to quality depending on the purpose of the model: General assessment structure, assessment for realisation, and assessment for communication

The information system modelling process is intentionally or explicitly ruled by a number of development strategies, development steps, and development policies. These quality properties may alternatively be grouped into correctness, generality, usefulness, comprehensibility, and novelty. Models have very different purposes. Therefore, assessment of quality may very a lot. Figure 1 displays the general view, a specific portfolio of quality requirements for realisation, and another one for the communication purpose of a model. It shows that quality considerations are not equality relevant during development and modelling.

We may concentrate on some of these properties. Our first choice for quality properties that drive other quality properties are an *explicit statement about the*

applicability of the concept, *modality* of the concept for the model, and *confidence* of the concept inclusion into the model.

1.5 Outline and Tasks of the Paper

The three dimensions of conceptual modelling within a constructive, empirical, contextual, and quality framework lead directly to a separation of concern into a *theory of modelling constructs*, a *theory of modelling activities*, a *theory of modelling properties*, a *theory of application domain reflections*, and a *theory of engineering*. These theories can be support by other theories such as a *theory of model management* and a generalised activity theory such as a *theory of modelling styles and pattern*. These theories must have their constituents.

The paper aims in introducing a theory of conceptual modelling. The paper is directly structured by these theories. Section 2 discusses the matter of the choice of modelling constructs and model constructors. We use one example that demonstrates the impact of wrong choices. Section 3 provides an insight into different actions, activities and general tactics and strategies that can be used during conceptual for modelling. Section 4 discusses whether we can enhance conceptual modelling by properties. Finally, Section 5 summarises the achievements of this paper and derives a research agenda for a theory of conceptual modelling.

The paper targets on a general understanding of the field on conceptual modelling. Figure 2 displays a general structuring of this field into strategic, tactical and tool support layers.

It has not been our intention to survey the modelling literature[2]. This would be far too large already for the conceptual modelling research.

This paper also extends approaches of "design science" [Hevner et al.(2004)] that relates models to its purposes. Modelling creates and evaluates models intended to solve identified organisational problems. The process of modelling enables stakeholders to understand both the problem addressed by the model and the feasibility of the model to the problem solution. The theory of conceptual modelling also aims in both the study of the modelling process and the models themselves from one side and the development of methods to enhance the body of knowledge developed for a theory of modelling from the other side. This paper is thus only a first try for a general theory of conceptual modelling.

[2] Good sources to this research are [Chen et al.(1998); Olivé(2007); Thalheim(2000)].

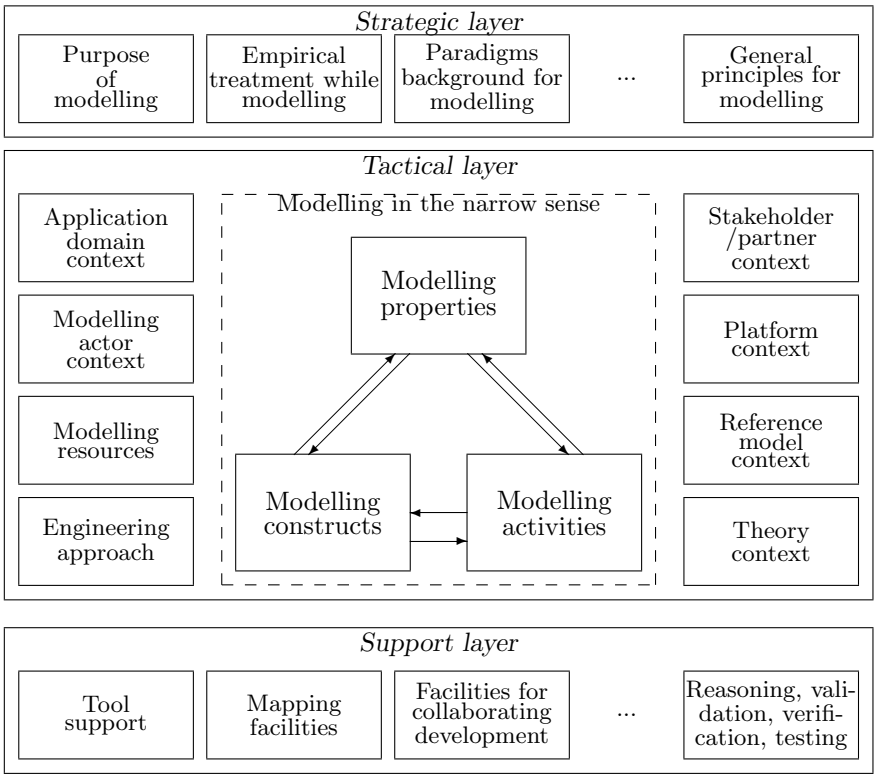


Figure 2: The strategic, tactical and support layers of modelling

2 Towards a Theory of Modelling Constructs

2.1 The Notion of Model in Science Theory and in Information Systems Development

Information system models are typically representations (how specified) of certain application solutions (origin, whereof) for a community of users (whom), for certain application goals and intentions (for what), within a certain time span (when), and with certain restrictions (normal, exception and forbidden cases). Therefore, information systems models are a corrected, rectified, regimented, and in many instance idealised version of the application domain we gain from immediate observation of the application domain. Characteristically, one first eliminates flaws and errors in the application domain and then present the concepts in a 'neat' way. These two steps are commonly referred to as concept reduction and application fitting. Models are thus vehicles for learning about the application. Once we have knowledge about the model, this knowledge is

translated into knowledge about the application domain.

A model represents subjects or things

- based on an *analogy* of structuring, functionality, or behaviour,
- considering certain *application purposes*, and
- providing a simple handling or *service* or consideration of the things under consideration.

The model definition given is one [Stachowiak(1992)] of many options. A model has typically a *model capacity*:

- the model provides some understanding of the original;
- the model provides an explanation of demonstration through auxiliary information and thus makes original subject easier or better to understand;
- the model provides an indication and facilities for making properties viewable;
- the model allows to provide variations and support optimisation;
- the model support verification of hypotheses within a limited scope;
- the model supports construction of technical artifacts;
- the model supports control of things in reality;
- the model allows a replacement of things of reality and acts as a mediating means.

Typically these functions are used simultaneously and in competition. Therefore *to model* means different activities at the same time: to plan or form after a pattern or shape, to make into an organization (as an army, government, or parish), to produce a representation or simulation to model a problem, and to construct or fashion in imitation of a particular model.

This competition of meanings results in a number of *problems* of conceptual modelling such as competing attitudes and profiles of modelers, varieties of styles of specification, multi-model reasoning, and integration into a general coherent model ensemble (or model suite). Therefore we face the *"grand challenge of harmonisation"*.

Models are typically given by the triple (original, image mapping) that is extended by properties of the image, of the mapping, of the system under consideration, that are based on a common modelling "culture" or understanding, and depends on the aim of the model. Therefore we envision that modelling can be considered as an art similar to the 'art of programming'.

2.2 A Theory of Modelling Concepts

Concepts are the basis for conceptual modelling. They specify our knowledge what things are there and what properties things have. They typically represent categories, i.e., sets of abstract or concrete entities that share a set of common properties. Concepts are used in everyday life as a communication vehicle and as a reasoning chunk. Concepts can be based on definitions of different kinds.

Modelling concepts must be represented through representations such as symbols, icons, annotations, ontology units or topics. They are based on *annotations* given in some language. The binding of concepts to their representations cannot be strict. Consider, for instance, the annotation 'bridge' and its dozen of meanings. Typically, the binding between concepts and representations remains to be stable over a longer period of time. It depends however on context within the application domain for certain users or user groups. We therefore bind representations r to their meaning or their concepts $concept(r, g, w)$ within a world w for a group g.

Modelling *judgements*

$$(r, t, (a, m, c), g, w)$$

are elements of certain languages

- for the representation (r)
- of things (t) under consideration,
- with restrictions for their applicability (a), with a rigidity or modality (m), with a confidence (c) on their validity,
- based on a common understanding of a group (g) within their world (w) or culture or application domain context.

These languages are typically well-structured. For instance, representations can be based entirely on the extended ER model [Thalheim(2000)].

The relation among modelling judgements may be based on entailment relations (e.g., material or strict logical implications), contrariety (e.g., exclusion constraints among properties), contradiction (e.g., existence exclusion), and independence (e.g., interaction-free interpretation) beyond the fundamental structural relations discussed below [Albrecht et al.(1998)].

Judgements of models are additionally characterised by their quantity (universal, particular, singular), their quality (affirmative, negative, infinite), their relation (categorical, hypothetical, disjunctive), and their modality (problematical, assertorical, apodictical) [Deppert(2009)]. We may limit our characterisation to modality.

A concept has typically a manyfold of definitions. Their utilisation, exploration and application depend on the user (e.g. the education profile), the usage, and context. Concepts typically also depend on the application context, i.e. the application area and the application schema. The association itself must be characterised by the kind of association. Concepts are typically hierarchically ordered and can thus be layered. We assume that this ordering is strictly hierarchical and the concept space can be depicted by a set of concept trees. A concept is also dependent on the community that prefers this concept. A concept is also

typically given through an embedding into the application domain and into the knowledge space.

The main part of our concept definition is a tree-structured structural expression of the following form

SpecOrderedTree(StructuralTreeExpression
(DefinitionItem, Modality(Sufficiency, Necessity),
Fuzziness, Importance, Rigidity,
Relevance, GraduationWithinExpression, Category))) .

This general understanding allows a number of approaches to modelling such as the styles considered below (e.g., Russian doll, Venetian, ...; see below) or the collection style that collects all quadruples without any additional structuring or layering. If in the application domain things can be categorized we might use this categorisation also for a general skeleton with a model.

We may therefore restrict the *theory of models* to a set of judgements $\{(r, t, (a, m, c), g, w)\}$ and the meaning of these representations $concept(r, g, w)$. Representations as well as concepts are expressions within a language that is appropriate for the model purpose. Consequently we need to define these languages in a flexible and expressive way.

2.3 The Fundamental Structural Relations

The five fundamental structural relations used for construction abstraction are aggregation/participation, generalisation/specialisation, exhibition/characterisation, classification/instantiation, and separation between introduction and utilisation.

Aggregation (agglomeration)/participation characterizing which object consists of which object or resp. which object is part of which object. Aggregation is based on constructors such as sets, lists, multisets, trees, graphs, products etc. It may include naming. *Generalization/specialization* characterizing which object generalizes which object or resp. which object specializes which object. Hierarchies may be defined through different classifications and taxonomies. So, we may have a different hierarchy for each point of view. Hierarchies are built based on inheritance assumptions. So, we may differentiate between generalization and specialization in dependence on whether characterization are not or are inherited and on whether transformation are or are not applicable. *Exhibition/characterization* specifying which object exhibits which object or resp. which object is characterized by which object. Exhibitions may be multi-valued depending of the data type used. They may be qualitative or quantitative. *Classification/instantiation* characterizing which object classifies which object or resp. which object is an instance of which object. *Introduction/utilisation* allows to distinguish between an introduction of an object, the shared or exclusive utilisation and the finalisation of an object.

2.4 Building Principles for Modelling Languages

Models are expressions in a modelling language. The language itself may be build ion principles such as the following ones:

Compositionality supports combination of models. Given any two models $\mathcal{M}_1$, $\mathcal{M}_2$ combined into a complex one $\mathcal{M}_1 \oplus \mathcal{M}_2$ (for any composition operator $\oplus$ of the language syntax), the semantics of $\mathcal{M}_1 \oplus \mathcal{M}_2$ is defined by $Sem(\mathcal{M}_1 \oplus \mathcal{M}_2) = Sem(\mathcal{M}_1) \oplus Sem(\mathcal{M}_2)$.

Inductivity is typically based on inductive construction of expressions in a language. For instance, ER logics separates attribute, entity, relationship, and cluster types and uses also some concept of variables for a layered construction of models.

Separation of characterisation and coexistence scopes the attention either to the properties of an object itself or to the associations of the object (or things) to other objects or things.

Separation of introduction and co-use allows to distinguish for the CRUD (Create-Read-Update-Delete) lifecycle of objects between the CUD features and the R features for the object itself from one side and the co-use (through R) or referencing mechanism that co-use these objects form the other side. Entity types are used to introduce and to structure objects and relationship or cluster types reference and co-use these objects. Reference is typically based on some concept of identification (tuple identifier, key value for one of the keys, identifier as surrogate or artificial construct) [Beeri and Thalheim(1999)].

Context-free expression construction implies the coincidence theorem and allows to limit consideration of language expressions only on the expression itself.

These principles are typically taken for granted in formal languages. They are neither naturally given in an application nor generally achievable within a modelling process. For instance, natural languages use idioms that support clustering and encapsulation, noun compounds that allow to combine nouns into a singleton one, implicit active zones that allow to tighten meaning of constructs, and complex categories with prototype semantics [Schewe and Thalheim(2008)]. From the other side, these principles are very powerful and useful. Inductivity and context-freeness allow to manage model constructs in separate.

Typical pragmatic assumptions applied to conceptual models are the unique name assumption, unique flavour assumption, existence of full-fledged domains, non-triviality of structures of associations, strict hierarchical structures, and non-triviality of identification.

A typical example of principles is the set of principles used for the extended entity-relationship model: inductivity (updates are essentially atomic),

compositionality for any type construction pragmatic assumptions such as the usage of names through noun as standard markers, closed schemata, context-free specifications, canonical semantics (e.g. sets instead of multisets, ...), value-identifiability of objects, and restrictions such as limiting computation to functions of low computational complexity.

2.5 Inductive and Abstraction Layered Typed Modelling Constructs

Typically, a model is defined in a certain language. A model language $\mathcal{L}$ for a model uses some signature $\mathbb{S}$ and a set of constructors $\mathbb{C}$ that allows to build a set of all possible expressions in this language. Typically constructors are defined by structural recursion [Thalheim(2000)]. The set of constructors may allow to build expressions that do not fulfill certain quality or more generally integrity conditions. Therefore we introduce a set $\Sigma_{\mathbb{S},\mathbb{C}}$ well-formedness conditions.

Well-formedness conditions separate 'normal' expressions from 'abnormal' ones. The later can be separated into construction abnormality and semantic abnormality. We may allow such abnormalities that are corrigible to normal expressions. The avoidance of abnormality is still research in progress. Kinds of abnormality that should be handled within a theory of conceptual modelling are pleonasm (e.g., redundancy), semantic clashes (e.g., contradictions), Zeugma (e.g., overloading of constructs, combining separable semantic units into one concept), and improbability (e.g., almost empty classes).

Well-formedness restrictions influence the *modelling style* [Klettke(2007)].

- The *Strong Venetian style* rigidly separates basic constructs and builds a fully compositional structuring. ER schemata and UML class diagrams are typically based on this style.
- The *Weak Venetian style* separates constructs to same degree but not more than it is necessary. Performance-tuned physical relational schemata are typically based on this style.
- The *Strong Russian Doll style* is based on a full expansion of objects, i.e. objects in a database are potentially expandable through navigational substructures.
- The *Weak Russian Doll style* uses a layered representation similar to tree languages.

ER modelling is typically based on the Salami slice style whereas XML modelling typically uses the strong Russian doll (DTD style) or the weak Venetian or weak Russian doll (XML schema) style. The weak Venetian blind style is also the basis for component-based development of models since amalgams constructs as small models of coexisting and coevolving facets of objects.

A model type $\mathcal{T}_{\mathcal{L}_\mathbb{S}} = (\mathcal{L}_\mathbb{S}, \Sigma_{\mathcal{L}_\mathbb{S}})$ is defined by a pair consisting of the language of the model and by constraints $\Sigma_{\mathcal{L}_\mathbb{S}} \in \mathcal{L}(\Sigma_\mathbb{S}^{\text{WellFormed}})$ applicable to all models defined in the given language.

Model languages $\mathcal{L}_{\mathbb{S}_1}, ...\mathcal{L}_{\mathbb{S}_n}$ may be bound to each other by *partial* mappings $\mathbb{R}_{i,j} : \mathcal{L}_{\mathbb{S}_i} \rightarrow \mathcal{L}_{\mathbb{S}_j}$ based on their signatures. These mapping typically define the association of elements among the languages.

A model is based on an expression in the given language. Typically, it has a structure definition, a semantics definition, and a pragmatics definition. Semantics restricts the models we are interested in. Pragmatics restricts the scope of the users of models. We explicitly define a model $\mathcal{M}$ by an expression $struct_\mathcal{M}$ in a language $\mathcal{L}_\mathbb{S}$ that obeys $\Sigma_{\mathcal{L}_\mathbb{S}}$, by a set of constraints $\Sigma_\mathcal{M}$ defined in the logics of this language. Therefore, each model has its model type. We denote by $\mathcal{M}_\mathcal{T}$ or $\mathcal{M}_i$ for some i the set of all models of this type.

2.6 Coexistence of Equivalent Models

Models support a number of purposes such as *construction of systems*, *communication*, *analysis*, *examination and check*, *documentation*, *mastering of application complexity*, *improvement*, *evolution* and *realisation and construction*. For instance, we can distinguish construction models that are product-focussed and business user models that are use-focussed. The last kind also refers to the broader societal context in which the information system is going to be used beside the interaction of the business user with the information system. We might overburden a model in order to satisfy all these purposes. Instead, we better use a number of models for each of its purposes. These models are then bound to each other. The binding may be implicit or explicit. Implicit binding may lead to incoherence. Therefore, we shall better request a *coherent coevolution and coexistence of models*. This coexistence may either be based

- on *model suites* or ensembles that use an explicit association among the constructs of the models or
- on *global and specific models* that are based on a global model which combines all aspects of the specific models and an ensemble of specific models which are views of the global model and which reflect the different purposes.

The first approach seems to be the most appropriate one since a stakeholder needs a model on the appropriate abstraction level that is not overburden by any unnecessary detail. We have been introducing the notion of a model suite for the support of this approach [Thalheim(2009)].

The later approach to model coexistence may be supported by two approaches:

Global model as combined view of the specific models: The global model is constructed from the constructs of the local models. It may not reflect all constructs of the specific models. It has however to reflect all those constructs that are common in at least two of the specific models.

Specific models as views of the global model: The specific model are obtained through filtration (e.g., by application of view construction, aggregation or summarisation functions) from the global model.

2.7 Integration of Static Integrity Constraints

A number of approaches are used for the integration of static integrity constraints into a model.

Built-in semantics: Each constructor in the language has its constraints that are built in.

Parameterised constructs: Constructors in the language are parameterised by attachable constraints. The constructor may be chosen without completion of the constraints. These constraints are considered to be parameters of the constructors that can be instantiated at a later development stage.

Constraint logic: The model $\mathcal{M}$ (or the model type $\mathcal{T}_{\mathcal{L}_s}$) allow an introduction of a specific logic $\mathfrak{L}_{\mathcal{M}}$. Formulas $\Sigma_{\mathcal{M}}$ of this language restrict the instances of this model.

The last approach is the most flexible one but requires a sophisticated logic background. For instance, models can be unsatisfiable. This approach has been chosen for the definition of the notion of the model. The second approach is used in many graphical models such as object-role modelling [Halpin(2009)]. The first approach is used for specific constructors such as the Is-A constructor.

2.8 Restricting Modelling by the Choice of Modelling Languages

Each modelling framework uses a number of modelling languages. Therefore, we are bound to the conceptions of the language, the expressivity of the language, and the methodology for language utilisation. These modelling languages are typically used in a sequential or partial concurrent way. This application of languages layers the development process. Stages (layers) and steps of modelling activities follow one another. They may iterate and can be applied in parallel. They also restrict what quality or capability properties must be satisfied. The design and development quality depends on main success factors: *structuring of the process itself*, *culture of people involved*, *skills of actors*, and *process capabilities*.

Languages may however also restrict modelling. The Sapir-Whorf hypothesis [Whorf(1980)] or the "*principle of linguistic relativity*" postulates that actors skilled in a language may not have a (deep) understanding of some concepts of other languages. This restriction leads to problematic or inadequate models or limits the representation of things and is not well understood.

3 Towards a Theory of Modelling Activities and Acts

Modelling activities are based on modelling acts. Modelling is a specific form and we may thus develop workflows of modelling activities. These workflows are based on work steps [Thalheim(2000)] such as 'decompose' or 'extend', abstraction and refinement acts, validation and verification, equivalences of concepts, transformation techniques, pragmatistic solutions and last but not least the domain-specific solutions and languages given by the application and implementation domains.

3.1 The "Act" of Modelling

Modelling typically means the construction of models which can be used for detection of insights or for presentation of perceptions of systems. Modelling is typically based on languages and thus has a semiotic foundation.

The act of modelling consists of

1. a selection and construction of an appropriate model depending on the task and purpose and depending on the properties we are targeting and the context of the intended system and thus of the language appropriate for the system,
2. a workmanship on the model for detection of additional information about the original and of improved model,
3. an analogy conclusion or other derivations on the model and its relationship to the real world, and
4. a preparation of the model for its use in systems, to future evolution and to change.

The modelling act can be understood as a generalisation of the speech act. We may distinguish a number of activities depending of the subject. For instance, Figure 3 displays the conceptualisation act for application domain gathering, understanding, and modelling. *Conceptualisation* aims to collect objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them. It is thus is an abstract, simplified view of the world that we wish to represent. A similar modelling act may be observed for evaluation. In this case, the resource dimension will be the the evaluation background instead of the application domain.

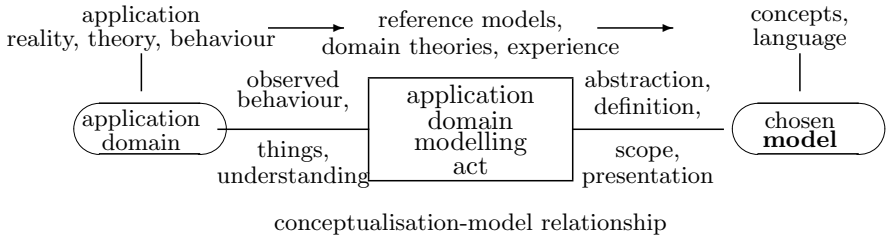


Figure 3: Conceptualisation dimensions of the application domain modelling act

The act of modelling is based on an *activity* that is characterised by the *work products*, the *aspects* under consideration (scope), the *resources* used in an activity, and the *partners* involved into the activity. Additionally we might extend this characterisation by activity goals and intentions (for what), time span (when), and restrictions (normal, exception and forbidden cases) or obligations for later activities.

We may distinguish a number of activities and acts, e.g. the following ones:

Understand: The understanding act support reasoning within the application domain. It results typically in drafts that can be used for development of conceptualisations. The problems and possible solutions are comprehended. Conjectures are drawn.

Conceptualise: The conceptualization act aims in formalising the part of the application domain that is of interest. We form a number of concepts of of the application domain and represent those formally within the concept language. These concepts are conceptually interpreted in the application domain.

Abstract: The abstraction act aims in outlining main problems that must be supported by the information system. It generalises these problems and abstracts from unnecessary details on the basis of forgetful mappings.

Define: The definition act is used to unambiguously specify, to delineate, and to delimit main concepts or annotations used for the development of the model. Definitions might be given in a variety of forms. We can use also different languages and target on visualisation of concepts.

Construct: The construction act is often considered to be the main act during modelling. It aims in creating a model by organizing and linking ideas, judgements, or concepts. It may include the sub-act of rebuilding, i.e., reconstructing, framing up, and customising. When we talk about anticipated behaviour it includes activities of conjecturing and hypothesising.

Refine: The refinement act is a basic act of iterative development. The model itself becomes enriched, more elaborated or sophisticated while preserving its main structures and behaviour. It matches thus in a better, more precise manner the needs of the application. The refinement act is typically based on some evaluation or assessment and on analysis for improvement potential. Refinement uses scoping for restricting changes to a necessary extent.

Evaluate: The evaluation act is based on a set of quality characteristics that we have agreed to satisfy in advance. These quality characteristics are typically given in an abstract form and are not solely based on metrics. Evaluation is typically applied to a model or parts of it. It results in determination of the value of the judgements under evaluation.

3.2 Principles of Understanding

Principles of understanding are not well developed in the information systems modelling field. Understanding is based on *judgements* and *believes*. We have to figure out why the part of the application domain is under consideration. We need to scope the part of the application domain. Finally, if we already know the modelling target and how to approach it, then we need to be consistent within the model.

Understanding also aims in becoming knowledgeable in the application domain and to develop a *sense* and sensibility for the application tasks. This includes an understanding of evolution within the application domain, of context for the application problems, of causal mechanisms within this domain, of basic units thereof, of structure and functioning, of levels and forms of organisation, of information flow within the application domain, and of stability with this application.

Application problems are driven by complex and deep *motivations* that must be understood in order to support their solution. They are limited in their capabilities and are shaped by past experience and approaches. They are using solutions that might be based on faulty logic and decision processes. Also a number of preferences has been selected in the past. We need to capture the *potential* of the new solutions.

A number of theories might be used for better understanding the application domain problems and the potential of solutions: Attribution theory, constructivism, focusing effect, framing, just-world phenomenon, objectification, organisational structures, and life case and story models.

3.3 Principles of Conceptualising

Principles of conceptualisation generalise the seven principles of Universal Design [Patil et al.(2003)]. We may differentiate between mandatory and optional

principles. These principles are best reflected by requirements to the model and to the judgements. Their realisation is an open research issue.

Typical mandatory principles are usefulness, flexibility, simplicity, realisability, and rationality. The model should *useful* and marketable to people with diverse abilities. It provide the same means of use for all users, i.e., identical whenever possible and equivalent when not. It includes certain views that support quality properties such as privacy, security, and safety. The model must be *flexible* in use and accommodates a wide range of individual preferences and abilities. It facilitate different viewpoints depending on the stakeholder and depending on a wide range of individual preferences and abilities. The model must be simple and *intuitive in use*. The judgements are easy to understand, regardless of the user's experience, knowledge, language skills, or current concentration level. A side-principle is the elimination of unnecessary complexity. Models must be consistent with user expectations and intuition, must accommodate a wide range of stakeholder skills, should arrange judgements consistent with its importance. The model can be *used efficiently* and comfortably and with a minimum of fatigue. It does not require additional reasoning capabilities or skills from its user. The model must be *checkable* within a validation, verification or testing procedure depending on the model's purpose. Its adequateness for the purpose must be checkable and improvable.

Optional conceptualisation principles are perceptability, error-proneness, and parsimony. The model must be *perceptible*. It communicates necessary information effectively to the stakeholder, regardless of ambient conditions or the stakeholder's abilities. It may use different representations depending on its use by the stakeholder, may be redundant presentation of essential information, and should allow to enlighten different viewpoints. The model does not allow unintended uses and is *error-prone*. It minimizes hazards and the adverse consequences of accidental or unintended actions. Elements that may be interpreted in unintended ways are avoided. The model can be surveyed without wrong interpretations. The model uses in an appropriate way space and resources. It is *parsimonious* both at the systems and at the stakeholder's side. It provide a clear line of sight to important elements for any usage. It accommodate variations in hand and grip size and is well-supported.

3.4 Principles of Abstraction

The development of a model is *the* result of modelling. It relates things $\mathcal{D}$ under consideration with concepts $\mathcal{C}$. This relationship $\mathcal{R}$ is characterised by restrictions ρ to its applicability, by a modality θ or rigidity of the relationship, and by the confidence Ψ in the relationship. The model is agreed within a group $\mathcal{G}$ and valid in a certain world $\mathcal{W}$. Stachowiak [Stachowiak(1992)] defined three characteristic properties of models: the *mapping* property (have an original),

truncation property (the model lacks some of the ascriptions made to the original), and *pragmatic* property (the model use is only justified for particular model users, tools of investigation, and period of time). We can additionally consider the *extension* property. The property allows that models represent judgments which are not observed for the originals. In computing, for example, it is often important to use executable models. Finally, the *distortion* property is often used for improving the physical world or for inclusion of visions of better reality.

These principles result typically result in *forgetful mappings* from the origin to the model.

3.5 Principles of Definition

Defining is an act of determining, especially the logical meaning. A definition is either a statement expressing the essential nature of something or a statement of the meaning of a thing The result meaning of definition the action or the power of describing, explaining, or making definite and clear Definitions may use the clarity of visual presentation. They provide a sharp demarcation of outlines or limits.

The *definition act* is a formal passage describing the meaning of a concept or annotation. The concept to be defined is the definiendum. A concept may have many subtly different meanings and may be defined in a variety of ways. For each such meaning, a definiens is an expression in a certain language that defines that specific meaning of the concept or annotation.

We distinguish six different kinds of definitions. Typically, modelling uses distinct kinds of definitions in a combination. We thus may compose a definition by selection of definition parameters that are set by the hexagon in Figure 4.

The *real* or matter definition refines generic terms (genus proximum) by kind generating properties (differentia specifica). Specific kinds are genetic definitions and declarations. The *nominal* or stipulative definition is a type of definition in which a new or currently-existing concept or annotation is given a new meaning for the purposes of modelling in a given context. Specific kinds are precising definitions that narrow down the set of things meeting the definition. The *assignment* or attribution definition determines the relation among already defined concept or annotation. Specific kinds of such definitions are *referential* and *association* definitions that directly relate the concept or annotation to already existing ones. The *inductive* definition uses an initial concept or annotation, a set of constructors and a closure condition and typically defines a set of concept or annotation each of which is then considered to be a singleton one. The *axiomatic* definition uses a set of axioms and implicitly defines the definiens. The *direct* or explicit definition clearly separates definiens and definiendum. The *indirect* or implicit definition does not use such explicit separation. A definition may be *complete* or *partial*. Furthermore definitions may provide *syntactical*,

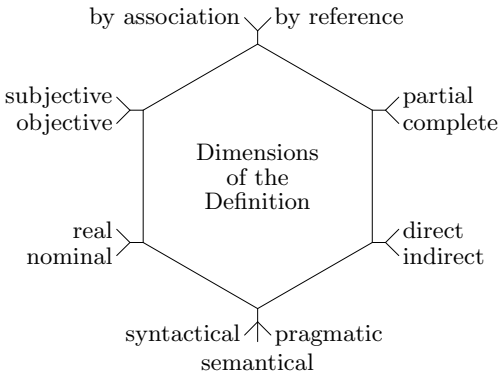


Figure 4: Ingredients for selection of the style and of the formulation used of stating a definition

semantical, or *pragmatical* issues. Intensional definitions are a specific kind of definitions. They give the meaning of a concept or annotation by specifying all the necessary and sufficient conditions for belonging to the collection of objects being defined.

We might use during a definition act also other forms of definitions beside these 'precise' definitions. The *lexical* definition of a of a concept or annotation is the meaning of the concept or annotation in common usage. An *extensional* definition of a concept or annotation formulates its meaning by specifying its extension, that is, every object that falls under the definition of the concept or annotation in question.

3.6 Principles of Construction

Construction principles have mainly been developed for construction methods such as IDEF or specific paradigms such as object-oriented software construction. They might however be developed in a more general setting. We know a number of requirements to construction: right orientation and methodology, expectation orientation, user and usage focusing, realisability, satisfaction of main quality criteria such as safety, novel and creative decisions, and team orientation both for development and for deployment.

These requirements directly lead a number of engineering principles that are applicable to construction of models. *Engineering* is oriented towards encapsulation of of experiences with design problems pared down to a manageable scale. *Real-life engineering* is full of uncertainties and risks, impossible to replicate effectively in a formalised way in a text. An *engineering component* means any

engineering structure, which may be constructed from several interconnected elements into a single entity. *Effectively* withstanding loads is defined as the capacity to accept service loads without exceeding either the specified maximum stress, specified maximum deflection, of both of this specifications. *Service loads* are those loads, specified or unspecified, that the designer considers as creditable to be imposed on the component during its service life. Engineering may be performed in a systematic way based on conceptual modelling *methodologies*.

Therefore conceptual modelling inherits most principles of engineering. *Modularity* is probably the most fundamental principle of good engineering design. A large system should be analysed into smaller subsystems. Modularisation may either be explicit based on an architecture or implicit based on a name space. Systems should be constructed through *subsystems*. A *hierarchical ordering* should be specified so that modules are divided into layers, where each layer may interact with the layers just above and below, but not with distant layers. Therefore it is often useful to define system *layers* explicitly.

Engineering is based on a system architecture. We might distinguish a number of different architectural views such as the *component architecture* based on modules, the *application architecture* based on views depending on tasks under consideration, the *infrastructure architecture* based on embeddings into supporting systems, *evaluation architectures* depending on the purposes of the model (communication, construction, detection of solutions, exploration of the application domain, etc.), and *interface architectures* for embedded information systems.

Typically, information systems use a meta-structure called *skeleton* that either explicates the component structure with associations on the basis of views or interface for the component association or separates different concerns in the application from each other. Therefore, top-down development may start with drafting a skeleton and refining elements step by step. Similarly, bottom-up or inside-out development may use the skeleton for zooming-out unessential component details.

There are specific principles that are based on paradigms for *object-relational information systems construction* such as the Liskov-substitution-principle, the open-closed-principle, the dependency-inversion-principle, the interface-segregation-principle, the reuse-release-equivalence-principle, the common-closure-principle, the common-reuse-principle, the acyclic-dependencies-principle, the stable-dependencies-principle, the stable-abstractions-principle. and the law of Demeter. There are other principles that might also be of interest such as the linguistic-modular-units-principle, the few-interfaces-principle, the small-interfaces-principle, the explicit-interfaces-principle, and the information-hiding-principle. Most of these principles are well-known for software construction and can be used in the same way for conceptual modelling.

3.7 Principles of Refinement

Refinement relations are the key to formalise modelling activities and the modelling process. Classically [Broy(1997)], refinement methods are separated into

- *property refinement* - enhancing requirements - allows us to add properties to a specification,
- *glass box refinement* - designing implementations - allows us to decompose a component into a distributed system or to give a state transition description for a component specification, and
- *interaction refinement* - relating levels of abstraction - allows us to change the granularity of the interaction, the number and types of the channels of a component.

These notions of refinement are sufficient to describe all activities needed in the idealistic view of a strict top down hierarchical system development.

The theory of conceptual modelling may also be used for a selection and development of an assembly of modelling styles. Typical well-known refinement styles [Thalheim(2000)] for structure specification[3] are the following ones:

Inside-out refinement: Inside-out refinement uses the given specification for extending it by additional part. These parts are hocked onto the current specification without changing it.

Top-down refinement: Top-down refinement uses decomposition of functions in the vocabulary and refinement of rules. Additionally, the specification may be extended by functions and rules that have not yet been considered.

Bottom-up refinement: Bottom-up refinement uses composition and generalisation of functions and of rules to more general or complex ones. Bottom-up refinement also uses generation of new functions and rules that have not yet been considered.

Modular refinement: Modular refinement is based on parqueting of applications and separation of concern. Refinement is only applied to one module and does not affect others. Modules may also be decomposed. Modules are typically associated through a skeleton that reflects the application architecture or the technical architecture.

Mixed skeleton-driven refinement: Mixed refinement is a combination of refinement techniques. It uses a skeleton of the application or a draft of the architecture. This draft is used for deriving plans for refinement. Each component

[3] A theory of refinement for functionality, control or interface specification is still an open research issue.

or module is developed on its own based on top-down or bottom-up refinement.

These different kinds of refinement styles allow one to derive *plans* for refinement and and *primitives* for refinement.

3.8 Principles of Evaluation

Evaluation and assessment of quality depends on the purposes of the model. Figure 1 shows that rather different quality criteria apply to a model depending on the goal of the model. Assessment is not targeting the quality of the modelling language[4]. It cannot be the final goal of a modelling act to develop a most correct or a most adequate model.

Evaluation assessment is one of the research issues in software development. There are many quality criteria that might be applied in different stages and life cycles [Jaakkola and Thalheim(2005)]. Most of these ISO/IEC 9126 or 25010, SPICE or CMMI standards introduce a large variety of quality characteristics and propose to measure quality fulfillment on the basis of metrics. Since judgements cannot be of equal rights these metrics fail for conceptual modelling.

A systematic way to quality-driven development is introduced in the *Quality Attribute Driven Software Design Method* by [Bass et al.(2003)] . This approach is based on the use of quality scenarios, in which a stimulus activates the operations specified by the selected quality tactics (operations). The execution of the tactics will cause an expected response in the system to meet the requirements of a quality attribute.

We might evaluate quality of the model or of the modelling acts. The first evaluation direction is easier to access and can be used for evaluation of quality assessment as already discussed in Section 1:

Development quality acts: Models, representations and judgements are subject to revision during development. The impact of each revision must be trackable for and understandable by the developer. Therefore, quality improvement acts include activities supporting analysability, changeability, stability, and testability of models. These activities must be pervasive and ubiquitous. At the same time, these activities must be efficient (e.g. use the resources in an appropriate way, be parsimonious). Since these tasks cannot be performed by a singleton person development acts are based on collaboration of developers with stakeholder and among developers. Each partner in a development process reflects his/her view on the application and on the system. Therefore, a tool support becomes essential.

[4] The model language is also characterised by its expressive power and thus might support or hurt the model (size of the model, artificial constructs, independence of constructs used, reasoning capabilities, correctness of transformations, model size, etc.). Additionally, the representation language might also introduce difficulties.

Developers require that models must be easy to maintain, to analyse, to change, to test and must be stable. Therefore, a specific quality activity is the localisation of stable, evolving and unmature model part.s Developers and coders are interested in facilities that support efficiency (time behaviour, resource utilization, efficiency compliance, scalability). Therefore, model languages must also be extended by performance improvement acts.

Internal quality acts: We may distinguish between functionality and general quality characteristics for internal quality. Functionality includes accuracy, suitability, interoperability, security, flexibility and self-contained/independence. The co-design approach to development includes development of structures and functionality. Therefore, it also includes acts that improve these quality characteristics.
Information systems must also be accurate, suitable, robust, self-contained, independent and internally parsimonious. Additionally they must be portable (adaptable, installalable, replaceable, deployable, transferable, reusable, ...) and reliable (mature, fault tolerant, recoverable, ...). Therefore, quality assurance acts are folded into refinement activities. Classical information system models do neither provide measures and reasoning facilities for this second kind of internal quality nor have appropriate transformation techniques that improve quality.

Quality in use acts: Domain application engineers highly rate documentation activities that improve direct understandability and surveyability. User parsimony is an issue for business users. At the opposite side, developers are interested in evaluation of the model for extensibility, appropriateness, and operability. Quality in use deeply depends on the quality of representation. Therefore, quality improvement acts directly influence representation. Business user request availability, usability compliance and configuratability for their information systems applications. Therefore, quality improvement acts also use activities that scope applications to users' needs.

Assessment of model quality is defined as an open process that encompasses the following principles:

- It is mission-focused, at both the institutional and programmatic levels.
- It is systematic, iterative, collaborative, documented, and adaptable.
- It applies multiple measures, both qualitative and quantitative.
- It identifies strengths and areas that warrant improvement.
- It informs planning and decision-making for the purpose of ascertaining learning and development, thereby improving programs, services, functions, performance, and the overall value of the educational experience.

Our framework is currently based on a seven-level model. The *specification level* is used for a description of quality depending on the purposes of the model and its main quality characteristics. The description consists of a specification of the quality property, the measurement, and the policies for evaluation. It can be extended by specific policies for various development methods such as agile development, by transformations of quality properties into others, and by associations among quality properties. Finally, we may derive constraints for the application of the quality property. The *control* or *technical level* deals with the application of the quality model. It provides guidance for the control procedures such as setting the control management, deriving the scope of control, definition of the control tasks and its actors. The application of the quality framework is based on a quality property portfolio. The portfolio consists of tasks and the necessary supporting instruments. They generalize portfolio known in project management. The *application* or *technology level* handles the management of quality evaluation within software etc. projects based on the technology of development. The *establishment* or *organizational level* is based on a methodology and may be supported by a quality maintenance system. The *value* or *prediction level* provides facilities for handling satisfaction of quality properties and for predicting changes in satisfaction whenever software evolves. The *optimisation level* integrates quality management into the optimisation of the software development process. The *maturity level* uses experiences gained for the innovation and adaptation of other processes and products that have not yet reached this maturity.

3.9 General Principles

Conceptual modelling acts are typically well organised. Therefore we may observe a number of general principles. The *conceptualization principle* restricts development of models to exclusively conceptual aspects of the application domain. The *95% -principle* requires that almost all relevant aspects of the application domain should be described in the conceptual schema. The *formalization principle* explicitly requires a potential formalisation of models and thus guarantees existence of a realisation. The *semiotic principle* restricts models to those that are easily interpretable and understandable. The *correspondence condition for representation* requires that the modellens should be such that the recognizable constituents of it have a one-to-one correspondence to the relevant constituents of the modellum. The *invariance principle* restricts models to those things in the application domain that are invariant during certain time periods within the application area. The *sub-schemata principle* explicitly bases modelling on skeletons.

In the case of multi-layered modelling acts we may derive a number of additional principles. The *downward-dependency principle* requests that the main

data dependency structure is top-down. Objects at a higher level depend on objects at a lower level. The *upward-notification principle* restricts objects at a higher level to act as subscribers to database changes at lower level. They may decide whether they eagerly or lazily enforce observed changes at lower level. Objects at lower level report however their changes. The *neighbor-communication principle* restricts object data exchange data to those objects at the same layer. The neighborhood may also require that neighboring databases should be synchronised. The *explicit association principle* request that the data exchange between database components is explicitly documented and recorded. Whenever a database at a higher level perceives data from a lower level then this exchange is logged. The *cycle elimination principle* brakes cyclic data exchange between layers based on the log information. The *layer naming principle* requires that data can be identified at their level.

3.10 Methods of Conceptual Modelling

The theory of conceptual modelling is based on a small number of methods. The main methods are *abstraction, modularisation, inheritance, generalisation/refinement, transformations, selection and application of modelling styles,* and *separation of concern*. Abstraction and refinement are well understood. Modularisation is based on an architectural decomposition of a large model into components and a development of a linking or binding scheme for the separated components. Typically, conceptual modelling only considers one transformation technique for the mapping among layers, e.g. from conceptual to logical schemata. The mapping technique and the mapping rules may however vary depending on the goals. Separation of concern allows to provide a clear understanding of parts of the application.

We are not going to develop a theory of these methods in depth although it is necessary for a theory of conceptual modelling. Abstraction is discussed in [Thalheim(2000)]. Modularisation and inheritance has found a deep foundation for programming languages. Refinement is still an open research issue. Transformations, modelling styles and pattern are dependent on languages chosen for specification.

4 Towards a Theory of Modelling Properties

4.1 Properties describing the Purpose of Conceptual Modelling

Models are developed with different goals, different scope, within different context, with different appeal to the receiver of the model, with different granularity, with different background, and with different realisation forms. Therefore we have to explicitly handle modelling purpose properties.

The *mission* of modelling is described by scope of the model, the users community, the tasks the model might support, the major and minor purposes satisfied by the model and the benefits obtained from the model for the given user community. The *goals* of a model are based on the impact of the model, restricted by the relationships among users and their roles they are playing. The *brand* of the model is given by the who-what-whom-action pattern. The *meta-model* can be used to provide information about the model such as the context of the model, the context in which the model might be useful to the auditory, the usage of the model, and the restrictions of the model.

It surprises that these model properties are not explicitly handled in most modelling approaches. The same surprise can be observed for a declaration of the main goals of the modelling act such as

construct a model, a part of the model, a concept or a judgement, etc. (describe, delineate, fabricate, master),
communicate the judgements, the observations, the concepts, etc. (explain, express, verbalise or display),
understand the application domain, the system opportunities, etc. (cognise, identify, recognise, percept),
discover the problems, the potential, the solutions, etc. (interact, identify),
indicate properties of importance, relevance, significance, etc. (visualise, measure, suggest, inform),
variate and *optimise* a solution, a judgement, a concept, a representation depending on some criteria,
verify or *validate* or *test* a model, a solution, a judgement, a representation or parts of those,
control the scope of modelling, the styles or pattern, parts of a model, judgements, etc. (rule, govern, proofread, confirm, restrain, administer, arrange, stratify, standardise),
alternate or *compensate* or *replace* or *substitute* or *surrogate* models or parts of them, judgements, concepts, etc. (transfer, reassign, evolve, migrate, balance, correct, novate, truncate, ersatz).

The first and last four goals lead to a *datalogical* model that is structured according to technology. The other goals result in an *infological* model that is delivered to the needs of the user. We thus use a different frame of reference. The application of the results may thus be descriptive or prescriptive, constitutive or prognosticating, categorical or exegetic or contemplative or formulaic.

4.2 Properties of the Modelling Process

The modelling process can be characterised by a number of (ideal) properties:

Monotonicity: The modelling process is monotone if any change to be applied to one specification leads to a refinement. It thus reflects requirements in a better form.

Incrementality: A modelling process is iterative or incremental if any step applied to a specification is only based on new requirements or obligations and on the current specification.

Finiteness: The modelling process is finite if any quality criteria can be checked in finite time applying a finite number of checks.

Application domain consistency: Any specification developed corresponds to the requirements and the obligations of the application domain. The appropriateness can be validated in the application domain.

Conservativeness: A modelling process is conservative if any model revision that cannot be reflected already in the current specification is entirely based on changes in the requirements.

Typical matured modelling processes are at least conservative and application domain consistent. Any finite modelling process can be transformed into a process that is application domain consistent. The inversion is not valid but depends on quality criteria we apply additionally. If the modelling process is application domain consistent then it can be transformed in an incremental one if we can extract such area of change in which consistency must be enforced.

4.3 The Implicitly Assumed Modelling Context Properties

Modelling is inherently incomplete, biased and ruled by scoping by the initiators of a project, by restricting attention to parts of the application that are currently under consideration and ruling out any part of the application that will never be under consideration. This intentional restriction is typically not communicated and directly results in the "modelling gap" [Kaschek(2003)]. Additionally, modelling culture results in different models and different understandings.

Incompleteness of specifications is caused by incomplete knowledge currently available, incomplete coverage of the specification or by inability to represent the knowledge in the application. Incompleteness may be considered as the main source of the modelling gap beside culture and skills of modelers. The application is either partially known, or only partially specified, or cannot be properly specified. This incompleteness leads to the modelling gap displayed in Figure 5.

To overcome the problems of specifications we may either use

– *negated specifications* that specify those cases which are not valid for the application,

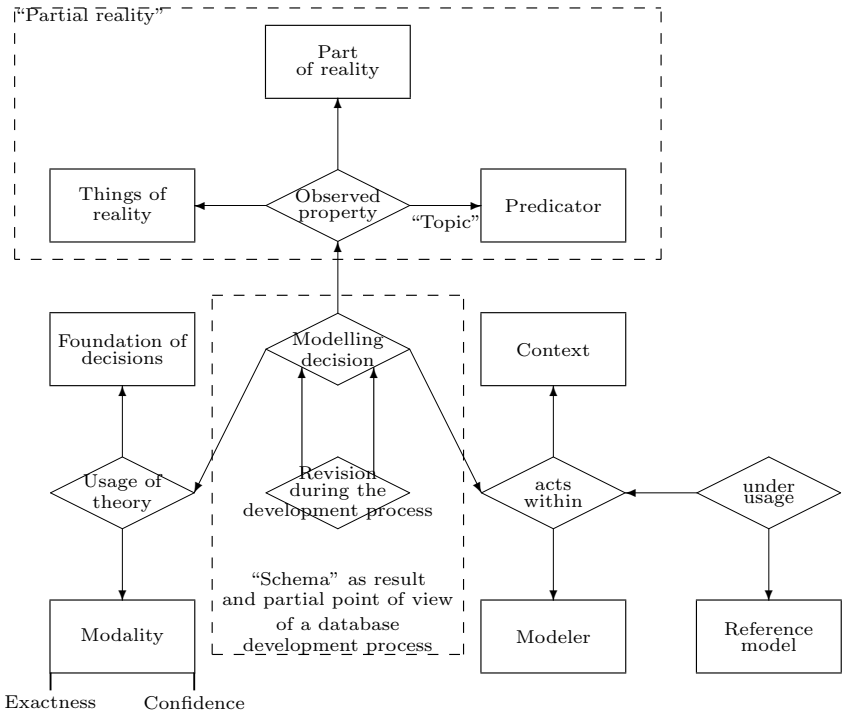


Figure 5: Modelling decisions made in the database development process

- *robust specifications* that cover the main cases of the applications, or
- *approximative specifications* that cover the application on the basis of control parameters and abstract from order parameters.

5 Conclusion

5.1 Goals of this Paper

Modelling and especially conceptual modelling is not yet well understood and misinterpreted in a variety of ways. The first goal of the paper is to overcome some myths of conceptual modelling such as:

1. Modelling equals documentation.
2. You can think everything through from the start.
3. Modelling implies a heavyweight software process.
4. You must "freeze" requirements and then you can start with modelling.

5. Your model is carved in stone and changes only from time to time.
6. You must use a CASE tool.
7. Modelling is a waste of time.
8. The world revolves around data modelling.
9. All developers know how to model.
10. Modelling is independent on the language.

The second goal of this paper is the development of a framework to modelling. Modelling is based on an explicit choice of languages, on application of restrictions, on negotiation and on methodologies. Languages are defined through their syntactics, their semantics, and their pragmatics. We typically prefer inductive expression formation based on alphabets and behaviour defined on expressions. Restrictions depend on logics (deontic, epistemic, modal, belief, preferences) and use shortcuts, ambiguities, and ellipses. Negotiation support management or resolution of conflicts and the development of strategies to overcome these strategic, psychological, legal, and structural barriers. Development methodology are based on pragmatism and on paradigms. Since modelling is an activity that involve a number of actors the choice of languages becomes essential. Modelling is a process and based on modelling acts. These modelling acts are dependent from the purpose of modelling itself. Therefore we can distinguish different modelling acts such as understand, define, conceptualise, communicate, abstract, construct, refine, and evaluate. Depending on the purpose of model development we might use modelling act such as construct and evaluate as primary acts.

The third goal of this paper is to draw attention to explicit consideration of modelling properties both for the models themselves and for the modelling acts. This side of conceptual modelling is often only considered in an implicit form. The modelling process is governed by goals and purposes. Therefore, we must use different models such as a construction model, a communication model or a discussion model. Modelling is restricted by the application context, the actor context, the system context and the theory and experience context. These kinds of context restrict the model and the modelling process.

5.2 The Theory Framework

The aim of the paper has not been to develop a complete theory of conceptual modelling. Instead we aimed at the development of a programme for the theory. We described the general purpose of this theory, demonstrated how different paradigms can be selected, and showed which scope, modelling acts, modelling methods, modelling goals and modelling properties might be chosen for this theory.

5.3 Towards Modelling Principles

A theory of conceptual modelling can be based on a system of guiding principles. This paper shows that at least three guiding principles must be explored in detail:

Internal principles are based on a set of ground entities and ground processes.

Bridge principles explain the results of conceptual modelling in the context of their usage, for instance for explanation, verification/validation, and prognosis.

Engineering principles provide a framework for mastering the modelling process, for reasoning on the quality of a model, and for termination of a modelling process within a certain level of disturbance tolerance (error, incompleteness, open issues to be settled later, evolution).

Information systems modelling principles are specialisations of design principles [Denning(2007)]. They are conventions for planning and building correct, fast or high performance, fault tolerant, and fit information systems. Conceptual modelling is based on architecture of a system into components, uses their interactions, and pictures their layout. Modelling is the process of producing models. It is thus adapted from engineering and may thus use the separation of activities into requirements, specification, development, and testing.

Depending on the purpose of the model several quality criteria may be preferred. For instance, construction models should fulfill criteria for good models such as correctness of models, refinement to highly effective systems, fault tolerance of systems, ubiquity of systems, and fitness of systems.

Modelling principles are not laws of nature, but rather conventions that have been developed by modellers to the most success when pursuing quality properties. Therefore, various sets of principles might be developed depending on the community. For instance, modelling based on extended ER models is based on compositionality, incrementality, structure-orientation, and conservativeness. Modelling principles for sets of models such as UML are far more difficult to develop and to maintain.

5.4 Future Work

The programme requires far more work. The theory needs a variable taxonomy that allows a specialisation to languages chosen for a given application domain, must be based on a mathematical framework that allows to prove properties, must be flexible for coping with various modelling methodologies, must provide an understanding of the engineering of modelling, and finally should be supported by a meta-CASE tool that combines existing CASE to to a supporting workbench.

The programme aiming in the development of a general modelling theory becomes more crucial since model-driven approaches are going to be applied to practice and since *modelling is going to be the programming of the future*[5].

The theory of conceptual models is developed in [Thalheim(2010)]. It is based on a theory of languages that are used for concepttual modelling, on a notion of a conceptual model and concepts deployed for the model, on an explicit treatment of the information exchange between the stakeholders that are involved into conceptual modelling, on language-based mapping between an original and the model, and on postulates of conceptual modelling.

5.5 Towards an Agenda for Research

Finally we derive a number of open research fields for a theory of conceptual modelling:

Adjustable selection of principles depending on modelling goals: Since models satisfy different needs and purposes we should be able to unerringly and purposefully select target-aimed principles, to adjust models and modelling acts to these principles and to govern the process of modelling by appropriate properties.

Model suites with explicit model association: Since different application areas, different participating stakeholders, different modelling cultures and different modelling theories and experience result in a large variety of languages and a "Babylonian language confusion and muddle", novel methods for model coexistence, for development of views on the same general model and for model management become more important.

Development of a language culture: Many languages and standards have been developed without insight into theories and without consideration of achievements of research in the past. The knowledge or culture seems to vanish. Holistic compilations of modelling culture and handbooks of conceptual modelling might a starting point if they find their way into teaching and research.

Models 2.0: Models are evolving and maturing. Although this evolution is well reported in papers or books old variants of models are still taken as the starting point without consideration of improvements. These results are however scattered and not compiled or collected. They are neither integrated into the body of knowledge obtained so far nor evaluated for their appropriateness.

[5] Programming languages have been developed from first generation languages to fourth generation languages. The next generation of programming languages is going to be based on models that are used for transformation to programs. The claim in [Embley et al.(2010)] that programming is actually "conceptual-model programming" is the first starting point.

The evolution, progress and maturation should however be taken into account whenever a variant of the model is used. Therefore, a task of a science community is to garden models and to provide support for any newcomer.

Explicit treatment of model value: Model results leave a narrow gap to complete models. The model itself has a value according to the goal, maturity, and usage. The value depends on whether a models is used as an artifact, used for construction, used for negotiation or contracting among stakeholders, used for documentation or used for services and continuous evolution. The development effort for development of a model should match its value.

Coexistence of theory, languages, and tools: Modelling languages are often exclusively syntax-defined, often do not have an adequate theoretical foundation, overestimate the value of sophisticated graphical notations, and do have only partial tool support. Instead we need well-grounded languages with at least both syntax and semantics, with a variety of representations and with tool support that allow customisation to the needs of modellers.

Adequate representation variants of models: Models must be easy to learn and to understand. Users from any community should easily develop a familiarity with the representations. Domain-specific representations and consistency with user expectations do not distract users from the content of the model. Standard meanings support pragmatical treatment of models. Robustness and error-proneness allow the user to concentrate on the essential elements of the model and to abstract from the unessential shapes etc. of visual or textual elements.

Compiler development for models: Models are becoming omnipresent and omnipotent. Application engineers will be able to develop their models. These models are already 'programs' at a higher level. They are currently mainly interpreted and will become compiled to executable code in the future.

Model families and variants: Since modelling must support different goals and purposes models should be adaptable to those purposes and should be extensible in dependence on changing purposes.

Acknowledgement

This work is not yet complete and was a matter of discussions with colleagues or within workshops over a longer period of time. I am deeply indebted to Klaus-Dieter Schewe and Roland Kaschek for their interest, discussions, challenges, common work on this topic, etc. I am indebted to many of my colleagues to support me with their comments on this work, especially Bernd Mahr and Ulrich

Frank. I would like to acknowledge the discussions at the workshops Modellierung 2006 and Modellierung 2009. I want to express my gratitude to IIftC for inspiration and (re-)questioning this approach.

References

[Albrecht et al.(1998)] Albrecht, M., Buchholz, E., Düsterhöft, A., Thalheim, B.: "An informal and efficient approach for obtaining semantic constraints using sample data and natural language processing"; Proc. Semantics in Databases, LNCS 1358; 1–28; Springer, Berlin, 1998.

[Bass et al.(2003)] Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice; Addison-Wesley, 2003.

[Beeri and Thalheim(1999)] Beeri, C., Thalheim, B.: "Identification as a primitive of database models"; Proc. FoMLaDO'98; 19–36; Kluwer, London, 1999.

[Bjørner(2009)] Bjørner, D.: Domain engineering; volume 4 of COE Research Monographs; Japan Advanced Institute of Science and Technolgy Press, Ishikawa, 2009.

[Broy(1997)] Broy, M.: "Compositional refinement of interactive systems"; Journal of the ACM; 44 (1997), 6, 850–891.

[Chen et al.(1998)] Chen, P. P., Akoka, J., Kangassalo, H., Thalheim, B., eds.: Conceptual modeling: current issues and future directions; LNCS 1565; Springer, 1998.

[Denning(2007)] Denning, P.: "Great principles of computing"; http://cs.gmu.edu/pjd/GP/ (2007).

[Deppert(2009)] Deppert, W.: "Theorie der Wissenschaft"; Christian-Albrechts-University at Kiel, Lecture notes for academic year 2008/09, http://wolfgang.deppert.de (2009).

[Embley et al.(2010)] Embley, D. W., Liddle, S. W., Pastor, O.: "Conceptual-model programming: A manifesto"; The Handbook of Conceptual Modeling: Its Usage and Its Challenges; chapter 1, 1–15; Springer, Berlin, 2010.

[Halpin(2009)] Halpin, T. A.: "Object-role modeling"; Encyclopedia of Database Systems; 1941–1946; Springer US, 2009.

[Hevner et al.(2004)] Hevner, A., March, S., Park, J., Ram, S.: "Design science in information systems research"; MIS Quaterly; 28 (2004), 1, 75–105.

[Jaakkola and Thalheim(2005)] Jaakkola, H., Thalheim, B.: “Software quality and life cycles”; ADBIS’05; 208– 220; Springer, Tallinn, 2005.

[Kangassalo(2007)] Kangassalo, H.: “Approaches to the active conceptual modelling of learning”; Active Conceptual Modeling of Learning; LNCS 4512; 168–193; Springer, Berlin, 2007.

[Kaschek(2003)] Kaschek, R.: Konzeptionelle Modellierung; Ph.D. thesis; Advanced PhD (Habilitation Thesis); University Klagenfurt (2003).

[Klettke(2007)] Klettke, M.: Modellierung, Bewertung und Evolution von XML-Dokumentkollektionen; Advanced PhD (Habilitation Thesis); Rostock University, Faculty for Computer Science and Electronics (2007).

[Mittelstraß(2004)] Mittelstraß, J., ed.: Enzyklopädie Philosophie und Wissenschaftstheorie; J.B. Metzler, Stuttgart, 2004.

[Olivé(2007)] Olivé, A.: Conceptual modeling of information systems; Springer, Berlin, 2007.

[Patil et al.(2003)] Patil, B., Maetzel, K., Neuhold, E. J.: “Design of international textual languages: A universal design framework”; IWIPS; 41–56; Product & Systems Internationalisation, Inc., 2003.

[Schewe and Thalheim(2008)] Schewe, K.-D., Thalheim, B.: “Semantics in data and knowledge bases”; SDKB 2008; LNCS 4925; 1–25; Springer, Berlin, 2008.

[Simsion(2007)] Simsion, G.: Data modeling - Theory and practice; Technics Publications, LLC, New Jersey, 2007.

[Stachowiak(1992)] Stachowiak, H.: “Modell”; H. Seiffert, G. Radnitzky, eds., Handlexikon zur Wissenschaftstheorie; 219–222; Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.

[Thalheim(2000)] Thalheim, B.: Entity-relationship modeling – Foundations of database technology; Springer, Berlin, 2000.

[Thalheim(2009)] Thalheim, B.: “The conceptual framework to multi-layered database modelling”; Proc. EJC; 118–138; Maribor, Slovenia, 2009.

[Thalheim(2010)] Thalheim, B.: “The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling”; The Handbook of Conceptual Modeling: Its Usage and Its Challenges; chapter 17, 547–580; Springer, Berlin, 2010.

[Web(1991)] “Websters ninth new collegiate dictionary”; (1991).

[Whorf(1980)] Whorf, B.: Lost generation theories of mind, language, and religion; Popular Culture Association, University Microfilms International, Ann Arbor, Mich., 1980.

The Art of Conceptual Modelling

Bernhard THALHEIM [1]

Christian-Albrechts-University Kiel, Computer Science Institute, 24098 Kiel, Germany

Abstract. Conceptual modelling is one of the central activities in Computer Science. Conceptual models are mainly used as intermediate artifact for system construction. They are schematic descriptions of a system, a theory, or a phenomenon of an origin thus forming a model. A conceptual model is a model enhanced by concepts. The process of conceptual modelling is ruled by the purpose of modelling and the models. It is based on a number of modelling acts, on a number of correctness conditions, on modelling principles and postulates, and on paradigms of the background or substance theories. Purposes determine the (surplus) value of a model. Conceptual modelling is performed by a modeller that directs the process based on his/her experience, education, understanding, intention and attitude.

Conceptual models are products that are used by other stakeholders such as programmers, learners, business users, and evaluators. Conceptual models use a language as a carrier for the modelling artifact and are restricted by the expressiveness of this carrier. This language is often also used for the description of the concepts that are incorporated into a modelling result. Concepts can be explicitly defined or can be implicitly assumed based on some common sense within an application domain, a Computer Science sub-culture and within a community of practice.

A theory of conceptual models and a theory of modelling acts have been developed in [26,27]. This paper aims at a development of a general theory of *modelling as an art* in the sense of [9]. A general theory of modelling also considers modelling *as an apprenticeship* and *as a technology*. We distinguish between the art of modelling within a creation and production process, the art of modelling within an explanation and exploration process, the art of modelling within an optimisation and variation process, and the art of modelling within a verification process. This distinction allows to relate the specific purpose with macro-steps of modelling and with criteria for approval or refusal of modelling results.

Keywords. conceptual modelling, modelling workflow, foundations of modelling.

1. The Conceptions of a 'Model', of 'To Model' and of 'Modelling'

Conceptual modelling is a widely applied practice in Computer Science and has led to a large body of knowledge on constructs that might be used for modelling and on methods that might be useful for modelling. It is commonly accepted that database application development is based on conceptual modelling. It is however surprising that only very few publications have been published on a *theory of conceptual modelling*. We continue the approach [27,26] and aim in a theory of modelling within this paper. An approach to a theory of models has been developed in [27]. An approach to a theory of model activities is discussed in [26].

[1] thalheim@is.informatik.uni-kiel.de http://www.is.informatik.uni-kiel.de/∼ thalheim

1.1. Three Guiding Concerns during Conceptual Modelling

Conceptual modelling is often only discussed on the basis of modelling constructs and illustrated by some small examples. It has however three guiding concerns:

1. *Modelling language constructs* are applied during conceptual modelling. Their syntactics, semantics and pragmatics must be well understood.
2. *Application domain gathering* allows to understand the problems to be solved, the opportunities of solutions for a system, and the requirements and architecture that might be prescribed for the solution that has been chosen.
3. *Engineering* is oriented towards encapsulation of experiences with design problems pared down to a manageable scale.

The first concern is handled and well understood in the literature. Except few publications, e.g. [2], the second concern has not yet got a sophisticated and well understood support. The third concern has received much attention by data modelers [19] but did not get through to research literature. It must therefore be our goal to combine the three concerns into a holistic framework.

1.2. Implications for a Theory of Conceptual Modelling

The three concerns of conceptual modelling must be integrated into a framework that supports the relevant concern depending on the modelling work progress. The currently most difficult concern is the engineering concern. Engineering is inherently concerned with failures of construction, with incompleteness both in specification and in coverage of the application domain, with compromises for all quality dimensions, and with problems of technologies currently at hand.

At the same time, there is no universal approach and no universal language that cover all *aspects of an application*, that have a well-founded *semantics* for all constructions, that reflect any *relevant facet in applications*, and that *support engineering*. The choice of modelling languages is often a matter of preferences and case, empirical usage, evolution history, and supporting technology.

1.3. Differences between 'Model', 'To Model' and 'Modelling'

The conceptions of model, of the activity 'to model' and of modelling are often used as synonyms. We must however distinguish these conceptions for a theory of models, a theory of model activities and a theory of the modelling process.

Based on the notions in the Encyclopedia Britannica [17] we distinguish between the conception of a model, the conception of a model activity, and the conception of modelling processes.

The model as an artifact: The model is something set or held for guidance or imitation of an origin and is a product at the same time. Models are enduring, justified and adequate artifacts from one side. From the other side, models represent the state of comprehension or knowledge of a user.

To model as an activity: 'To model' is a scientific or engineering activity beside theoretical or experimental investigation. The activity is an additive process. Corrections are possible during this activity. Modelled work may be used for construction of systems, for exploration of a system, for definition and negotiation, for communication, for understanding and for problem solving.

Modelling as a systematically performed technological process: Modelling is a technique of systematically using knowledge from computer science and engineering to introduce technological innovations into the planning and development stages of a system. At each stage the modeller is likely to ask both why and how, rather than merely how. Modelling is thus based on paradigms and principles.

Additionally, the notion of model may be used in an adjective sense as serving as or capable of serving as a pattern or being a usually miniature representation of something. This notion is often used for sample representations such as a 'model chair'. Another notion of the model that is not of interest within this paper is the miniature representation of something.

1.4. The Simultaneity of Art, Technology and Techniques in Modelling

Modelling can be understood as a technique[2] or as a technology[3]. [17] distinguishes between science and technology: Technology is the systematic study of techniques for making and doing things; science is the systematic attempt to understand and interpret the world. While technology is concerned with the fabrication and use of artifacts, science is devoted to the more conceptual enterprise of understanding the environment, and it depends upon the comparatively sophisticated skills of literacy and numeracy.

At the same time, modelling is an art[4]. Modelling is a highly creative process. It requires skills in planning, making, or executing. It is often claimed that it is not to be formalisable. It requires deep insight into the background as well as skills, careful simplification, experience and ingenuity. Due to the variety of viewpoints, modelling is also based on judgement and clever selection with different alternatives.

1.5. Conceptual Modelling: Modelling Enhanced by Concepts

An information system model is typically a schematic description of a system, theory, or phenomenon of an origin that accounts for known or inferred properties of the origin and may be used for further study of characteristics of the origin. *Conceptual modelling* aims to create an abstract representation of the situation under investigation, or more precisely, the way users think about it. Conceptual models enhance models with concepts that are commonly shared within a community or at least between the stakeholders involved in the modelling process. A general definition of concepts is given in [8,27]. Concepts specify our knowledge what things are there and what properties things have. Concepts are used in everyday life as a communication vehicle and as a reasoning chunk. *Conceptualisation* aims at collection of objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them. It is thus an abstract, simplified view or description of the world that we wish to represent.

---

[2] I.e., the fashion, manner, mode, modus, system, way, wise in which a system etc. is mastered. Techniques consist of methods of accomplishing a desired aim.

[3] Technology is an element of engineering. It consists of the practical application of knowledge especially in a particular area. It provides a capability given by the practical application of knowledge. Therefore, it is a manner of accomplishing a task especially using technical processes, methods, or knowledge.

[4] Art requires capability, competence, handiness, and proficiency. Art is based on finesse, i.e. on refinement or delicacy of workmanship. Models and art share a Janus head evaluation: The judgement of beauty evaluates the model within a community of business users. The judgement of the sublime evaluates the model against its technical realisation. A model has thus both an extrinsic and intrinsic value. Art will be used in the sequel in this paper within the wide understanding of [9].

1.6. The Theory of Conceptual Models

The theory of conceptual models [4] extends the framework [22,23]. A model can be characterised by four main dimensions: (1) *purpose*, (2) *mapping of an origin*, (3) *use of languages as a carrier*, and (4) *providing a value*.

The purpose of a model covers a variety of different intentions and aims such as *perception support* for understanding the application domain, *explanation and demonstration* for understanding an origin, *preparation* to management and handling of the origin, *optimisation* of the origin, *hypothesis verification* through the model, *construction* of an artifact or of a program, *control* of parts of the application, *simulation* of behaviour in certain situations, and *substitution* for a part of the application. Depending of the purpose we shall use different models.

Models are author-driven and addressee-oriented. They depend therefore on the culture, attitude, perceptions, education, viewpoints etc. of the stakeholders involved into the modelling process. Models are purposeful/situated/easily-modifiable/sharable/reusable/multi-disciplinary/multi-media chunks of knowledge about the application domain. They are both bigger and smaller than theories, i.e., bigger since they integrate ideas from different theories, since they use different representations, and since they are directed by their purpose; smaller since they are created for their purpose in a specific situation and since they are developed to be sharable and reusable. One of the most important quality characteristics of a model is that it should be easy to modify and to adapt.

1.7. The Theory of Model Activities as a Process

Activities for 'to model' (model activities) are based on modelling acts. The process for 'to model' is a specific form of a process. We may thus develop workflows of such activities. These workflows are based on work steps [24] such as 'decompose' or 'extend', abstraction and refinement acts, validation and verification, equivalences of concepts, transformation techniques, pragmatistic solutions and last but not least the domain-specific solutions and languages given by the application and implementation domains.

The act of 'to model' is based on an *activity* that is characterised by the *work products*, the *aspects* under consideration (scope), the *resources* used in an activity, and the *partners* involved into the activity. Additionally we might extend this characterisation by activity goals and intentions (for what), time span (when), and restrictions (normal, exception and forbidden cases) or obligations for later activities. We may distinguish a number of activities and acts, e.g., *understand, conceptualise, abstract, define, construct, refine, document* and *evaluate*. A theory of model activities has been developed in [26]. Model activities should be governed by *good practices* which can be partially derived from modelling as an apprenticeship or technology.

1.8. Orientation of this Paper

This paper explores modelling as an art. We base the discussion on a theory of models and of model activities. We abstract therefore in this paper from micro-, meso-, macro-models used in many natural sciences or model suites [25], e.g., model ensembles used in UML or OWL. We do not yet consider modelling competency or MDA/D.

Based on this understanding we may conclude that modelling requires apprenticeship and technology. The orientation towards an expert mode can be reached if modelling

is based on systematic development and if modelling is considered to be a craft of model activities. This approach shows that modelling incorporates design science in a wider sense as it has been considered in the literature.

We base our ideas on our observations on model developments for very large database schemata and very large database systems.[5]. Such systems require a well organised modelling process. They must be evolution-prone and revision-prone. Therefore, the model is typically on much higher quality than those models in textbooks. The paper concentrates thus one of the main workflows for information system development: description of application worlds followed by prescription for system worlds.

2. The Model World and its Dimensions

2.1. Main Dimensions of Models

Models are *artifacts*. They can thus be characterised by *main (primary) dimensions*:

purpose *("wherefore")* of models and modelling with the intentions, goals, aims, and tasks that are going to be solved by the model,

result of mapping *("whereof")* with a description of the solution provided by the model, the characterisation of the problem, phenomena, construction or application domain through the model,

language *("wherewith")* based on a careful selection of the carrier (or cargo[11] or language) with all its restrictions that allows to express the solution, the specification of the world or the construction, and

value *("worthiness")* of a model by explicit statement of the internal and external qualities, and the quality of use, e.g. either by explicit statement of invariance properties relating the model to its associated worlds or by preservation properties that are satisfied by the model in dependence on the associated worlds.

The mapping associates the origin and the artifact or the artifact and its realisation. The mapping dimension is discussed in [26]. As far as we are interested in modelling of information systems, we may use a (semi-)formal language for language dimension of the artifact. The main dimensions of models and modelling govern the model and the modelling acts. The value dimension can be described based on [7].

The main dimensions are extended by different *context dimensions* that are used to shape and to adapt the model, e.g., (a) the *application domain dimension* describes the scope and the neglection of the model, (b) the *systems dimension* reflects the realisation

[5] Due to our involvement into the development and the service for the CASE workbenchs $(DB)^2$ and ID^2 we have collected a large number of real life applications. Some of them have been really large or very large, i.e., consisting of more than 1.000 attribute, entity and relationship types. The largest schema in our database schema library contains of more than 19.000 entity and relationship types and more than 60.000 attribute types that need to be considered as different. Another large database schema is the SAP R/3 schema. It has been analyzed in 1999 by a SAP group headed by the author during his sabbatical at SAP. At that time, the R/3 database used more than 16.500 relation types, more than 35.000 views and more than 150.000 functions. The number of attributes has been estimated by 40.000. Meanwhile, more than 21.000 relation types are used. The schema has a large number of redundant types which redundancy is only partially maintained. The SAP R/3 is a very typical example of a poorly documented system. Most of the design decisions are now forgotten. The high type redundancy is mainly caused by the incomplete knowledge on the schema that has been developed in different departments of SAP.

opportunities, weaknesses, strengths and threats, and (c) the *user* or stakeholder *dimension* describes the viewpoint, orientation and background of users involved.

2.2. The Model as a Physical or Virtual Artifact

The main product of modelling and model activities is the model, i.e. an artifact that is considered to be worth for its purpose by the author. The model can, for instance, be used for the description of the world of origins or for the prescription of constructions. There are a number of explicit choices an author makes and that rule applications of models. Modelling of information systems

depends on the *abstraction layer*, e.g. requirements, specification, realisation or implementation layer,
depends on chosen *granularity and precision* of the work product itself,
depends on *resources* used for development of a model such as the language,
depends on *level of separation of concern* such as static/dynamic properties, local/global scope, facets,
depends on *quality properties of the input* , e.g. requirements, completeness, conciseness, coherence, understandability,
depends on decomposition of the work products in ensembles of sub-products, and
satisfies quality characteristics such as quality in use, internal quality, and external quality.

The task of model development is never completed (ta panta rhei ($\tau\alpha\ \pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$), 'the rivers flow'; narrative: everything flows). Models are changing artifacts due to changes imposed by

scope insight for conscious handling of restriction, capabilities, opportunities,
guiding rules for convenience, for completion, refinement, and extension,
development plans for partial delivery of models, partial usage and deployment,
theories supporting development of models,
quality characteristics for model completion, model evolution, model engineering, and
mapping styles for mapping models among abstraction layers.

2.3. The Purpose Dimension

The purpose dimension is *ruling* and *governing* the model, the development process and the application process because of the main reason for using a model is to provide a solution to a problem. Therefore the purpose is characterised by the solution to the problem provided by the model. We may distinguish a number of concerns such as

the impact of the model *("whereto")* for a solution to a problem,
the insight into the origin's properties *("how")* by giving details how the world is structured or should be structured and how the functionality can be described,
restrictions on applicability and validity *("when")* of a model for some specific solutions, for the validity interval, and the lifespan of a model,
providing reasons for model value *("why")* such as correctness, generality, usefulness, comprehensibility, and novelty, and
the description of functioning of a model *("for which reason")* based on the model capacity.

The purpose dimension governs the workflows applied in conceptual modelling. It also governs the kind of model application. We may distinguish a number of workflows in conceptual modelling such as the following ones:

Description-prescription workflows result in creation of models that are used for production of systems.

Explanation workflows result in new insight into the world of the origins.

Optimisation-variation workflows result in an improvement and adaptation of the origins.

Verification-validation-testing workflows result in an improvement of the one of the subjects considered, in most cases in an improvement of models.

Reflection-optimisation workflows are typical for mathematical modelling of the world of origins.

Explorative workflows are using models for learning about origins.

Hypothetical workflows are typical for discovery science, e.g., science used for climate research.

Documentation-visualisation workflows target on better understanding and comprehension of models.

These workflows can be intertwined or shuffled with each other. They may be performed one after another. In this paper we concentrate on the description-prescription (or *creation-production*) workflow which seems to be central for information systems.

2.4. The Language Dimension

Models are represented by *artifacts* that satisfy the pragmatic purposes of users. We restrict our discussion within this subsection to formal languages that are typically used for conceptual models. In this case, artifacts are linguistic expressions that describe the model. Linguistic expressions are built within a language with some understanding. Therefore, artifacts use syntax, semantics and pragmatics built within the chosen language.

Figure 1 displays the relationship between an artifact and its objectives and properties. A model should support its objectives. Optimally, these objectives $\Psi(G)$ can be expressed in the same language $\mathcal{L}_{\tilde{G}}$ that is also used for the model G. A model has a number of properties. Some of them are of interest and used for characterisation of the model, e.g., $\Phi(G)$. This characterisation depends on the model and its purpose.

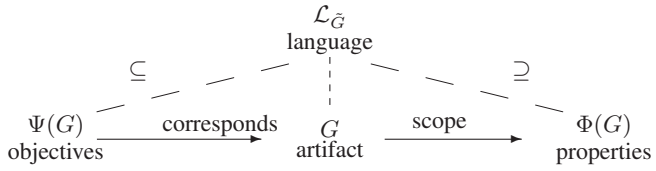


Figure 1. Artifacts with a language, their properties and objectives within a given language for the artifact

Constructive languages are a special case and support

- the prescription of the objectives or postulates that restrict the judgement that an artifact can be accepted as a model,

- the scope of our attention to those artifacts that can be considered for a model or for parts of a model, and
- the orientation of the user on certain properties that are of interest for the purpose of modelling.

2.5. The Context Dimensions

2.5.1. The User Dimension

A number of users are involved into the development of models. The user dimension thus reflects intentions, the understanding, the comprehension and other characteristics of users in a variety of roles, e.g.,

the role of an *author* *("by whom")* that results in reflections of the educational level, application of templates, pattern or reference models,

the role of an *addressee* *("to whom")* that restricts the utilisation of the model or that supports the extended application beyond the purpose originally intended, and

the role of *broad public* *("whichever")* that develops a common understanding of the model depending on the group or the culture of the public.

2.5.2. The Application Domain Dimension and the World of Origins

The application domain consists of people, organisational systems, and technical systems that interact to work towards a goal. This dimension clarifies

the *domain depending on models purpose* *("for what")* such as an application domain, properties reflected or neglected,

the *scope to specific elements* *("what")* that are considered to be typical and whose properties should be reflected,

the *attention within the domain depending on models purpose* *("where")* that limits the model to the 'normal' aspects,

the *orientation of the domain* *("wherefrom")* that restricts the attention and the issues for the current activities supported by the model,

the sources for origins or the infrastructure *("whence")* considered for the model, and

the *restrictions of the world* *("wherein")* associated with the model.

3. The Modelling Process

3.1. Conceptual Modelling Activities Governed by its Purpose

Models are developed with different goals, different scope, within different context, with different appeal to the receiver of the model, with different granularity, with different background, and with different realisation forms. Therefore we have to explicitly handle modelling purpose properties.

The *mission* of modelling is described by scope of the model, the users community, the tasks the model might support, the major and minor purposes satisfied by the model and the benefits obtained from the model for the given user community. The *goals* of a model are based on the impact of the model, restricted by the relationships among users and their roles they are playing. The *brand* of the model is given by the who-

what-whom-action pattern. The *meta-model* can be used to provide information about the model such as the context of the model, the context in which the model might be useful to the auditory, the usage of the model, and the restrictions of the model.

It surprises that these model properties are not explicitly handled in most modelling approaches. The same surprise can be observed for a declaration of the main goals of the modelling act such as

construct a model, a part of the model, a concept or a judgement, etc. (describe, delineate, fabricate, master),
communicate the judgements, the observations, the concepts, etc. (explain, express, verbalise or display),
understand the application domain, the system opportunities, etc. (cognise, identify, recognise, percept),
discover the problems, the potential, the solutions, etc. (interact, identify),
indicate properties of importance, relevance, significance, etc. (visualise, measure, suggest, inform),
variate and *optimise* a solution, a judgement, a concept, a representation depending on some criteria,
verify or *validate* or *test* a model, a solution, a judgement, a representation or parts of those,
control the scope of modelling, the styles or pattern, parts of a model, judgements, etc. (rule, govern, proofread, confirm, restrain, administer, arrange, stratify, standardise),
alternate or *compensate* or *replace* or *substitute* or *surrogate* models or parts of them, judgements, concepts, etc. (transfer, reassign, evolve, migrate, balance, correct, novate, truncate, ersatz).

The first and last four goals lead to a *datalogical* model that is structured according to technology. The other goals result in an *infological* model that is delivered to the needs of the user. We thus use a different frame of reference. The application of the results may thus be descriptive or prescriptive, constitutive or prognosticating, categorical or exegetic or contemplative or formulaic.

3.2. Properties of Activities To Model

Activities to model form a process and can be characterised by a number of (ideal) properties:

Monotonicity: Activities are monotone if any change to be applied to one specification leads to a refinement. It thus reflects requirements in a better form.

Incrementality: Activities are iterative or incremental if any step applied to a specification is only based on new requirements or obligations and on the current specification.

Finiteness: Activities are finite if any quality criteria can be checked in finite time applying a finite number of checks.

Application domain consistency: Any specification developed corresponds to the requirements and the obligations of the application domain. The appropriateness can be validated in the application domain.

Conservativeness: Activities are conservative if any model revision that cannot be reflected already in the current specification is entirely based on changes in the requirements.

Typical matured activities to model are at least conservative and application domain consistent. Any finite sequence of activities can be transformed into a process that is application domain consistent. The inversion is not valid but depends on quality criteria we apply additionally. If the modelling process is application domain consistent then it can be transformed in an incremental one if we can extract such area of change in which consistency must be enforced.

3.3. Towards Modelling Principles

A theory of conceptual modelling can be based on a system of guiding principles. We conclude that at least three guiding principles must be explored in detail:

Internal principles are based on a set of ground entities and ground processes.

Bridge principles explain the results of conceptual modelling in the context of their usage, for instance for explanation, verification/validation, and prognosis.

Engineering principles provide a framework for mastering the modelling process, for reasoning on the quality of a model, and for termination of a modelling process within a certain level of disturbance tolerance (error, incompleteness, open issues to be settled later, evolution).

Information systems modelling principles are specialisations of design principles [3]. They are conventions for planning and building correct, fast or high performance, fault tolerant, and fit information systems. Conceptual modelling is based on architecture of a system into components, uses their interactions, and pictures their layout. Modelling is the process of producing models. It is thus adapted from engineering and may thus use the separation of activities into requirements, specification, development, and testing.

Depending on the purpose of the model several quality criteria may be preferred. For instance, construction models should fulfill criteria for good models such as correctness of models, refinement to highly effective systems, fault tolerance of systems, ubiquity of systems, and fitness of systems.

Modelling principles are not laws of nature, but rather conventions that have been developed by modellers to the most success when pursuing quality properties. Therefore, various sets of principles might be developed depending on the community. For instance, modelling based on extended ER models is based on compositionality, incrementality, structure-orientation, and conservativeness. Modelling principles for sets of models such as UML are far more difficult to develop and to maintain.

4. Modelling of Information Systems as an Engineering Science

In the sequel we concentrate on one of the workflows: the prescription of systems imposed by the description of an application domain and of the problems under solution. This workflow is often considered to be one of the main workflows. We may also use other workflows. The description-prescription workflow is however a typical example of an engineering workflow[6]. Engineering is nowadays performed in a systematic and

[6]The difference between scientific exploration and engineering is characterised by [18] as follows: "Scientists look at things that are and ask 'why'; engineers dream of things that never were and ask 'why not'. Engineers use materials, whose properties they do not properly understand, to form them into shapes, whose

well-understood form. We can thus include engineering approaches to modelling. We first review contributions made by design science, e.g., [5,6,12,28] and then develop our approach.

4.1. The Design Science Approach

MIS design science aims at the development of a general theory for models, model activities and modelling. We shall use the approach for a deeper insight into modelling. Models are called 'design' in [6].

The management information system community characterises the modelling process by seven guidelines [6]:

(1) model are purposeful IT artifacts created to address a problem;
(2) models are solutions to relevant and important problems;
(3) the utility, quality, and efficacy of models must be evaluated by quality assessment;
(4) modelling research must contribute to the state of the art;
(5) modelling research relies upon the application of rigorous methods;
(6) modelling is a search process and use termination conditions;
(7) models must be communicated both to technology-oriented as well as to management audiences.

We observe that guidelines (1), (2), and (7) are characterising the model. Guidelines (3), (6) characterise model activities. Guideline (3), (5) is related to modelling as a technology. Guideline (4) is a general statement that relates modelling to a science.

Main ingredients of modelling can be derived from these guidelines [1,5,12,14,20, 29]. Core components are purpose and scope (causa finalis), artifacts (causa materialis), the oneness of form and function (causa formalis), artifacts mutability, testable propositions about the model, and theoretical underpinning. Additional requests are the potential implementation (causa efficiens) and utility for exposition and testing [5].

Design science separates three cycles [28]: the relevance (or description) cycle, the design (or modelling) cycle, and the rigor (or conceptualisation) cycle.

4.2. Reasoning Support for Modelling

Design science [6] has been targeting on an explicit support for the modelling process. This support includes an explicit consideration of the quality of the model, of the quality of the modelling process, and of the quality of supporting theories. We may combine the informal discussions with our approach and separate the modelling acts by the things that are under consideration. Figure 2 displays the different ways of working during a database systems development. We use here the two-phase model: Description followed by prescription.

These different "ways of working" characterise

- the *modelling acts* with its specifics; [26]
- the *foundation for the modelling acts* with the theory that is going to support this act, the technics that can be used for the start, completion and for the support of the modelling act, and the reasoning techniques that can be applied for each step;

geometries they cannot properly analyse, to resist forces they cannot properly assess, in such a way that the public at large has no reason to suspect the extent of their ignorance." Modelling incorporates both engineering and science. It is thus considered to be an engineering science.

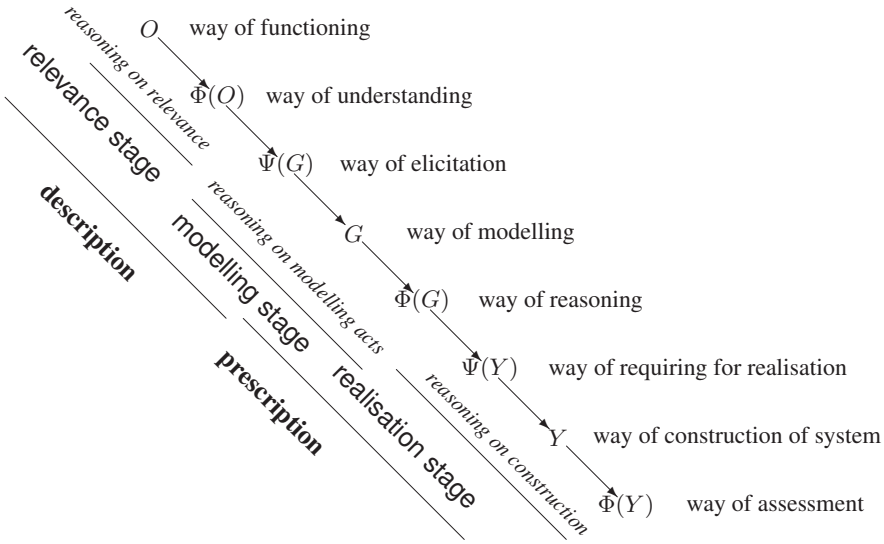


Figure 2. Reasoning processes and reasoning support for description followed by prescription

- the *partner* involved with their obligations, permissions, and restrictions, with their roles and rights, and with their play;
- the *aspects* that are under consideration for the current modelling acts;
- the *consumed and produced elements of the artifact* that are under consideration during work;
- the *resources* that must be obtained, that can be used or that are going to be modified during a modelling act.

Consider, for instance, the way or requiring. It includes specific facets such as

- to command, to require, to compel, and to make someone do something with supporting acts such as requesting, ordering, forbidding, proscribing;
- to consider obligatory, to request and expect with specific supporting acts such as transmitting, communicating, calling for, demanding;
- to need with supporting acts of wanting, requiring;
- to necessitate, to postulate, to take, to involve, and to require as useful, to just, or to proper.

The ways of functioning, understanding, elicitation, modelling, reasoning, assessment, and construction can be characterised in a similar form.

The realisation stage may be replaced by other stage that support different purposes. We concentrated on prescription and construction of new systems. Another application is *model refinement*.

Design science aims at another kind of model refinement by adding more rigor after evaluation of a model. This refinement is essentially *model evolution* and *model evaluation*. Another refinement is the enhancement of models by concepts. This refinement is essentially a 'semantification' or *model conceptualisation* of the model.

Experimentation and justification of models is a third kind of adding rigor to (conceptual) models.

4.3. Observations for Information Systems Model Engineering

Engineering of conceptual models inherits both facets of didactically ruled learning [21] and of engineering [18]. The following characteristics of engineering sciences are observed also for conceptual modelling:

(α) The origin of a model is partly a product of creativity.

Systems developed in our field are a product of developers and thus dependent on these stakeholders. They must be understood, well-explained and used with a purpose.

(β) The origin of a model is a complex systems.

The attention focuses both on the creation of complex artifacts and on conceptualisiations of the application world. They are typically modularly constructed. Modularisation is only one of the underlying design principles. Conceptual modelling targets at useful concepts. It goes through a series of iterative design cycles in dependence on its purpose.

(γ) Models satisfy the purpose, are sharable, useful and reusable.

Models are not developed just as an intermediate result of the implementation process and for satisfaction of purpose. They are shared within a community and are reusable in other situations. Moreover, models support a better understanding of the origins.

(δ) The origins are continuously changing and thus the models too.

The application domain is continuously evolving. Models must correspondingly evolve too. Significant changes tend to be applied to the starting model so that the original concepts become unrecognisable after model evolution. Models are also used for changing the application world. This change must again be reflected within the model.

(ϵ) Origins being modelled are influenced by social constraints and affordances.

Models are influenced as much by purposes as by physical and economical aspects of the contexts in which they are used. These influences are changing and evolving as well. Therefore, models are going to be used in ways their stakeholders did not imagine. Models are influenced as much by socially generated capital, constraints, and affordances as by the capabilities of stakeholders who created them.

(ζ) No single "grand theory" is likely to provide realistic solutions to realistic complex application problems.

In realistic modelling situations that involve information systems, there almost never exist unlimited resources. Relevant stakeholders have typically conflicting goals. Therefore, ways of working displayed in Figure 2 usually need to integrate approaches drawn from different disciplines.

(η) Development of a model usually involves a series of iterative modelling cycles.

Artifacts that serve as models are developed through a series of model activities and are iteratively tested and revised in dependence on the purpose.

Consequences for model engineering: The modelling process itself also changes the application domain and the understanding of the origin. Therefore, modelling is not reducible to condition-action rules. Modelling is a matter of engineering. Experienced modellers not only right develop a model but they also develop the right model - by developing models at the right time, with the right background and context, and for the right purpose. Model engineering is therefore based on advanced skills of handcrafting, i.e., making substitutions and adaptations depending on purpose and application situation, understanding which compositions perform best, continuously adapt the result of the process, and understand difficult-to-control things in their handcraft environment. The same situation is valid for information systems development if performance of the system becomes crucial and heavily depends on the DBMS.

4.4. Modelling Generalising Engineering Approaches

The development of models offer many challenges. Modelling is essentially synthetic rather than analytic in substance. Identifying the real task of the modelling problem is probably the greatest challenge. Models can be based on building blocks. Another challenge is to find the right modelling method. Engineering targets at capacity of products to withstand service load both effectively and efficiently during their service life [18]. Efficiency also considers performance of the system. Engineering is also concerned with avoidance of technical, operational or unpredictable failures, i.e. to develop a system that deflect all service loads.

Engineering science for modelling is based on many different supporting sciences and technologies: industrial design, ergonomics, aesthetics, environments, life sciences, economics, mathematics, marketing, and manufactoring and forming processes. Engineering design includes five facets: design for effective function, design for manufacture, design for human users, socially responsible design, and economically responsible design.

Engineering distinguishes three dimensions: the stakeholder dimension, the procedure or process dimension, and the product dimension. It uses many techniques such as enformulation for structuring the purposes and objectives, problem decomposition together with component engineering, problem evolution, organising the engineering process, result evaluation, and result management. It also considers economic, social and environmental issues.

Therefore, it seems to be natural to use achievements of engineering for understanding modelling. This similarity is not only applicable to the description-prescription workflow but also for all the other workflows.

4.5. Maieutics for Mastering Iterations

Modelling of information systems is not only aiming at achieving a nominal system but aims too at satisfaction of real interests of all stakeholders involved into modelling. It must consider all relevant aspects of an application and thus results in co-design of structuring, functionality, and supporting systems such as view and interaction support [24]. Stakeholders (or users) iteratively obtain a deeper insight and understanding about the necessities and conditions of the problem and the strengths, weaknesses, opportunities and threats of the solution depending of the purpose of the modelling within a modelling process. Therefore, modelling integrates ideas developed for maieutics [10,13].

The maieutics frame [15] is essentially a specific form of a dialogue. In conceptual modelling, it consists (1) of an open-ended process, (2) of the elaboration of ideas that are grounded in references to the application domain, to the users, prior knowledge and experience, and to the languages as carriers, and (3) of the discussion (in form of conceptualisation, interpretation, explanation, diverging ideas, and new understandings) that is inductive and exploratory rather than deductive and conclusive.

Modelling requires to utilise the knowledge in dependence on the purpose of the model. Answers found during modelling may not be evident in the material on hand; modellers may have to delve into subtleties or ambiguities they had not thought of. Information systems modelling is based on elaboration and conceptualisation of model elements. The inductive and exploratory discussion facilitates the development of argumentation by fostering the (re)consideration of alternatives and versions.

Conceptual modelling is based on references to the application domain, connections across the model, elaboration based on prior knowledge and/or experience, interpretations, explanations and conceptualisations, diverging ideas, and new understandings. Therefore the modelling process is highly iterative and revising/remastering decisions that have already been made.

5. The Description-Prescription Information Systems Modelling Workflow

Modelling is based on an evolutionary process and thus consists of at least three subprocesses:

- *selection* including rigorous testing against the origin,
- *communication* for generation of a common understanding and a productive way of thinking within a community, and
- *accumulation* of results and integration of these results into future developments.

5.1. Examples of Workflows

The description-prescription workflow is one of the most prominent workflows in information system modelling. Methodologies developed for software engineering can be directly applied to this workflow. They are however mainly oriented towards system construction. The systems dimension is not as well explored. The combination of these two sub-workflows is shown in Figure 3. We need to include into this combination also the quality dimension. The body of knowledge of software engineering includes also a large set of quality characteristics. [7] develops an approach to systematic quality development. We integrate this systematic quality management.

We can also develop other workflows such as agile modelling, spiral modelling and incremental modelling workflows. We restrict our attention in the sequel to the workflow in Figure 3 due to the length of this paper. This workflow separates three different worlds: the world of applications, e.g., the application domain in dependence on the purpose; the world of models, e.g. conceptual models used for information system development; the world of systems, e.g., information systems combined with presentation systems. Based on this separation we can distinguish three stages: the relevance stage, the modelling stage, and the realisation stage. This workflow reflects the separation into objectives, the artifact and the properties within the language dimension.

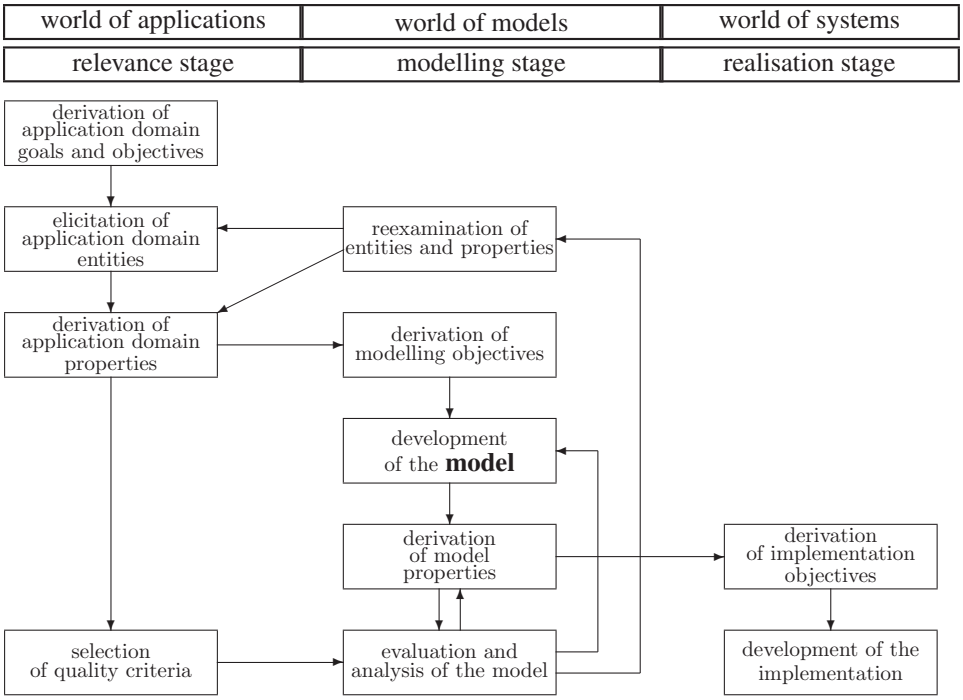


Figure 3. The description-prescription workflow that includes quality assurance

5.2. Sub-Workflows

Modelling integrates the classical problem solving four-phase cycle [16] in each of the sub-workflows as well:

1. Developing an understanding of the task: The task is analysed within its context and is compared with the goal for completion of the task. The initial situation is characterised. in the case of modelling processes the understanding is based on objectives derived from the purpose or from previous steps.
2. Development of a plan for the solution of a task: The instruments and tools for the solution of the task are reconsidered. The plan consists of heuristic forward and backtracking steps, steps for problem restructuring and for quality control.
3. Application of the plan for the development of the solution: The plan is consecutively applied for the generation of the solution. If certain steps are considered to be inappropriate then the plan is revised as well.
4. Development of an understanding of the solution: The result is evaluated based on criteria that either follow from the purpose or from the problem. Properties of the solution are derived.

This approach uses four state spaces:

The ***state space*** consists of the collection of all those states that are reachable from the *initial state*. Some of the states are considered to be desirable, i.e. are *goal states*. States can be modelled through languages such as ER. States may have properties such as suitability for certain purposes.

The ***actions*** allow to move from one state to another state under certain conditions. We may assume that the effect of the actions is observable to a certain extent by the user. User may use several actions in parallel. Actions may be blocked or enabled depending on conditions. Actions may be used at some cost.

The ***goal test*** determines whether a given state or state set satisfies the goals. The goal test may be defined through a set of states or through properties. The goal test may also allow to state which quality has the state set for the problem solution.

The ***controller*** evaluates the actions undertaken by the stakeholder. Some actions may be preferred over other, e.g. have less costs, or are optimal according to some optimality criterion. Controllers can be based on evaluators of the paths from the initial state to the current state.

Creation steps are the most complex steps in modelling. They typically consist of an orientation or review substep, of a development step performed in teams, and of a finalisation substep . Creation steps are composed of a number of substeps that can be classified into:

- Review of the state-of-affairs: The state of the development is reviewed, evaluated, and analysed. Obligations are derived. Open development tasks can be closed, rephrased or prioritised.
- Study of documents and resources: Available documents and resources are checked whether they are available, adequate and relevant for the current step, and form a basis for the successful completion of the step.
- Discussions and elicitation with other partners: Discussions may be informal, interview-based, or systematic. The result of such discussions is measured by some quality criteria such as trust or confidence. They are spread among the partners with some intention such as asking for revision, confirmation, or extension of the discussion.
- Recording and documentation of concepts: The result of the step is usually recorded in one work product or consistently recorded in number of work products.
- Classification of concepts, requirements, results: Each result developed is briefly examined individually and in dependence of other results from which it depends and to which it has an impact.
- Review of the development process: Once the result to be achieved is going to be recorded the work product is examined whether is has the necessary and sufficient quality, whether is must be revised, updated or rejected, whether there are conflicts, inconsistencies or incompleteness or whether more may be needed. If the evaluation results in requiring additional steps or substeps then the step or the substep is going to be extended by them.

This cycle follows the ‘maieutics’ cycle proposed by Sokrates [10,13]. Maieutics uses the methods of irony (or background elication), of induction towards generalisation and of definition (or concept development). For the modelling stage we derive the fol-

lowing correspondence between the maieutics cycle and the modelling stage:

problem initiation	problem differentiation and understanding	problem evaluation and selection by relevancy	model	justification and consensus	experimentation and field exploration	application
O origin	$\Phi(O)$ origin properties	$\Psi(G)$ modelling objectives	G artifact as model	$\Phi(G)$ model properties	$\Psi(Y)$ implementation objectives	Y implementation

We therefore arrive at a modelling process in Figure 4 that refines the general workflow in Figure 3.

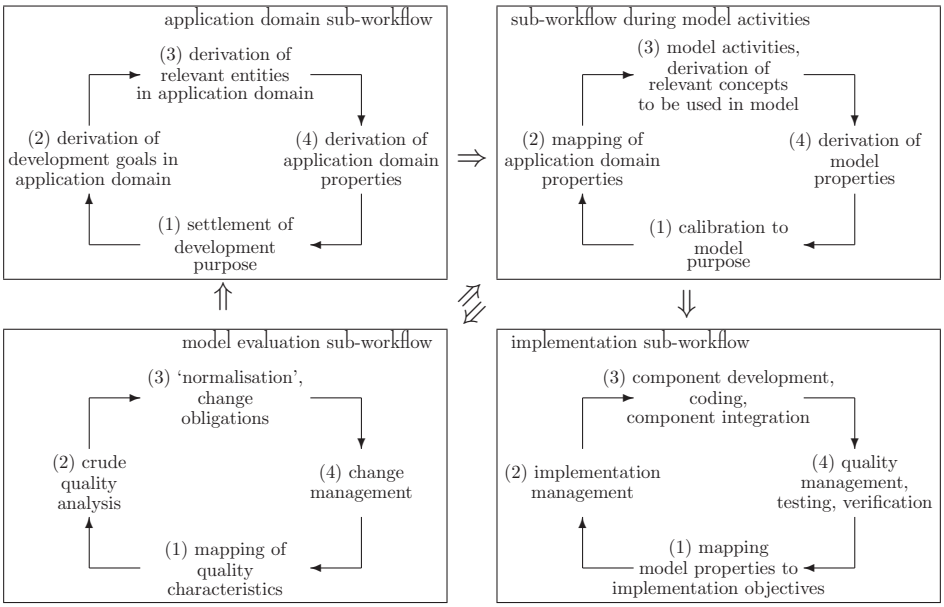


Figure 4. The sub-workflows for description-prescription modelling processes

We may zoom-in into these sub-workflows. For instance, one of the most interesting steps is step (3) in the model activities. This step consists of a number of substeps. Since the rigor or conceptualisation stage is orthogonal, the framework to database design in [24] is extended by conceptualisation in Figure 5.

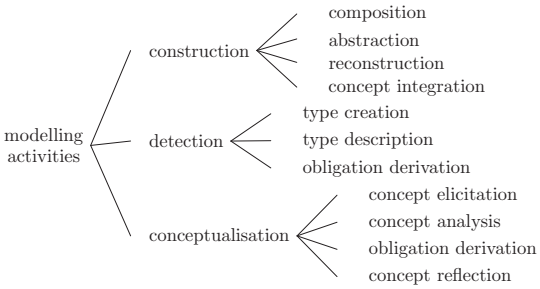


Figure 5. Sub-steps of the modelling and concept derivation step in the modelling sub-workflow

Conceptualisation is based on the notion of concepts introduced in [8,27]. Design science [5,6,12,28] uses the rigor cycle as one of its three cycles aiming at model development. The rigor cycle has not yet been defined. We can use the maieutics approach for the development of a conceptualisation cycle and arrive at a conceptualisation cycle displayed in Figure 6.

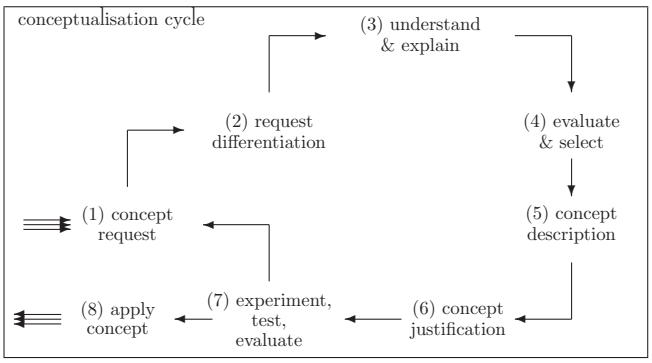


Figure 6. The sub-workflow for conceptualisation steps within the modelling step

6. Conclusion

Models are artifacts that can be specified within a (W^4+$W^{17}H$)-frame based on the classical rhetorical frame introduced by Hermagoras of Temnos[7]. They are primarily characterised by W^4: wherefore (purpose), whereof (origin), wherewith (carrier, e.g., language), and worthiness ((surplus) value). Secondary characterisation $W^{17}H$ is given by:

- user or stakeholder characteristics: by whom, to whom, whichever;
- characteristics imposed by the application domain: wherein, where, for what, wherefrom, whence, what;
- purpose characteristics characterising the solution: how, why, whereto, when, for which reason; and
- additional context characteristics: whereat, whereabout, whither, when.

Modelling is the art, the systematics and the technology of model (re)development and model application. It uses model activities and techniques. This paper is going to be extended by more specific aspects of the modelling art. One of them is modelling competency. Information systems modelling *competency* means being able to autonomously and insightful carry out all aspects of the modelling process in a certain context. It thus means the ability to identify relevant questions, variables, relations, postulates and assumptions in a given application domain, to translate these into computer engineering notions and to interpret and validate the model in relation to the purpose of modelling, as well as to analyse or compare given models by investigating the postulates and assumptions being made, checking properties and scope of given models, etc. This competency can be instantiated to the different ways of working in Figure 2.

[7] Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (W^7: Who, what, when, where, why, in what way, by what means). The Zachman frame uses a simplification of this frame.

References

[1] L. Apostel. Towards the formal study of models in the non-formal sciences. In H. Freudenthal, editor, *The concept and the role of the model in mathematics and natural and social sciences*, pages 1–37. Reidel, Dordrecht, 1961.

[2] D. Bjørner. *Domain engineering*, volume 4 of *COE Research Monographs*. Japan Advanced Institute of Science and Technolgy Press, Ishikawa, 2009.

[3] P.J. Denning. Great principles of computing. http://cs.gmu.edu/pjd/GP/, 2007.

[4] D. Embley and B. Thalheim, editors. *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*. Springer, 2011.

[5] S. Gregor and D. Jones. The anatomy of a design theory. *Journal of Association for Information Systems*, 8(5):312–335, 2007.

[6] A. Hevner, S. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quaterly*, 28(1):75–105, 2004.

[7] H. Jaakkola and B. Thalheim. Framework for high-quality software design and development: a systematic approach. *IET Software*, 4(2):105–118, 2010.

[8] Y. Kidawara, K. Zettsu, Y. Kiyoki, K. Jannaschk, B. Thalheim, P. Linna, H. Jaakkola, and M. Duzí. Knowledge modeling, management and utilization towards next generation web. In *Information Modelling and Knowledge Bases XXI*, volume 206, pages 387–402. IOS Press, 2010.

[9] D.E. Knuth. *The art of programming I-III*. Addison-Wesley, Reading, 1968-1973.

[10] H. Krauch. System analysis. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon zur Wissenschaftstheorie*, pages 338–344. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.

[11] B. Mahr. Information science and the logic of models. *Softw. Syst. Model.*, 8:365–383, 2009.

[12] S.T. March and V.C. Storey. Design science in the information systems discipline: An introduction to the special issue on design science research. *MIS Quarterly*, 4:725–730, 2008.

[13] M.D. Mesarovic and Y. Takahara. *General systems theory: Mathematical foundations*. Academic Press, New York, 1975.

[14] P.-A. Muller, F. Fondement, and B. Baudry. Modeling modeling. In *MoDELS*, volume 5795 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2009.

[15] P. Orellana. *Maieutic frame presense and quantity and quality of argumentation in a Paideia seminar*. Doctor of philosophy, University of North Carolina at Chapel Hill, 2008.

[16] G. Polya. *How to solve it: A new aspect of mathematical method*. Princeton University Press, Princeton, 1945.

[17] J.E. Safra, I. Yeshua, and et. al. *Encyclopædia Britannica*. Merriam-Webster, 2003.

[18] A. Samuel and J. Weir. *Introduction to Engineering: Modelling, Synthesis and Problem Solving Strategies*. Elsevier, Amsterdam, 2000.

[19] G. Simsion. *Data modeling - Theory and practice*. Technics Publications, LLC, New Jersey, 2007.

[20] B. Ja. Sovetov and S.A. Jakovlev. *Modelling of Systems*. Nauka, Moscov, 2005.

[21] B. Sriraman and L. English. *Theories about mathematics education*. Springer, Berlin, 2010.

[22] H. Stachowiak. Modell. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon zur Wissenschaftstheorie*, pages 219–222. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.

[23] W. Steinmüller. *Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993.

[24] B. Thalheim. *Entity-relationship modeling – Foundations of database technology*. Springer, Berlin, 2000.

[25] B. Thalheim. Model suites for multi-layered database modelling. In *Information Modelling and Knowledge Bases XXI*, volume 206 of *Frontiers in Artificial Intelligence and Applications*, pages 116–134. IOS Press, 2010.

[26] B. Thalheim. Towards a theory of conceptual modelling. *Journal of Universal Computer Science*, 16(20):3102–3137, 2010. `http://www.jucs.org/jucs_16_20/towards_a_theory_of`.

[27] B. Thalheim. The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In Embley and Thalheim [4], chapter 17, pages 547–580.

[28] J. R. Venable. Design science research post Hevner et al.: Criteria, standards, guidelines, and expectations. In *DESRIST*, volume 6105 of *Lecture Notes in Computer Science*, pages 109–123. Springer, 2010.

[29] C. von Dresky, I. Gasser, C. P. Ortlieb, and S Günzel. *Mathematische Modellierung: Eine Einführung in zwölf Fallstudien*. Vieweg, 2009.

Contents

Chapter 17
The Theory of Conceptual Models, the Theory of Conceptual Modelling and Foundations of Conceptual Modelling

Bernhard Thalheim

Abstract Conceptual modelling is a widely applied practice and has led to a large body of knowledge on constructs that might be used for modelling and on methods that might be useful for modelling. It is commonly accepted that database application development is based on conceptual modelling. It is however surprising that only very few publications have been published on a *theory of conceptual modelling*. *Modelling* is typically supported by languages that are well-founded and easy to apply for the description of the application domain, the requirements and the system solution. It is thus based on a *theory of modelling constructs*. Modelling is ruled by its purpose, e.g., construction of a system, simulation of situations in real world, theory construction, explanation of phenomena, documentation of an existing system, etc. Modelling is also an engineering activity with engineering steps and engineering results. It is thus *engineering*.

17.1 Towards a Theory of Conceptual Models and Conceptual Modelling

Models are different for different purposes. We may develop a model for analysis of an application domain, for construction of a system, for communicating about an application, for assessment, and for governance. These different purposes result in different goals and task portfolios. Models are an essential part in computer science. While preparing a survey on models we realised that computer science uses more than 50 different models. Analysing these different models we discover however four commonalities of these models:

Bernhard Thalheim
Department of Computer Science, Christian-Albrechts University Kiel, 24098 Kiel, Germany, e-mail: thalheim@is.informatik.uni-kiel.de

Purpose: Models and conceptual models are governed by the purpose. The model preserves the purpose. Therefore the purpose is an invariant for the modelling process.

Mapping: The model is a mapping of an origin. It reflects some of the properties observed or envisioned for the origin.

Language as a carrier: Models are using languages and are thus restricted by the expressive power of these languages. Candidates for languages are formal or graphical languages, media languages, illustration languages, or computer science constructions.

Value: Models provide a value or benefit based on their utility, capability and quality characteristics.

The purpose of a model covers a variety of different intentions and aims. Typical purposes are:

perception support for understanding the application domain,
explanation and demonstration for understanding an origin,
preparation to management and handling of the origin,
optimisation of the origin,
hypothesis verification through the model,
construction of an artifact or of a program,
control of parts of the application,
simulation of behaviour in certain situations, and
substitution for a part of the application.

Depending of the purpose we shall use different models.

Models are author-driven and addressee-oriented. Therefore, the association between an origin and the model as a mapping is given in Figure 17.1.

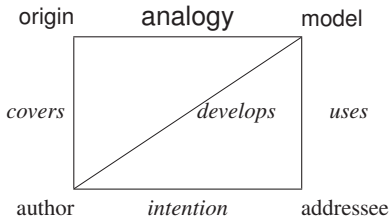


Fig. 17.1 The origin-model-author-addressee relationship for models

A *model* is typically a schematic description of a system, theory, or phenomenon of an origin that accounts for known or inferred properties of the origin and may be used for further study of characteristics of the origin. *Conceptual modelling* aims to create an abstract representation of the situation under investigation, or more precisely, the way users think about it. Therefore conceptual models enhance models by concepts that are commonly sharing within a community or at least between the stakeholders involved into the modelling process.

This chapter extends the theory of conceptual models, conceptual modelling and modelling act proposed by [20] and systematises the four main dimensions of models: purpose, mapping, language, and value. It is based on an explicit application of concepts and constructs of languages.

The value of a model is given by the objective value and by the subjective value.

Models as enduring, justified and adequate artifacts: The artifact can be qualified as an 'objective' model, if

1. the artifact is adequate by certain notion of 'adequacy',
2. is reusable in a rule system for new models and refinement of models and
3. is not equivalent to models, which can be generated with the aid of facts or preliminary models in the particular inventory of models by a rule system.

Models as the state of comprehension or knowledge of a user: Models are used for comprehension of a user or stakeholder. Therefore, a model can be understood as the knowledge of a user. Different kinds of `to know` are:

1. The state or fact of knowing.
2. Familiarity, awareness, or understanding gained through experience or study.
3. The sum or range of what has been perceived, discovered or learned.
4. Learning; erudition: teachers of great knowledge.
5. Specific information about something.
6. Carnal knowledge.

Therefore, we conclude that it is necessary to deliver models as enduring, justified and adequate artifacts to users depending on context, users demands, desiderata and intention, whereby these aspects are supported by the environment, the profile and tasks of the users. The tasks of users require a special model quality.

17.1.1 Artifacts, Concepts and Intentions

17.1.1.1 The Conceptual Model Space

At the same time we may distinguish four different aspects of conceptual models. Conceptual model use concepts. Therefore, the model space is characterised through (1) its origin, (2) its concepts, (3) its representation of model elements, and (4) its comprehension by users or stakeholders involved. Model elements cannot be considered in an isolated form. For this reason we imagine to use *model chunks* as a suite of model elements consisting of images of pieces observed for the origin, concepts, representations and comprehension. These aspects are interdependent from each other. Figure 17.2 displays the conceptual model space space.

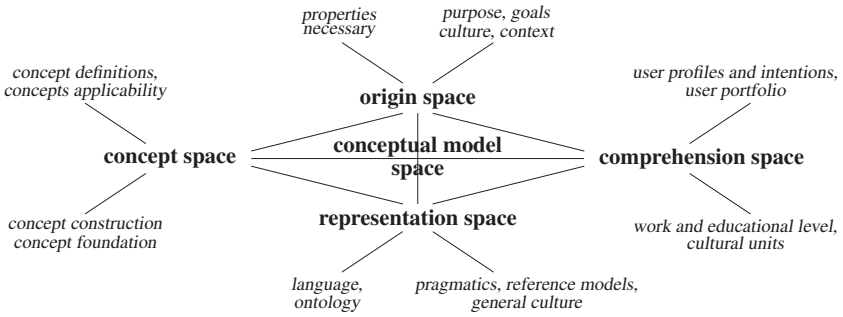


Fig. 17.2 The four aspects of the model space: Origin aspect through properties, foundation aspect through concepts, representation aspect through language, user aspect through user comprehension

17.1.1.2 Intentions Driving Modelling

Modelling and especially conceptual modelling is not yet well understood and misinterpreted in a variety of ways. The first goal of the chapter is to overcome some myths of conceptual modelling such as:

1. Modelling equals documentation.
2. You can think everything through from the start.
3. Modelling implies a heavyweight software process.
4. You must "freeze" requirements and then you can start with modelling.
5. Your model is carved in stone and changes only from time to time.
6. You must use a CASE tool.
7. Modelling is a waste of time.
8. The world revolves around data modelling.
9. All developers know how to model.
10. Modelling is independent on the language.

The second goal of this chapter is the development of a framework to modelling. Modelling is based on an explicit choice of languages, on application of restrictions, on negotiation and on methodologies. Languages are defined through their syntactics, their semantics, and their pragmatics. We typically prefer inductive expression formation based on alphabets and behaviour defined on expressions. Restrictions depend on logics (deontic, epistemic, modal, belief, preferences) and use shortcuts, ambiguities, and ellipses. Negotiation support management or resolution of conflicts and the development of strategies to overcome these strategic, psychological, legal, and structural barriers. Development methodology are based on pragmatism and on paradigms. Since modelling is an activity that involve a number of actors the choice of languages becomes essential. Modelling is a process and based on *modelling acts*. These modelling acts are dependent from the purpose of modelling itself. Therefore we can distinguish different modelling acts such as understand, define, conceptualise, communicate, abstract, construct, refine, and evaluate. De-

pending on the purpose of model development we might use modelling act such as construct and evaluate as primary acts.

The third goal of this chapter is to draw attention to explicit consideration of modelling properties both for the models themselves and for the modelling acts. This side of conceptual modelling is often only considered in an implicit form. The modelling process is governed by goals and purposes. Therefore, we must use different models such as a construction model, a communication model or a discussion model. Modelling is restricted by the application context, the actor context, the system context and the theory and experience context. These kinds of context restrict the model and the modelling process.

17.1.2 Dimensions of Models and Modelling

17.1.2.1 Main Dimensions of Modelling

We use commonalities observed above for Computer Science for an introduction of *main dimensions* of models and modelling that considers

purpose ("wherefore") of models and modelling with the intentions, goals, aims, and tasks that are going to be solved by the model,

mapping ("whereof") with a description of the solution provided by the model, the characterisation of the problem, phenomena, construction or application domain through the model,

language ("wherewith") with a careful selection of the the carrier or cargo [10] that allows to express the solution, the specification of the world or the construction, and

value ("worthiness") of a model by explicit statement of the internal and external qualities, and the quality of use, e.g. explicit statement of invariance properties relating the model to its associated worlds or by preservation properties that are satisfied by the model in dependence on the associated worlds.

These main dimensions of models and modelling govern the model and the modelling acts. There are extended by secondary dimensions that are used to shape and to adapt the model. We are going to discuss these dimensions in the sequel after a discussion of the *ruling dimension*: the purpose dimension.

The task of model development is never completed (p´tamoi rh´ousi 'the rivers flow'). Models are changing artifacts due to changes imposed by

scope insight for conscious handling of restriction, capabilities, opportunities,

guiding rules for convenience, for completion, refinement, and extension,

development plans for partial delivery of models, partial usage and deployment,

theories supporting development of models,

quality characteristics for model completion, model evolution, model engineering, and

mappings styles for mapping models among abstraction layers.

17.1.2.2 The Purposes Dimension

The purpose dimension is ruling the development of models and the application of models. The main reason for using a model is to provide a solution to a problem. We thus may describe the purpose by characterisation of the solution to the problem by the model. We may distinguish a number of concerns such as

the impact of the model ("whereto") for a solution to a problems,
the insight into the origin's properties ("how") by giving details how the worlds is structured or should be structured and how the functionality can be described,
restrictions on applicability and validity ("when") of a model for some specific solutions, for the validity interval, and the lifespan of a model,
providing reasons for model value ("why") such as correctness, generality, usefulness, comprehensibility, and novelty, and
the description of functioning of a model ("for which reason") based on the model capacity.

This general characterisation of purposes of models can be specialised for database and information system models The main purposes of information system models are given within Gregor's taxonomy [6]:

I. Analysis: Says what is.
The model does not extend beyond analysis and description. No causal relationships among phenomena are specified and no predictions are made. It thus provides a description of the phenomena of interest, analysis of relationships among those constructs, the degree of generalisability in constructs and relationships and the boundaries within which relationships, and observations hold.

II. Explanation: Says what is, how, why, when, and where.
The model provides explanations but does not aim to predict with any precision. There are no testable propositions. The model provides an explanation of how, why, and when things happened, relying on varying views of causality and methods for argumentation. This explanation will usually be intended to promote greater understanding or insights by others into the phenomena of interest.

III. Prediction: Says what is and what will be.
The model provides predictions and has testable propositions but does not have well-developed justificatory causal explanations. It states what will happen in the future if certain preconditions hold. The degree of certainty in the prediction is expected to be only approximate or probabilistic in IS.

IV. Explanation and prediction: Says what is, how, why, when, where, and what will be.
The model provides predictions and has both testable propositions and causal explanations. A special case of prediction exists where the model provides a description of the method or structure or both for the construction of an artifact (akin to a recipe). The provision of the recipe implies that the recipe, if acted upon, will cause an artifact of a certain type to come into being.

V. Design and action: Says how to do something.
The model gives explicit prescriptions (e.g., methods, techniques, principles of form and function) for constructing an artifact.

Based on this characterisation of the purpose we infer a number of *requirements* to languages used for modelling and to modelling methodologies:

Means of representation: The model must be represented physically in some way: in words, mathematical terms, symbolic logic, diagrams, tables or graphically. Additional aids for representation could include pictures, models, or prototype systems.

Constructs: These refer to the phenomena of interest in the model (Dubins "units"). All of the primary constructs in the model should be well defined. Many different types of constructs are possible: for example, observational (real) terms, theoretical (nominal) terms and collective terms.

Statements of relationship: These show relationships among the constructs. Again, these may be of many types: associative, compositional, unidirectional, bidirectional, conditional, or causal. The nature of the relationship specified depends on the purpose of the model. Very simple relationships can be specified.

Scope: The scope is specified by the degree of generality of the statements of relationships (signified by modal qualifiers such as "some", "many", "all", and "never") and statements of boundaries showing the limits of generalizations.

Causal explanations: The model gives statements of relationships among phenomena that show causal reasoning (not covering law or probabilistic reasoning alone).

Testable propositions (hypotheses): Statements of relationships between constructs are stated in such a form that they can be tested empirically.

Prescriptive statements: Statements in the model specify how people can accomplish something in practice (e.g., construct an artifact or develop a strategy).

17.1.2.3 The Artifact Dimension

The main product of modelling is the model, i.e. an artifact that is considered to be worth for its purpose by the author. The model can, for instance, be used for the description of the world of origins or for the prescription of constructions. There are a number of explicit choices an author makes and that rule application of models. Modelling of information systems

depends on the *abstraction layer*, e.g. requirements, specification, realisation or implementation layer,

depends on chosen *granularity and precision* of the work product itself,

depends on *resources* used for development of a model such as the language,

depends on *level of separation of concern* such as static/dynamic properties, local/global scope, facets,

depends on *quality properties of the input* , e.g. requirements, completeness, conciseness, coherence, understandability,

depends on decomposition of the work products in ensembles of sub-products, and
satisfies quality characteristics such as quality in use, internal quality, and external quality.

17.1.2.4 The User Dimension

A number of users are involved into the development of models. The user dimension thus reflects intentions, the understanding, the comprehension and other characteristics of users in a variety of roles, e.g.,

the role of an *author ("by whom")* that results in reflections of the educational level, application of templates, pattern or reference models,
the role of an *addressee ("to whom")* that restricts the utilisation of the model or that supports the extended application beyond the purpose originally intended, and
the role of *broad public ("whichever")* that develops a common understanding of the model depending on the group or the culture of the public.

Users are different and thus modelling has different results because of

attitudes of users and their preferences
the *ability* to understand, to model, to reason, to survey, to communicate with others, to analyse, to construct systems, to validate or to verify or to test models, to use or develop documentations
mastering of complexity, improvements, and realisations,
knowledge, skills, competency of users for representing world or for coping with representations,
restricted expressivity due to restricted leads or due to human preference of local reasoning instead of global consideration of all properties of an artifact,
experience to cope with varieties of problem solutions through generic problem solving, and
referential solutions to be used for solution of similar problems together with refinement of the given approach.

One important relationship among the users is the form of partnership during the development or application of models The partnership is characterised

by roles during activities such as stakeholder, developer, consultant, supplier, contractor, documentation developers, or finally business user,
by the practised collaboration partnership based on communication acts, cooperation business processes, and coordination agreements,
by the teamwork during all activities with separation of different tasks,
by historical people such as teachers, legacy (better heritage) developers, coders, and
builders of earlier models.

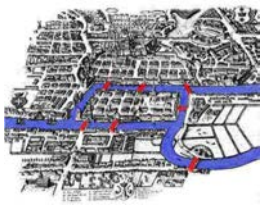
Finally, the user dimension imposes an important restriction to the development, the application, the understanding of models: Models tend to be too large for a singleton person.

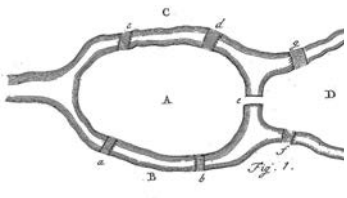
17.1.2.5 The Domain Dimension

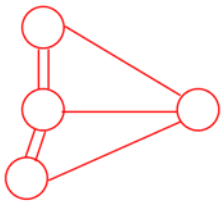
The domain dimension clarifies

the *domain depending on models purpose ("for what")* such as an application domain, properties reflected or neglected,
the *scope to specific elements ("what")* that are considered to be typical and whose properties should be reflected,
the *attention within the domain depending on models purpose ("where")* that limits the model to the 'normal' aspects,
the *orientation of the domain ("wherefrom")* that restricts the attention and the directions for the current activities supported by the model,
the sources for origins or the infrastructure considered ("whence") for the model, and
the *restrictions of the world ("wherein")* associated with the model.

A typical influence of the application domain can be illustrate by an example in [10]. Areas in Königsberg are connected through bridges. The question is whether there is a path that uses each bridge but only once. Such path is called Euler path.


application domain


topographical model


graph-theory model

The two models display the same problem as in the original domain. We might also use a tree model that enumerates each starting point and associates a node with its predecessor and potential next point if there is an unused bridge. This model is inadequate for the general problem whether there is an euler path within a topographical model. The main quality property for the models is the preservation of the Euler path problem.

17.1.2.6 The Context Dimension

The context dimension is typically used for restricting a model to a specific scope and thus limits the general utilisation of models. It additionally requires an explicit consideration of these restrictions if the model is used outside its main application

area. Context abstraction is a useful vehicle for restricting attention. Typical specific context restrictions to models are

the *worlds ("whereat")* considered for the model such as the world that is currently accepted, the world that will be never considered, and the world that might be considered in future in dependence on the model value,

the *background knowledge ("whereabout")* that forms the model and limits the model, and

envisioned *evolution pathes ("whither")* for the adaptation of model to future requirements.

17.1.3 Postulates of Modelling

17.1.3.1 General Properties of Models

This discussion can be summarised in a number of postulates that are of importance for models, modelling, and modelling acts.

Mapping property: Each model has an origin and is based on a mapping from the origin to the artifact.

Truncation property: The model lacks some of the ascriptions made to the original and thus functions as an Aristolean model by abstraction of irrelevant.

Pragmatic property: The model use is only justified for particular model users, tools of investigation, and period of time.

Amplification property: Models use specific extensions which are not observed for the original,

Distortion property: Models are developed for improving the physical world or for inclusion of visions of better reality, e.g. for construction via transformation or in Galilean models.

Idealisation property: Modelling abstracts from reality by scoping the model to the ideal state of affairs.

The first three properties are based on Stachowiaks theory of models [16, 12]. The fourth property has been formulated in [17]. The fifth property has been discussed in [9]. The sixth property has been developed within Natural Sciences, e.g. Chemistry.

17.1.3.2 Prescription by Models and Description for Models

This association between origin and artifacts is depicted in Figure 17.3.

In a similar form we may describe the application of models for construction of other artifacts such as software and hardware. In this case, the reflection of the postulate is given in Figure 17.4.

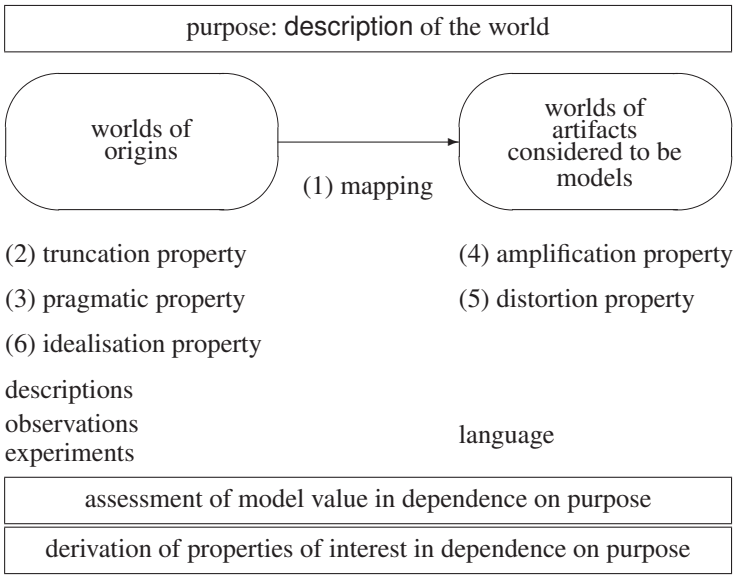


Fig. 17.3 The association of worlds and resulting postulates of models

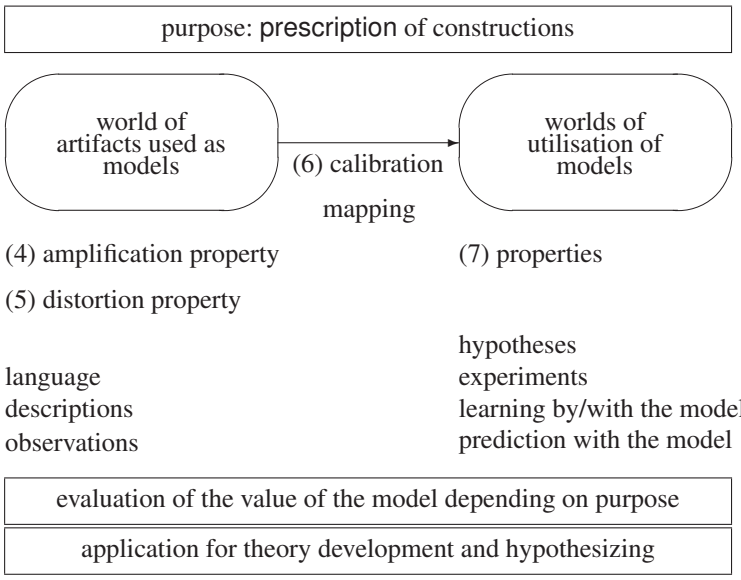


Fig. 17.4 The association of models with artifacts constructed with models as a blueprint

17.1.3.3 The Model Capacity

The model

- is based on an *analogy* of structuring, functionality, or behaviour,
- satisfies certain *model purposes*, and
- provides a simple handling or *service* or consideration of the things under consideration.

Any model is therefore characterised by a *model capacity* that describes

- how the model provides some understanding of the origin or can be used depending on the purpose,
- how the model provides an explanation of demonstration through auxiliary information and thus makes the origin or the associated elements easier or better to understand,
- how the model provides an indication and facilities for making properties viewable,
- how the model allows to provide variations and support optimisation,
- how the model support verification of hypotheses within a limited scope,
- how the model supports construction of technical artifacts,
- how the model supports control of things in reality, or
- how the model allows a replacement of things of reality and acts as a mediating means.

17.1.3.4 Resulting Restrictions to be Accepted by Stakeholders

Models are governed by their purpose. They may support this purpose or not. They have a value and may thus be used depending on their capacity.

Prohibition of *estrangement*: Models serve a purpose and cannot be used in general outside the scope of the purpose.

17.1.4 Artifacts and Models

The four aspects of the conceptual model space in Figure 17.2 are interwoven. Models use artifacts. Models have their specific representation. Models are supported by conceptualisations. The interrelationship between models, representations and concepts should be very flexible. We can assume that models may use different representations or artifacts. Artifacts may contain sub-artifacts. Conceptual models are based on concepts. Concepts may be typical for a model within a certain degree of typicality. Concepts may consist of sub-concepts. Therefore, we may associate a model with one concept that has its sub-concepts. We assume that concepts are independent on representations. Additionally, we may assume that representations are dependent on the language and some ontology to be used. They are typically

commonly accepted or shared within a community or culture. This understanding leads to a structure displayed in Figure 17.5.

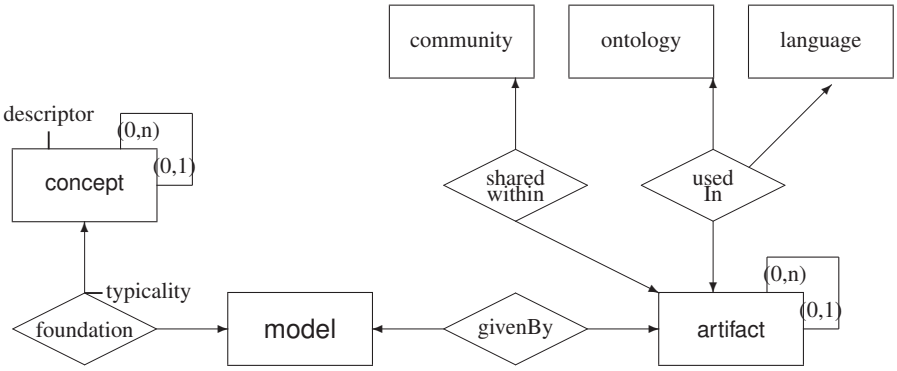


Fig. 17.5 The association between artifacts, representations, and concepts

17.2 The Theory of Conceptual Models

17.2.1 Conceptual Models and Languages

17.2.1.1 The Language Dimension

Models are represented by *artifacts* that satisfy the pragmatic purposes of users. We restrict our discussion within this subsection to formal languages that are typically used for conceptual models. In this case, artifacts are linguistic expressions that describe the model. Linguistic expression are built within a language with some understanding. Therefore, artifacts use syntax, semantics and pragmatics built within the chosen language.

Semantic annotation in current content management systems is usually restricted to preselected ontologies and parameter sets. Rich conceptual data models are only available in more sophisticated systems. They are adapted to certain application domains incorporate preselected and tailored ontologies.

The model-artifact association is agreed within a community. This community is based on a web of knowledge of their members (see Chapter **??**).

17.2.1.2 Languages used for Representation of Models

Languages are the carrier for models. We may accept a logics approach to semiotics and define the language similar to Chapter **??**. Each constructive language is based on a signature, on a set of base items, and a set of constructors. The language consists of words that are allowed due to well-formedness constraints. Languages $\mathscr{L}$ are used for a number of reasons, e.g. reasoning, representation, illustration, etc. These reasons are driven by the model purposes in our case.

We may now use a subset of words and accept those as *postulates* for a model. Each model to be considered faithful or useful must satisfy these postulates. We may develop different artifacts as a potential models. We are however interested in some properties that these models must satisfy. We therefore develop a general understanding of artifacts that are used for models within a language. Postulates must be explicitly given. They might be changed whenever the purpose of modelling is changing. They do not restrict models to one artifact. Instead we might also use a number of artifacts in parallel. Figure 17.6 displays the relationship between an artifact and its postulates and properties.

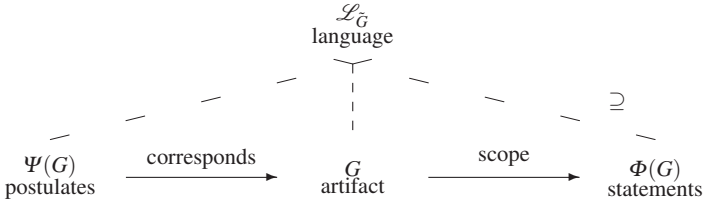


Fig. 17.6 Artifacts with a language, their properties and postulates

Constructive languages thus support

- to prescribe postulates that restrict the judgement that an artifact can be accepted as a model,
- to scope our attention to those artifacts that can be considered for a model or for parts of a model, and
- to orient the user on certain properties that are of interest for the purpose of modelling.

This approach is very general. It can be applied in many areas. Consider, for instance, the following table:

$\mathscr{L}_{\tilde{G}}$	$\Psi(G)$	corresponds	G	scope	$\Phi(G)$
logics	axioms	satisfy	structure	satisfy	essential properties
$\mathbb{N}$	Peano axioms	satisfy	standard model	derivable	Peano arithmetics
empirism	postulates	accepted	artifact	supports	observation
technics	construction requirements	enforce	product	has	properties

This approach also carries classical approaches used in mathematical logics:

$\mathscr{L}_{\tilde{G}}$	$\Psi(G)$	corresponds	G	scope	$\Phi(G)$
logics	axioms	satisfy	structure	consider	essential properties of G
logics	axioms	satisfy	structure	satisfy	relevant theorems of $(\mathscr{L}_{\tilde{G}}, \Psi(G))$

We, therefore, may define a theory by the pair $(\mathscr{L}_{\tilde{G}}, \Psi(G))$. The model class $Mod^{\mathscr{L}_{\tilde{G}}}(\Psi(G))$ is defined to be the set of all structures that satisfy $\Psi(G)$. A structure G is a model of $\Psi(G)$ and thus $G \in Mod^{\mathscr{L}_{\tilde{G}}}s(\Psi(G))$. A theory $Th(\mathscr{K}) \subseteq \mathscr{L}_{\tilde{G}}$ is given for a class $\mathscr{K}$ of structures and consists of all language expressions that are satisfied by each of the structures in $\mathscr{K}$.

The same approach can also be used for conceptual modelling:

$\mathscr{L}_{\tilde{G}}$	$\Psi(G)$	corresponds	G	scope	$\Phi(G)$
database	requirements	realise	DB schema	satisfy	integrity constraints
workflow	requirements	realise	WF schema	satisfy	integrity constraints

Therefore ingredients used for modelling of databases, information systems and workflow systems are languages, restrictions, negotiations for the property to be a model, and methodologies for artifact development. Languages are given with syntactics, semantics, and pragmatics. We typically use inductive expression formation based on alphabets. It also supports the description of behaviour defined on expressions. Restrictions depend on logics to be used, e.g., first-order hierarchical predicate logics [18], deontic logics, epistemic logics, modal logics, logics for belief reasoning of for preference derivation. Negotiations provide a means to identify, define analyse barriers and manage or resolve conflicts. Methodologies of development are based on engineering approaches and are guided by certain pragmatism and a number of paradigms.

17.2.1.3 Principles of Language Use

Languages may however also restrict modelling. This restriction may either be compensated by over-development of language components or by multi-models. Over-development of language components has been observed within the theory of integrity constraints in the relational model of data. More than 95 different and necessary classes of integrity constraints have been developed. Multi-modelling is extensively used for UML. The Sapir-Whorf hypothesis [22] results in the following principle:

Principle of linguistic relativity: Actors skilled in a language may not have a (deep) understanding of some concepts of other languages. This restriction leads to problematic or inadequate models or limits the representation of things and is not well understood.

The principle of linguistic relativity is not well understood. Therefore, we illustrate this principle by a discussion that highlights the deficiencies we need to overcome.

17.2.1.4 The Matter of Language Choice

Let us consider a well-known example: *traffic light control*. Given a crossroad, e.g. consisting of two streets (north-south, east-west) with an intersection and of traffic lights that direct traffic. We assume at the first glance that traffic lights might switch from red to green and from green to red. We also might assume that both opposite cross lights show the same colour. Software engineering approaches, Petri net approaches, process algebra approaches etc. typically start with a model for each cross light. Next the interdependence among the state changes is either modelled through integrity constraints or through implicit modelling constructs. The best solution we know so far is the Petri net solution depicted in Figure 17.7. It uses an external timer and switches between the directions.

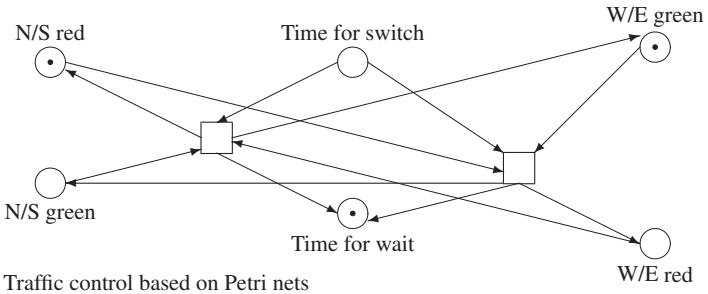


Fig. 17.7 Traffic control based on Petri nets

This model neither scales nor has a good development, internal, or dynamic quality. The extension to yellow colour is an intellectual challenge as well the extension to more flexible directing. This example is typically chosen due to everyday life experience of the students despite its complete inadequacy. This pitfall has already discussed in [8] who tried to find a better solution based on state change diagrams and failed due to complex integrity constraints. Implementations neglect this solution and implement a completely different solution.

The main reason for the poor quality and the conceptual and implementation inadequacy is its wrong attitude, wrong scope, wrong abstraction, and wrong granularity.

Explicit assumptions can also be derived for the traffic light control application. We first need to decide whether the analogy to real-life is based on the behaviour of the entire system or on the combined behaviour of the behaviour of components. This distinction directly implies a choice between a model that represents the entire application as one system and the components as its elements (*local-as-view model*) and a model that combines local models to a global one (*global-as-view model*). All conceptual solutions known in literature use the global-as-view model. In this case

state tables and (ASM) state transfer rules like the following ones are used:

Controller	location	state	clock	reset	switch
e	...	...	...	...	...

`if` Switch(e) `then` UPDATE(e,collocated(e)); CHANGESWITCH(e) .

These states and rules may obey a number of rather complex integrity constraints.

We might prefer the local-as-view approach. States reflect the entire state of the crossroad, i.e. *NSredEWgreen, NSredEWred, NSgreenEWred*. The last state reflects that the north-south direction is open and the east-weast direction is closed. We might add the state *NSredEWred* for representation of the exception state and the state *NSnothingEWnothing* for the start and the end state. The state *NSgreen-EWgreen* is a conflict state and thus not used for the model.

The other decisions discussed in this section can now made in a similar manner. We choose a full controller for all lights. We might however choose a local controller for each cross light. In this case, the local controller is nothing else than a *view* on the global schema. The model we propose supports simulation as well as understanding, reasoning, variation and extension, optimisation and technical artifacts. The workmanship includes also a collection of extensions that seems to be probable such as people calling a state change, exceptional situations, yellow lights, specific directions etc. The local schemata are based on views and on the master-slave principle. Update is central and display is local.

This model also allows to explicitly specify which states are never under consideration, which states are a 'must' and which states are used for later extensions. We further assume that reality can be mapped to discrete variables, clocks are based on linear time logics, control is restricted to vehicle and pedestrian direction gauge. This model extends also the real life application by adding a global, combined state. Its main advantage is however that the context conditions for correct traffic lights for all coexisting directions are directly coded into the model domain space and thus do not need any explicit support.

The local-as-view model is based on a *two-layer architecture* that uses a global schema and local view schemata. The extended ER model [18] provides a number of opportunities for the representation of hierarchies. A typical hierarchy in our traffic light application is the specialisation hierarchy for states. Since states can be multiply classified depending on the day time and the week day we might choose the bulk representation for the classification of types through a *StateKind* instead of explicit specialisation types. State changes may also be classified in a similar way. We might however prefer to separate calls for state change made by pedestrians and triggering of state changes through a times. Based on these choices we derive modelling activities for the database schemata and workflow rules. We explicitly specify properties and binding among the global and local schemata, e.g. master-slave binding.

The given application can be specified through different modelling concepts. These modelling concepts provide a number of alternatives and a number of opportunities. Therefore the ER schema in Figure 17.8 represents one of the possible schemata for the global schema. The state changes and the pedestrian calls are not recorded after they have been issued. The scheduler is based on this schema and

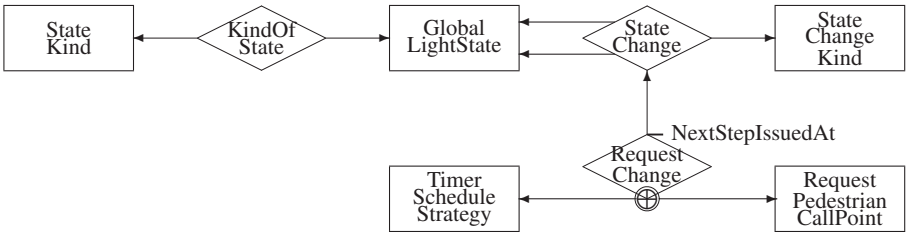


Fig. 17.8 The traffic light support database schema

might use workflow diagrams, trigger rules or ASM rules [2] for specification of BPMN diagrams. We can use a generic pattern approach that supports extensions, e.g. for kinds of states and kinds of state changes. Typical examples are the following:

CHANGEACTION := getState; choosePossibleStateChange(state);
apply(possibleStateChange(state)
ALARMACTION := `on` alarm changeStateToErrorState
CLOCK := `on` tick observeWhetherChangeRequired
NORMALACTION := `if` change = true then CHANGEACTION
PEDESTRIANCALL := `on` callAtPoint(cp) CHANGENEXTSTEPISSUEDAT(cp).

In a similar form we specify views for local display.

17.2.1.5 Pragmatics that cannot be Neglected

While syntax and semantics of language expressions has been well explored, its pragmatics apart from the use of metaphors has not. Pragmatics is part of semiotics, which is concerned with the relationship between signs, semantic concepts and things of reality. This relationship may be pictured by the so-called semiotics triangle. Main branches of semiotics are *syntactics*, which is concerned with the syntax, i.e. the construction of the language, *semantics*, which is concerned with the interpretation of the words of the language, and *pragmatics*, which is concerned with the current use of utterances by the user and context of words for the user. Pragmatics permits the use of a variety of semantics depending on the user, the application and the technical environment. Most languages defined in Computer Science have a well-defined syntax; some of them possess a well-defined semantics; few of them use pragmatics through which the meaning might be different for different users.

Syntactics (often called syntax) is often based on a constructive or generative approach: Given an alphabet and an set of constructors, the language is defined as the set of expressions that can be generated by the constructors. Constructions may be defined on the basis of grammatical rules.

Semantics of generative languages can be either defined by meta-linguistic semantics, e.g. used for defining the semantics of predicate logics, by procedural or referential semantics, e.g. operational semantics used for defining the semantics of

programming languages, or by convention-based semantics used in linguistics. Semantics is often defined on the basis of a set of relational structures that correspond to the signature of the language.

Pragmatics has to be distinguished from pragmatism. Pragmatism means a practical approach to problems or affairs. Pragmatism is the "balance between principles and practical usage". Here we are concerned with pragmatics, which is based on the behaviour and demands of users, therefore depends on the understanding of users.

Let us consider an example for a well-known class of constraints in databases. A similar observation can be made for multivalued, join, inclusion, exclusion and key dependencies. Functional dependencies are the most known class of database constraints and commonly accepted. They are one of the most important class of equality-generating constraints.

Given a type R and substructures X, Y of R.

The functional dependency $R : X \longrightarrow Y$ is valid in R^C if $o|_Y = o'|_Y$ whenever $o|_X = o'|_X$ for any two objects o, o' from R^C.

Functional dependencies carry at least five different but interwoven meanings. The notion of the functional dependency is thus overloaded. It combines different properties that should be separated:

Explicit declaration of partial identification: Functional dependencies are typically explicitly declaring a functional association among components of types. The left hand attribute uniquely identify right side attributes, i.e. $X \xrightarrow{Ident} Y$. Identification can either be based on surrogate or on natural attributes [1].

Tight functional coupling: Functional dependencies may also be numerical constraints. We denote such constraints by i.e. $X \xrightarrow{Num} Y$. Another denotation is based on cardinality constraints [18].

Semantic constraint specific for the given application: Constraints may be stronger than observed in usual life since the application has a limited scope and allows us to strengthen the constraint. In this case, constraints restrict the application only to those cases in which the left side has only one associated right side value despite that this restriction may not be valid for any application. We denote this case by $X \xrightarrow{Sem} Y$

Semantical unit with functional coupling: *Semantical units* are those reducts of a type that are essential in the given application. Their components cannot be separated without loosing their meaning. Semantical units may have their inner structure. This structure tightly couples dependent object parts to those that are determining them [18]. We denote this coupling by $X \xrightarrow{Unit} Y$.

Structural association among units: Semantical units may allow a separation of concern for certain elements. Their separation supports a more flexible treatment while requiring that the dependent part cannot exist without the determining part. If this dependence is functional we may represent such by the constraint $X \xrightarrow{Struct} Y$.

17.2.2 Concepts and Models

Concepts are the basis for conceptual models. They specify our knowledge what things are there and what properties things have. Concepts are used in everyday life as a communication vehicle and as a reasoning chunk. Concepts can be based on definitions of different kinds. Therefore our goal for the development of a theory of conceptual modelling and of conceptual models can only be achieved if the conceptual model definition covers any kind of conceptual model description and goes beyond the simple textual or narrative form.

A general description of concepts is considered to be one of the most difficult tasks. We analysed the definition pattern used for concept introduction in mathematics, chemistry, computer science, and economics. This analysis resulted in a number of discoveries:

- Any concept can be defined in a variety of ways. Sometimes some definitions are preferred over others, are time-dependent, have a level of rigidity, are usage-dependent, have levels of validity, and can only be used within certain restrictions.
- The typical definition frame we observed is based on definition items. These items can also be classified by the kind of definition. The main part of the definition is a tree-structured structural expression of the following form

  SpecOrderedTree(StructuralTreeExpression
  (DefinitionItem, Modality(Sufficiency, Necessity),
  Fuzziness, Importance, Rigidity,
  Relevance, GraduationWithinExpression, Category))) .
- Concepts typically also depend on the application context, i.e. the application area and the application schema. The association itself must be characterised by the kind of association.

Concepts are typically hierarchically ordered and can thus be layered. We assume that this ordering is strictly hierarchical and the concept space can be depicted by a set of concept trees. A concept is also dependent on the community that prefers this concept. A concept is typically given through an embedding into the knowledge space of users involved. The schema in Figure 17.9 displays the general structure for content definition. This schema also covers all aspects discussed in [13]. This schema extends the relationship between artifacts, representations and concepts introduced in Figure 17.5.

Concept gathering can be understood as a technique which combines concept representation [5, 13, 19] and (algorithmic) learning approaches.

A concept gathering system is *based on*

a *set of concepts and available experience* $\mathscr{C}$,
a *set of domain knowledge* $\mathscr{D}$,
a *set of representable meta knowledge* $\mathscr{M}$,
a *set of learning goals* $\mathscr{G}$, and
a *set of representable hypotheses* $\mathscr{H}$.

The set of representable knowledge and concepts is denoted by $\mathscr{R} = \mathscr{C} \cup \mathscr{D} \cup \mathscr{M} \cup \mathscr{G} \cup \mathscr{H}$.

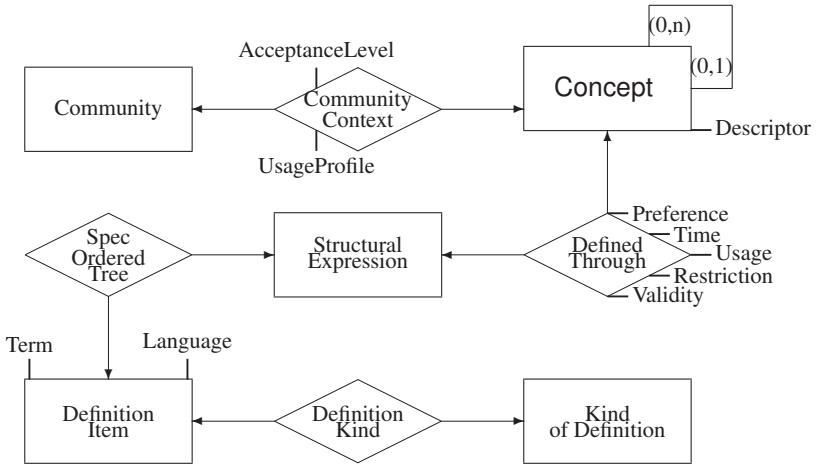


Fig. 17.9 The main schema for Concept Definition and Formation

The concept gathering system $(\gamma, \lambda, \nu, \mathscr{C}, \mathscr{R})$ *consists of*

a *concept generator* $\gamma : \mathscr{C} \times \mathscr{R} \to \mathscr{C}$,
a *learning function* $\lambda : \mathscr{C} \times \mathscr{R} \to \mathscr{H}$, and
an *evaluator* $\nu : \mathscr{C} \times \mathscr{R} \to \mathscr{Q}$ where $\mathscr{Q}$ denotes set of quality characteristics.

A run of the concept gathering system *results in*

a *concept detection sequence* $C_1, C_2, ..., C_f$ with $C_i \in \mathscr{C}$ and
a *learning sequence* $R_0, R_1, R_2, ..., R_f$
with $R_i \in \mathscr{R}$ where R_0 denotes the initial knowledge and R_f denotes the final knowledge.

The run is typically recorded and is dependent on the concepts gathered so far. Additionally, the concept gathering system *records*

the *background knowledge of the user* $\mathscr{B} \subseteq \mathscr{D} \cup \mathscr{M} \cup \mathscr{G}$ and
the *actual available knowledge* $\mathscr{B} \cup \mathscr{H}'$.

17.2.3 Information Exchange of Stakeholders based on Models

Stakeholders such as the author of a model and the addressee for a model use model in a variety of ways. The main use of models is *information* (or knowledge) *exchange* among stakeholders. There are several definitions for information.

- The first category of these definitions is based on the mathematical notion of entropy. This notion is independent of the user and thus inappropriate in our project context.

- The second category of information definitions bases information on the data a user has currently in his data space and on the computational and reasoning abilities of the user. Information is any data that cannot be derived by the user. This definition is handy but has a very bad drawback. Reasoning and computation cannot be properly characterised. Therefore, the definition becomes fuzzy.
- The third category is based on the general language understanding of information.
  Information is either the communication or reception of knowledge or intelligence.
  Information can also defined as
  - knowledge obtained from investigation, study, or instruction, or
  - intelligence, news or
  - facts and data.

  Information can also be the act of informing against a person.
  Finally information is a formal accusation of a crime made by a prosecuting officer as distinguished from an indictment presented by a grand jury.
  All these definitions are too broad.

We are thus interested in a definition that is more appropriate for the internet age.

Information as processed by *humans*,

- is carried by *data*
- that is perceived or noticed, selected and organized by its receiver,
- because of his subjective human interests, originating from his instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom,
- simultaneously processed by his cognitive and mental processes, and
- seamlessly integrated in his recallable knowledge.

Therefore, information is directed towards pragmatics, whereas content may be considered to highlight the syntactical dimension. If content is enhanced by concepts and topics, then users are able to capture the meaning and the utilisation of the data they receive. In order to ease perception we use *metaphors* or simply *names* from a commonly used namespace. Metaphors and names may be separated into those that support perception of information and into those that support usage or functionality. Both carry some small fraction of (linguistic) semantics.

The *information transfer* from a user A to a user B depends on the users A and B, their abilities to send and to receive the data, to observe the data, and to interpret the data. Let us formalise this process. Let s_X denote the function user by a user X for data extraction, transformation, and sending of data. Let r_X denote the corresponding function for data receival and transformation, and let o_X denote the filtering or observation function. The data currently considered by X is denoted by D_X. Finally, data filtered or observed must be interpreted by the user X and integrated into the knowledge K_X a user X has. Let us denote by i_X the binary function from data and knowledge to knowledge. By default, we extend the function i_X by the time t_{i_X} of the execution of the function.

Thus, the data transfer and information reception (or briefly information transfer) is formally expressed it by

$$I_B = i_B(o_B(r_B(s_A(D_A))), K_B, t_{i_X}) .$$

In addition, time of sending, receiving, observing, and interpreting can be taken into consideration. In this case we extend the above functions by a time argument. The function s_X is executed at moment t_{s_X}, r_X at t_{r_X}, and o_X at t_{o_X}. We assume $t_{s_A} \leq t_{r_B} \leq t_{o_B} \leq t_{i_B}$ for the time of sending data from A to B. The time of a computation f or data consideration D is denoted by t_f or t_D, respectively. In this extended case the information transfer is formally expressed it by

$$I_B = i_B(o_B(r_B(s_A(D_A, t_{s_A}), t_{r_B}), t_{o_B}), K_B, t_{i_B}) .$$

The notion of information considers senders, receivers, their knowledge and experience. Figure 17.10 displays the multi-layering of communication, the influence of explicit knowledge and experience on the interpretation.

The communication act is specified by

- the communication message with the content or content chunk, the characterisation of the relationship between sender and receiver, the data that are transferred and may lead to information or misinformation, and the presentation,
- the sender, the explicit knowledge the sender may use, and the experience the sender has, and
- the receiver, the explicit knowledge the receiver may use, and the experience the receiver has.

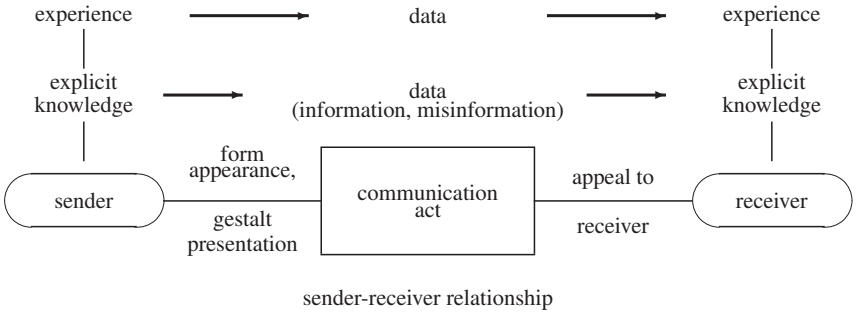


Fig. 17.10 Dimensions of the communication act

17.2.4 Mappings among Models and Originals

17.2.4.1 Modelling Supported by Mapping

So far two of the four main dimensions have been founded. Let us now consider the mapping between two worlds: source world and target world. Examples of source-target pairs are

- origins from the real real world are mapped to an artifact that is considered to be a model,
- elements of an artifact that serves as a model to a realisation of the artifact by an implementation, and
- elements of of one model to elements of another model.

The first mapping is typically based on a *description* of the origins that is represented by a model about these origins. The second mapping is typically based on a *prescription* made by the model for a realisation of the model by a technical artifact. The third mapping has been used above for the association between the topographical model and the graph model for the Königsberg bridge problem on page 9.

We observe also other pairs of such mappings depending on the purpose. For instance, documentation uses an artifact to be documented and another artifact that documents essential elements of the first artifact. It typically extends the first artifact for pragmatic rules for exploitation of the first artifact and by behavioural scenario as examples of deployment. It bases the documentation also on an idealisation of the first artifact. A similar association may be developed for the other purposes: Perception support for understanding the application domain; explanation and demonstration for understanding; preparation to management and handling of the original; optimisation of the application domain operating; hypothesis verification through the model; control of parts of the application; simulation of behaviour in certain situations; substitution for a part of the application.

We may now combine these observations with the treatment of languages introduced in Figure 17.5. We add to this treatment the logical framework. It defines for a set of artifacts that are of interest a language and a theory that can be used for reasoning on properties of the artifacts and for explicit consideration of postulates. In this case, we need to consider two languages: the language of the origin of the mapping and the language of the target of the mapping. Figure 17.11 displays the mappings between the different artifacts.

Furthermore we observe another important property of the mapping:

Principle of conservative stability of source properties: The properties of the source are relatively stable. It results in some kind of target conservativeness: Any target artifact revision that cannot be reflected already in the current set of properties of the source is entirely based on explicit changes considered for the first artifact.

Principle of consistency of mapping: Main properties of the source artifact should be stable for the target artifact.

These principles can be extended by other principles for mappings that are often assumed but not necessary:

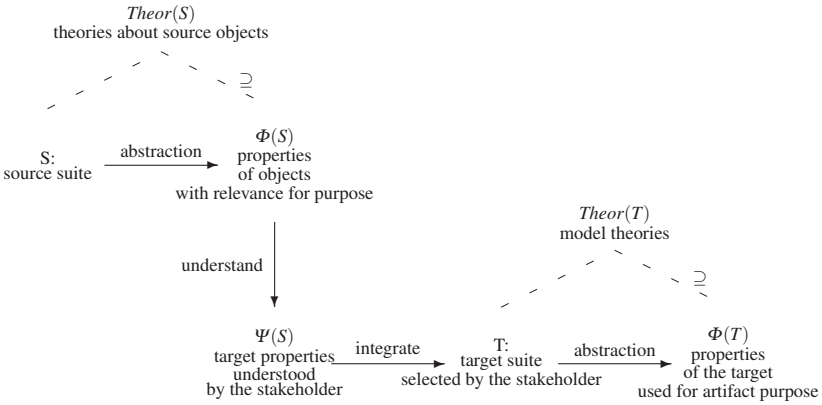


Fig. 17.11 Mappings between source artifacts and target artifacts

Conceptualization principle: Only aspects of the source artifact should be taken into account when constructing the target artifact.

95% -principle: All the relevant aspects of the source artifact should be described in the target artifact. We notice that this principle is weaker than the classical 100 % used in software engineering. It better reflects the engineering component of modelling.

Formalization principle: Target artifacts should be formalisable in order to be realisable.

Semiotic principle: Target artifacts should be easily interpretable and understandable.

Correspondence condition for knowledge representation: The target artifact should be such that the recognizable constituents of it have a one-to-one correspondence to the relevant constituents of source artifact.

Invariance principle: Target artifacts should be constructed on the basis of such entities detected for the source artifact that are invariant during certain time periods within world of the source artifact.

Construction principle: In order to construct a good target artifact it is important first to construct relevant sub-artifacts and then to search for connections between them.

The main postulate for the mapping is however the

Postulate of purpose invariance: The purpose of the modelling activity can be realised through the target artifact. It can both be considered for the source artifact as well as for the target artifact.

This postulate requires that the mapping must obey an invariance property for the purpose. It has a number of implications:

- The mapping is a realisation of an analogy property.
- It is possible to re-map properties observed for the target artifact to the source artifact if those are not caused by idealisation, distortion or amplification.

- The target artifact can also be used for other mapping with different intentions and goals.

We shall discuss specific forms of analogies below.

As an example we may refine Figure 17.11 to classical ER modelling displayed in Figure 17.12. This picture allows also to reason on the advantages and on the

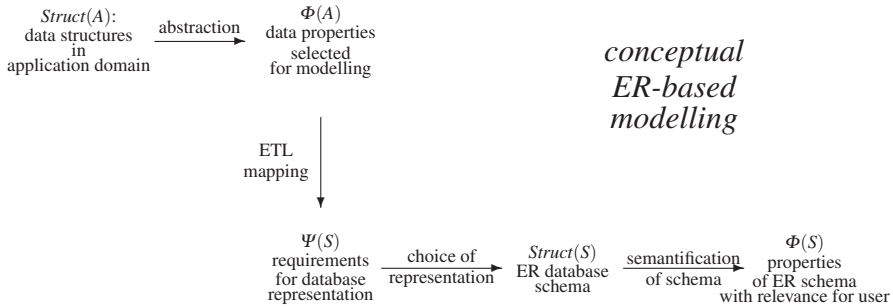


Fig. 17.12 The relationship between application domain world and Entity-Relationship modelling language world

disadvantages of the ER modelling approach.

17.2.4.2 Modelling with a Manifold of Models

Typical modelling follows a number of purpose. The UML is an example of model suites that are used at the same time. Class diagrams reflect the structuring of object sets and the functionality provided for the object sets. Object diagrams may be based on class diagrams. They may however reflect also things in the application domain as a combined set of class objects. Interaction diagrams reflect the message and control flow among objects in the first setting of object diagrams.

A similar picture is observed for models that are developed for different purposes. Consider, for instance, models that have been developed for construction of a technical artifact, for communication and discussion of properties among stakeholders, for documentation, and for analysis. Figure 17.13 display the manifold of models developed for different purposes for an origin.

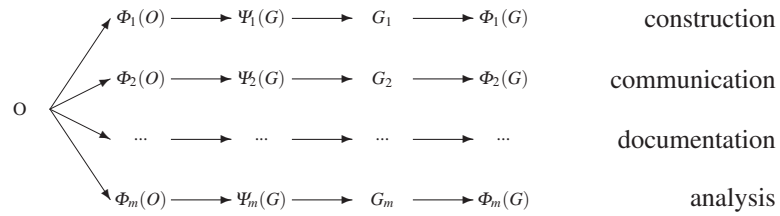


Fig. 17.13 Models reflecting different purposes

This picture displays a pitfall of multi-language modelling. The models may consider different aspects of the origin, they may contradict and they may not be integratable. For instance, if we use class diagrams, statecharts, activity diagrams, time diagrams, component diagrams, interaction diagrams and other s within a software development team then integration of different aspects might become infeasible. Therefore we may apply two rather rigid modelling restrictions: The main principles for the multi-language modelling are therefore the following the following ones:

Principle of coherence of models: Models are coherent if their common reflection of the origin is consistent, i.e. sub-models that reflect the same properties of the origin can be injective mapped to each other.

Principle of origin property completeness: Models partially reflect the same set of properties of the origin. None of the model uses properties that are different from properties that can potentially be used for another model.

Chapter **??** already considered co-evolution of models and introduced a formalism to handle coherence. *Coherence* describes a fixed relationship between the models in a model suite. Two models are coherent when each change in one of the models is propagated to the other model. This change transfer implicitly assumes that the integrity constraints of the corresponding model types remain to be valid. They are non-coherent if there is a random or changing relationship. We aim in an explicit specification of the association schema and use an explicit specification of the *collaboration* among models. For instance, the master-slave association or collaboration propagates any change of the master to its slaves. Slaves do not have any right the change the master without consensus with the master.

If we enforce these principles then the model variety can be handled in a simpler and feasible way. Figure 17.14 displays the advantages of a coherent set of models which is based on a complete set of properties of the origin. It allows to introduce a binding between these models. This binding can be mapped to a contract in the sense of Chapter **??**. The contract may be used for the derivation of constraints the different models must obey in order to be coherent.

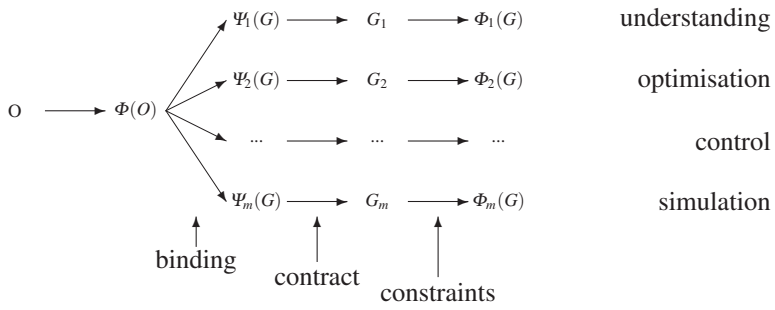


Fig. 17.14 Coherent models reflecting different purposes on a complete set of origin properties

This approach directly results in a *coordination of models* on the basis of *separation of aspects*.

17.2.5 Development Phases that use Models

17.2.5.1 Description through Models and Prescription by Models

One of the main combined purposes of models is the description of an application domain that is subsequently uses the developed model as a prescription of for realisation of a technical artifact. Conceptual modelling adds to the model a number of

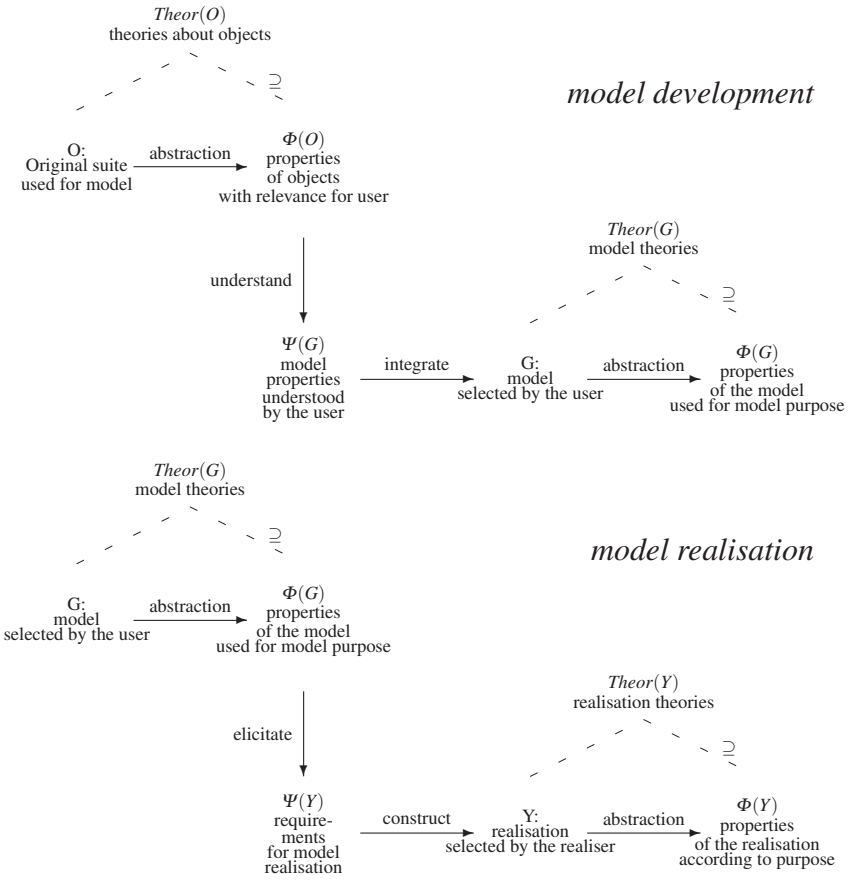


Fig. 17.15 Modelling for description of the origin and as the basis of realisation by a technical artifact

concepts that are the basis for an understanding of the model and for the explanation of the model to the user.

This two-phase development cycle of technical artifacts is the kernel of conceptual modelling of information systems and database systems. There are different other forms of this two-phase database system development. We may use the the association between the model and the application for model refinement and model evolution. Models are typically parameterised. Therefore, these parameters may be adopted to the actual or intended situation. Models are integrated during bottom-up modelling. They can be refined, optimised, validated, or improved before the realisation phase starts. Verification is typically checking the properties of a model and the the properties of a realisation. . Testing checks the relationship between properties in the application domain and properties of the realisation.

17.2.5.2 Reasoning Support for Modelling

Design science [7] has been targeting on an explicit support for the modelling process. This support includes an explicit consideration of the quality of the model, of the quality of the modelling process, and of the quality of supporting theories. We may combine the informal discussions with our approach and separate the modelling acts by the things that are under consideration. Figure 17.16 displays the different ways of working during a database systems development. We use here the two-phase model: Description followed by prescription.

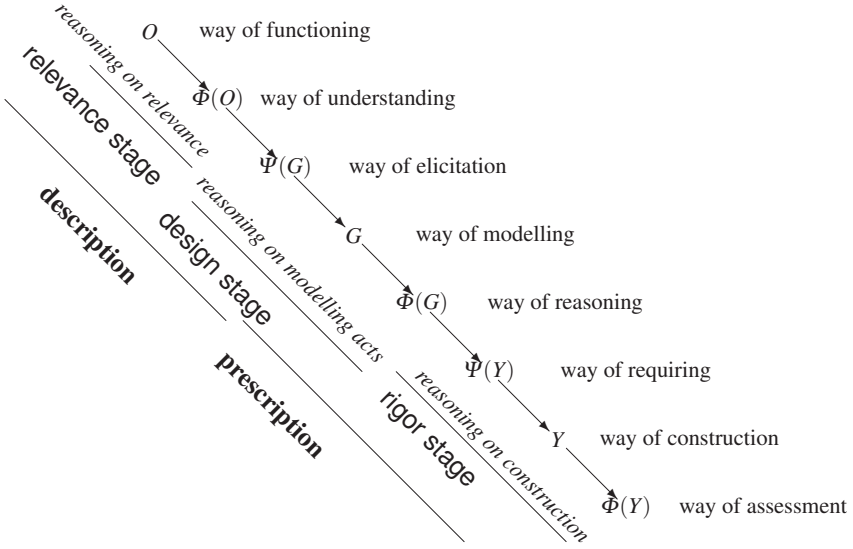


Fig. 17.16 Reasoning processes and reasoning support for description followed by prescription

These different "ways of working" characterise

- the *modelling acts* with its specifics; [20]
- the *foundation for the modelling acts* with the theory that is going to support this act, the technics that can be used for the start, completion and for the support of the modelling act, and the reasoning techniques that can be applied for each step;
- the *partner* involved with their obligations, permissions, and restrictions, with their roles and rights, and with their play;
- the *aspects* that are under consideration for the current modelling acts;
- the *consumed and produced elements of the artifact* that are under consideration during work;
- the *resources* that must be obtained, that can be used or that are going to be modified during a modelling act.

Consider, for instance, the way or requiring. It includes specific facets such as

- to command, to require, to compel, and to make someone do something with supporting acts such as communicating, requesting, bespeaking, ordering, forbidding, prohibiting, interdicting, proscribing;
- to ask, to expect, to consider obligatory, to request and expect with specific supporting acts such as transmitting, communicating, calling for, demanding;
- to want, to need, to require, to have need of with supporting acts of wanting, needing, requiring;
- to necessitate, to ask, to postulate, to need, to take, to involve, to call for, to demand to require as useful, to just, or to proper.

The ways of functioning, understanding, elicitation, modelling, reasoning, assessment, and construction can be characterised in a similar form.

The rigor stage may be replaced by other stage that suupport different purposes. We concentrated on prescription and construction of new systems. Another application is *model refinement* similar to two-model representation of the Königsberg bridge problem on page 9.

Design science aims at another kind of model refinement by adding more rigor after evaluation of a model. This refinement is essentially *model evolution*. Another refinement is the enhancement of models by concepts. This refinement is essentially a 'semantification' or *conceptualisation* of the model. Experimentation and justification of models is a third kind of adding rigor to (conceptual) models.

17.2.6 Properties of the Models-Origin and the Models-Reflections Analogies

Figure 17.1 bases modelling on a quadruple of origin, model, author and addressee. The origin-model association as well as the the experimentation, construction or reasoning with models is based on an explicit consideration of the notion of an *analogy* between the model and the origin or the model and its reflection in theories,

constructions, hypotheses, or illustrations. . Therefore we need a characterisation of analogies.

Analogies are statements on similarity, statements of adjustment, statements of emphases. They characterise the approximation made by the model. These characterisations can be given by:

Degree of *structural analogy:* This grad characterises the degree of similarity of either the original with the model or of the model with its reflection.

Degree of *qualitative analogy:* This grad characterises to which degree the character and constitution is reflected.

Degree of *structural adjustment:* This degree characterises to which extent the structure is considered despite from the later use.

Degree of *qualitative adjustment:* This degree characterises what is going to be used for the later exploitation and what part is not going to be used.

Degree of *functional adjustment:* This degree characterises the functions that are considered and the functions that are not considered.

Degree of *contrast and emphasis:* This degree provides a means to specifically consider the distortion, amplification and idealisation made by the model.

Degree measurement is based on the ratio between the good or bad cases against all possible cases. We may consider a number of ratio measurements. *Recall evaluation* relates the number of positive observations to the number of all possible observations. *Fallout evaluation* measures the negative observations against the number of all possible observations. *Precision evaluation* typically measures the relevant observations similar to recall observations. Measurement functions often use *metrics*. Another kind of measurement uses *model-checking* functions that are based on predicates that evaluate certain properties. These properties can be used to decide whether a work product is consistent and can be refined for work products at the implementation layer.

Additionally, we need an approach to provide *tolerance* of the results and deviations from the either the origin or the realisations.

Additionally we need a logics that provides us with a means for reasoning on analogy and for using analogy for transfer of derived statements and properties into the other domain. This directly results in a *logics of analogical reasoning*. . Such logics have been developed in artificial intelligence and logics research. We may use, for instance, derivation rules for a source object s and a target object t of the following form

$$\frac{t \approx_{\alpha} s\ ,\ \alpha \Vdash \beta}{\beta(t) := \beta(s)}$$

This rules allows us to conclude that whenever the source and the target object are analogue based on a certain predicate α and the predicate α entails another predicate β then we may transfer the value for β for the source to the target.

Another such rule is the following one:

$$\frac{t \approx_{\alpha} s\ ,\ \alpha(s)}{\diamond\alpha(t)}$$

If we know that s and t are α-analog and we observe the value $\alpha(s)$ then it is plausible to assume $\alpha(t)$.

We also may incorporate *lifting relations* or *bridge rules* between an origin and the model or the model and its reflections. These rules must consider a certain context between for both the model and the origin or both the model and its reflection. Therefore we use mappings between two languages with an additional context parameter for a context $\mathscr{C}$:

$F : \mathscr{L}_1 \times \mathscr{C} \to \mathscr{L}_2$.

If we consider formulas α in context C_i then rules need to extended:

$$\frac{(\alpha_1, i_1) \dots (\alpha_n, i_n)}{(\alpha, i)} \varphi$$

Such rules state that $(\alpha_1 \dots \alpha_n)$ in their contexts $(C_{i_1} \dots C_{i_n}$ imply α in the context C_i is the applicability condition i φ is valid.

Such rules are considered in calculi of plausible reasoning that incorporate abduction and induction. Plausible reasoning uses inference pattern which can yield to uncertain conclusions even if the premises are certain. It is typical for situations in which the knowledge is incomplete. The modelling situation is based on incomplete information or incomplete knowledge.

The most important property for the analogy relationship is *adequacy*. Adequacy requires the satisfaction of the following four properties.

Similarity between origin and model or between model and reflection in dependence on the purpose of the model is based on an explicitly given similarity relation that allows also to reason on the restrictions of similarity, i.e. in the case of origin and model we may base similarity in subsets of properties $\Phi(O)$ and $\Phi(M)$ that are defining the similarity. Similarity support the deployment of the model instead of the origin or the reflection instead of the model in all situations in which there is a similarity between the two sides.

Regulative factors form a standardisation on the basis of exact rules which are given within a well-defined system. These rules allow to derive the properties and do not result in exceptions that cover specific properties of the target that are not observed for the source.

Copiousness is based on the capacity of the model. The model is a far better medium for reasoning about the origin or the reflection. It allows to simpler draw conclusions, to simpler reason about properties and to simpler state postulates.

Simplicity of the model is based on its concentration on the essential and relevant properties in dependence on the models purpose.

17.3 Conclusion

The aim of the chapter has not been to develop an entire theory of conceptual modelling. Instead we aimed in the development of a programme for the theory. We

described the general purpose of this theory, demonstrated how different paradigms can be selected, and showed which scope, modelling acts, modelling methods, modelling goals and modelling properties might be chosen for this theory.

The programme requires far more work. The theory needs a variable taxonomy that allows a specialisation to languages chosen for a given application domain, must be based on a mathematical framework that allows to prove properties, must be flexible for coping with various modelling methodologies, must provide an understanding of the engineering of modelling, and finally should be supported by a meta-CASE tool that combines existing CASE to to a supporting workbench.

The findings of this chapter are:

- A model is a representation of something for a purpose of somebody developed by somebody else.
- Each model is author-driven and addressee-oriented, is aspect-related, is purpose-specific, is limited in space, context and time, and is perspective.
- The model quality is also given by those elements that are not observed for the origin or not realised in the reflection or realisation.
- Due to amplification, distortion and idealisation, models cannot be used outside their purpose. If the purpose changes then the model should change as well.
- Models are similar to concepts; they are abstract and concrete; they associate worlds, e.g., the world of origins and models.
- Conceptual models are similar to other systems context- and utilisation-dependent. They have their value within the purpose range.

Models are imperfect and diverge from the real world. They are incomplete, have a different behaviour, and also use other kinds of errors. Imperfectness is based on exceptional states (events, time lags), on incompleteness to limitations of the language and consideration, and errors either based on real errors and exceptional states or based on biases.

A theory of conceptual modelling can be based on a system of guiding principles. This paper shows that at least three guiding principles must be explored in detail:

Internal principles are based on a set of ground entities and ground processes.

Bridge principles explain the results of conceptual modelling in the context of their usage, for instance for explanation, verification/validation, and prognosis.

Engineering principles provide a framework for mastering the modelling process, for reasoning on the quality of a model, and for termination of a modelling process within a certain level of disturbance tolerance (error, incompleteness, open issues to be settled later, evolution).

References

1. C. Beeri and B. Thalheim. Identification as a primitive of database models. In *Proc. FoMLaDO'98*, pages 19–36. Kluwer, London, 1999.
2. E. Börger and B. Thalheim. A method for verifiable and validatable business process modeling. In *Software Engineering*, volume 5316 of *Lecture Notes in Computer Science*, pages 59 – 115. Springer, 2008.

3. P. P. Chen, J. Akoka, H. Kangassalo, and B. Thalheim, editors. *Conceptual modeling: current issues and future directions*. LNCS 1565. Springer, 1998.
4. W. Deppert. Theorie der Wissenschaft. Christian-Albrechts-University at Kiel, Lecture notes for academic year 2008/09, http://wolfgang.deppert.de, 2009.
5. G. Fiedler and B. Thalheim. Towards semantic wikis: Modelling intensions, topics, and origin in content management systems. *Information Modelling and Knowledge Bases*, XX:1–21, 2009.
6. S. Gregor. Building theory in the sciences of the artificial. In *DESRIST*. ACM, 2009.
7. A. Hevner, S. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quaterly*, 28(1):75–105, 2004.
8. M. Jackson. *Problem frames*. Pearson, Harlow, 2006.
9. R. Kaschek. *Konzeptionelle Modellierung*. PhD thesis, University Klagenfurt, 2003. Habilitationsschrift.
10. B. Mahr. Information science and the logic of models. *Softw. Syst. Model.*, 8:365–383, 2009.
11. T. Mäkinen. *Towards Assessment Driven Software Process Modeling*. PhD thesis, TUT Pori, 2010.
12. J. Mittelstraß, editor. *Enzyklopädie Philosophie und Wissenschaftstheorie*. J.B. Metzler, Stuttgart, 2004.
13. G. L. Murphy. *The big book of concepts*. MIT Press, 2001.
14. A. Olivé. *Conceptual modeling of information systems*. Springer, Berlin, 2007.
15. G. Simsion. *Data modeling - Theory and practice*. Technics Publications, LLC, New Jersey, 2007.
16. H. Stachowiak. Modell. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon Zur Wissenschaftstheorie*, pages 219–222. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.
17. W. Steinmüller, *Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993.
18. B. Thalheim. *Entity-relationship modeling – Foundations of database technology*. Springer, Berlin, 2000.
19. B. Thalheim. The conceptual framework to user-oriented content management. *Information Modelling and Knowledge Bases*, XVII:30–49, 2007.
20. B. Thalheim. Towards a theory of conceptual modelling. In *ER Workshops*, volume 5833 of *Lecture Notes in Computer Science*, pages 45–54. Springer, 2009.
21. B. Thalheim. The conceptual framework to multi-layered database modelling. In *Proc. EJC*, pages 118–138, Maribor, Slovenia, 2009.
22. B.L. Whorf. *Lost generation theories of mind, language, and religion*. Popular Culture Association, University Microfilms International, Ann Arbor, Mich., 1980.

The Science and Art of Conceptual Modelling

Bernhard Thalheim

Christian-Albrechts-University Kiel, Computer Science Institute, 24098 Kiel, Germany,
thalheim@is.informatik.uni-kiel.de,
http://www.is.informatik.uni-kiel.de/∼ thalheim

Abstract. Conceptual modelling is one of the central activities in Computer Science. Conceptual models are mainly used as intermediate artifact for system construction. They are schematic descriptions of a system, a theory, or a phenomenon of an origin thus forming a model. A conceptual model is a model enhanced by concepts. The process of conceptual modelling is ruled by the purpose of modelling and the models. It is based on a number of modelling acts, on a number of correctness conditions, on modelling principles and postulates, and on paradigms of the background or substance theories. Purposes determine the (surplus) value of a model. Conceptual modelling is performed by a modeller that directs the process based on his/her experience, education, understanding, intention and attitude. Conceptual models are products that are used by other stakeholders such as programmers, learners, business users, and evaluators. Conceptual models use a language as a carrier for the modelling artifact and are restricted by the expressiveness of this carrier.
This paper aims at a discussion of a general theory of *modelling as a culture and an art*. A general theory of modelling also considers modelling *as an apprenticeship* and *as a technology*. It is thus an *art*. Modelling is on of the main elements of Computer Science *culture* that consists of commonly accepted behaviour patterns, arts, consensus, institutions, and all other supporting means and thoughts.

Keywords
conceptual modelling, modelling workflow, foundations of modelling.

1 The Triptychon of Model as an Artifact, Modelling as an Activity and Modelling as an Art and Science, thus as a Culture

Conceptual modelling is a widely applied practice in Computer Science and has led to a large body of knowledge on constructs that might be used for modelling and on methods that might be useful for modelling. It is commonly accepted that database application development is based on conceptual modelling. It is however surprising that only very few publications have been published on a *theory of conceptual modelling*. We continue the approach [36, 37] and aim in a theory of modelling within this paper. An approach to a theory of models has been developed in [37]. An approach to a theory of model activities is discussed in [36].

1.1 Three Guiding Concerns during Conceptual Modelling

Conceptual modelling is often only discussed on the basis of modelling constructs and illustrated by some small examples. It has however three guiding concerns:

1. *Modelling language constructs* are applied during conceptual modelling. Their syntactics, semantics and pragmatics must be well understood.
2. *Application domain gathering* allows to understand the problems to be solved, the opportunities of solutions for a system, and the requirements and architecture that might be prescribed for the solution that has been chosen.
3. *Engineering* is oriented towards encapsulation of experiences with design problems pared down to a manageable scale.

The first concern is handled and well understood in the literature. Except few publications, e.g. [2], the second concern has not yet got a sophisticated and well understood support. The third concern has received much attention by data modelers [29] but did not get through to research literature. It must therefore be our goal to combine the three concerns into a holistic framework.

A model is nothing else than a material or virtual artifact that is used or can be used in dependence on some objectives, purposes and functions by somebody. As an artifact, somebody else was acting as a developer with some intentions and goals. The objective of models, the purpose of models and the function of models are often considered to be synonymous. There are however differences that should be taken into account [13, 18]. The objective of a model is the change of reality through the model that can be reached by activities and is a goal of stakeholders, i.e., it is a ternary relation between two states of reality and humans. The purpose is based on the objective and presupposes the existence of instruments through which the state change can be reached. The purpose is bound to intentions of stakeholders to reach this objective by activities. The function of a model is based on the use or deployment of a model in a practice. It is thus bound to processes in which the model has its applications ('Sprachspiel' (language game) [39] or deployment game).

Objectives can be abstract. Purposes need however a meaningful description. The purpose thus includes the intention, the meaning, the function, and the tasks. Typical functions of a model are deployment for illustration or explanation, for verification or validation, as a deputy for another artifact or surrogate within an investigation or experimentation process, for tests of theories, as basis for simulation, within a learning process, and last but not least for construction of systems augmenting current reality. The model plays a part within these processes. These parts are typically categorised by roles.

1.2 Implications for a Theory of Conceptual Modelling

The three concerns of conceptual modelling must be integrated into a framework that supports the relevant concern depending on the modelling work progress. The currently most difficult concern is the engineering concern. Engineering is inherently concerned with failures of construction, with incompleteness both in specification and in coverage of the application domain, with compromises for all quality dimensions, and with problems of technologies currently at hand.

At the same time, there is no universal approach and no universal language that cover all *aspects of an application*, that have a well-founded *semantics* for all constructions, that reflect any *relevant facet in applications*, and that *support engineering*. The choice of modelling languages is often a matter of preferences and case, empirical usage, evolution history, and supporting technology.

1.3 Differences between 'Model', 'To Model' and 'Modelling'

The conceptions of model, of the activity 'to model' and of modelling are often used as synonyms. We must however distinguish these conceptions for a theory of models, a theory of model activities and a theory of the modelling process.

Based on the notions in the Encyclopedia Britannica [23] we distinguish between the conception of a model, the conception of a model activity, and the conception of modelling processes.

The model as an artifact: The model is something set or held for guidance or imitation of an origin and is a product at the same time. Models are enduring, justified and adequate artifacts from one side. From the other side, models represent the state of comprehension or knowledge of a user.

To model as an activity: 'To model' is a scientific or engineering activity beside theoretical or experimental investigation. The activity is an additive process. Corrections are possible during this activity. Modelled work may be used for construction of systems, for exploration of a system, for definition and negotiation, for communication, for understanding and for problem solving.

Modelling as a systematically performed technological process: Modelling is a technique of systematically using knowledge from computer science and engineering to introduce technological innovations into the planning and development stages of a system. At each stage the modeller is likely to ask both why and how, rather than merely how. Modelling is thus based on paradigms and principles.

Additionally, the notion of model may be used in an adjective sense as serving as or capable of serving as a pattern or being a usually miniature representation of something. This notion is often used for sample representations such as a 'model chair'. Another notion of the model that is not of interest within this paper is the miniature representation of something.

1.4 The Simultaneity of Art, Culture, Technology and Techniques in Modelling

Modelling can be understood as a technique[1] or as a technology[2]. [23] distinguishes between science and technology: Technology is the systematic study of techniques for making and doing things; science is the systematic attempt to understand and interpret

[1] I.e., the fashion, manner, mode, modus, system, way, wise in which a system etc. is mastered. Techniques consist of methods of accomplishing a desired aim.

[2] Technology is an element of engineering. It consists of the practical application of knowledge especially in a particular area. It provides a capability given by the practical application of knowledge. Therefore, it is a manner of accomplishing a task especially using technical processes, methods, or knowledge.

the world. While technology is concerned with the fabrication and use of artifacts, science is devoted to the more conceptual enterprise of understanding the environment, and it depends upon the comparatively sophisticated skills of literacy and numeracy.

At the same time, modelling is an art [3]. Modelling is a highly creative process. It requires skills in planning, making, or executing. It is often claimed that it is not to be formalisable. It requires deep insight into the background as well as skills, careful simplification, experience and ingenuity. Due to the variety of viewpoints, modelling is also based on judgement and clever selection with different alternatives.

Modelling in one of the main activities in Computer Science. It consists of commonly accepted and practised behavior patterns, arts, consensus, institutions, and all other products of human work and thought. Turning to [23][4], culture is based on the capacity for rational or abstract thought. The meaning of abstraction is not sufficiently explicit or precise. The term symboling has been proposed as a more suitable name for assigning to things and events certain meanings that cannot be grasped with the senses alone.

This culture is learned and shared within communities which have their own behaviour pattern and approaches. It is not yet a science since it heuristically uses operational and/or scientific terms.

1.5 Orientation of this Paper

This paper explores modelling as an art and culture. We base the discussion on a theory of models and of modelling activities. We abstract therefore in this paper from micro-, meso-, macro-models or model suites used in many natural sciences or model suites [35], e.g., model ensembles used in UML or OWL. We do not yet consider modelling competency or MDA/D. We do not yet consider modelling competency. All notions used in this paper are based on [33]. The main goal of this paper is to show that modelling requires apprenticeship and technology. The orientation towards an expert mode can be reached if modelling is based on systematic development and if modelling is considered to be a craft of modelling activities. This approach shows that modelling incorporates design science in a wider sense as it has been considered in the literature.

[3] Art requires capability, competence, handiness, and proficiency. Art is based on finesse, i.e. on refinement or delicacy of workmanship. Models and art share a Janus head evaluation: The judgement of beauty evaluates the model within a community of business users. The judgement of the sublime evaluates the model against its technical realisation. A model has thus both an extrinsic and intrinsic value.

[4] The notion of culture combines at least eight facets: (1) cultivation, tillage; (2) the act of developing the intellectual and moral faculties especially by education; (3) expert care and training; (4) enlightenment and excellence of taste acquired by intellectual and aesthetic training; (5) acquaintance with and taste as distinguished from vocational and technical skills; (6) integrated pattern of human knowledge, belief, and behavior that depends upon man's capacity for learning and transmitting knowledge to succeeding generations; (7) the customary beliefs, social forms, and material traits of a group; and (8) the set of shared attitudes, values, goals, and practices that characterizes a company or corporation.
Culture requires different practices: education, enlightenment, erudition, learning from one side, gentility, manners, discrimination, taste from the other side, and sophistication, class, and elegance from a third side.

We base our ideas on our observations on model developments for very large database schemata and very large database systems[5]. Such systems require a well organised modelling process. They must be evolution-prone and revision-prone. The paper concentrates thus one of the main workflows: *description of application worlds* followed by *prescription for system worlds* and *specification of systems*.

2 The World of Models

2.1 The Conception of the Model

Models are *artifacts* selected by a *stakeholder* based on some stakeholder *judgement* or *perception* and governed by the *purpose*. Models can thus be characterised by *main dimensions*:

purpose ("wherefore") of models and modelling with the intentions, goals, aims, and tasks that are going to be solved by the model,

result of mapping ("whereof") with a description of the solution provided by the model, the characterisation of the problem, phenomena, construction or application domain through the model,

language ("wherewith") with a careful selection of the the carrier or cargo[15] that allows to express the solution, the specification of the world or the construction, and

value ("worthiness") of a model by explicit statement of the internal and external qualities, and the quality of use, e.g. explicit statement of invariance properties relating the model to its associated worlds or by preservation properties that are satisfied by the model in dependence on the associated worlds.

These four dimensions are driven by two *context dimensions*: the application domain dimension rules the scope and (explicit and implicit) disregard of the model; the user or stakeholder dimension governs the viewpoint, orientation and background of users involved. The mapping associates the origin and the artifact. As far as we are interested in modelling of information systems, we may use a (semi-)formal language for the artifact.

[5] Due to our involvement into the development and the service for the CASE workbenchs $(DB)^2$ and ID^2 we have collected a large number of real life applications. Some of them have been really large or very large, i.e., consisting of more than 1.000 attribute, entity and relationship types. The largest schema in our database schema library contains of more than 19.000 entity and relationship types and more than 60.000 attribute types that needs to be considered as different. Another large database schema is the SAP R/3 schema. It has been analyzed in 1999 by a SAP group headed by the author during his sabbatical at SAP. At that time, the R/3 database used more than 16.500 relation types, more than 35.000 views and more than 150.000 functions. The number of attributes has been estimated by 40.000. Meanwhile, more than 21.000 relation types are used. The schema has a large number of redundant types which redundancy is only partially maintained. The SAP R/3 is a very typical example of a poorly documented system. Most of the design decisions are now forgotten. The high type redundancy is mainly caused by the incomplete knowledge on the schema that has been developed in different departments of SAP.

These main dimensions of models and modelling govern the model and the modelling acts. They are extended by secondary dimensions that are used to shape and to adapt the model: the artifact, the user, the context and the application domain dimensions. The mapping dimension is discussed in [36]. The value dimension can be described based on [10]. The purpose dimension is ruling and *governing* both the development of models and the application of models. This tight governance is caused by the main aim of a model: to provide a solution to a problem.

2.2 The Model as a Physical or Virtual Artifact

The main product of modelling and model activities is the model, i.e. an artifact that is considered to be worth for its purpose by the author. The model can, for instance, be used for the description of the world of origins or for the prescription of constructions. There are a number of explicit choices an author makes and that rule applications of models. Modelling of information systems

depends on the *abstraction layer*, e.g. requirements, specification, realisation or implementation layer,
depends on chosen *granularity and precision* of the work product itself,
depends on *resources* used for development of a model such as the language,
depends on *level of separation of concern* such as static/dynamic properties, local/global scope, facets,
depends on *quality properties of the input,* e.g. requirements, completeness, conciseness, coherence, understandability,
depends on decomposition of the work products in ensembles of sub-products, and
satisfies quality characteristics such as quality in use, internal quality, and external quality.

The task of model development is never completed (ta panta rhei ($\tau\alpha\ \pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$), 'the rivers flow'; narrative: everything flows). Models are changing artifacts due to changes imposed by

scope insight for conscious handling of restriction, capabilities, opportunities,
guiding rules for convenience, for completion, refinement, and extension,
development plans for partial delivery of models, partial usage and deployment,
theories supporting development of models,
quality characteristics for model completion, model evolution, model engineering, and
mapping styles for mapping models among abstraction layers.

2.3 The Purpose Dimension

The purpose dimension is *ruling* and *governing* the model, the development process and the application process because of the main reason for using a model is to provide a solution to a problem. Therefore the purpose is characterised by the solution to the problem provided by the model. We may distinguish a number of concerns such as

the impact of the model ("whereto") for a solution to a problem,
the insight into the origin's properties ("how") by giving details how the world is structured or should be structured and how the functionality can be described,

restrictions on applicability and validity ("when") of a model for some specific solutions, for the validity interval, and the lifespan of a model,

providing reasons for model value ("why") such as correctness, generality, usefulness, comprehensibility, and novelty, and

the description of functioning of a model ("for which reason") based on the model capacity.

The task of model development is never completed (ta panta rhei ($\tau\alpha\ \pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$), 'the rivers flow', narrative: everything flows). Models are *evolving artifacts* due to changes imposed by

- scope insight for conscious handling of restriction, capabilities, opportunities,
- guiding rules for convenience, for completion, refinement, and extension,
- development plans for partial delivery of models, partial usage and deployment,
- theories supporting development of models,
- quality characteristics for model completion, evolution and engineering, and
- mappings styles for mapping models among abstraction layers.

2.4 The Language Dimension

Models are represented by *artifacts* that satisfy the pragmatic purposes of users. In this case, artifacts are linguistic expressions that describe the model. Linguistic expressions are built within a language with some understanding. Therefore, artifacts use syntax, semantics and pragmatics built within the chosen language.

Models are often expressed through expressions in a formal language $\mathcal{L}_{\tilde{M}}$. A model should support its objectives. Optimally, these objectives $\Psi(M)$ can be expressed in the same language $\mathcal{L}_{\tilde{M}}$ that is also used for the model M. A model has a number of properties. Some of them are of interest and used for characterisation of the model, e.g., $\Phi(M)$. This characterisation depends on the model and its purpose.

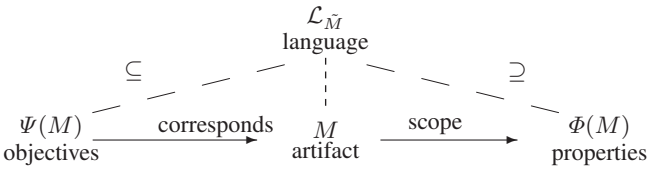


Fig. 1. Artifacts with a language, their properties and objectives within a given language for the artifact

Constructive languages are a special case and support

- the prescription of the objectives or postulates that restrict the judgement that an artifact can be accepted as a model,
- the scope of our attention to those artifacts that can be considered for a model or for parts of a model, and
- the orientation of the user on certain properties that are of interest for the purpose of modelling.

Natural languages have a high potential for deployment of deep semantics and cause a threat to everybody who does not use the language within the same semantical culture. Culture depends on participating stakeholders, their profile (educational, employment, psychological) and includes language, styles of communication, practices, customs, and views on roles and relationships. Deployment of natural language expressions may thus result in misunderstandings. There are two ways to avoid such: the development of a sophisticated ontology that includes all namespaces a user might use or the development of an *orthonormalised language* [20] that is restricted in expressivity and does not allow misinterpretations.

2.5 The Context Dimensions

The User Dimension. A number of users are involved into the development of models. The user dimension thus reflects intentions, the understanding, the comprehension and other characteristics of users in a variety of roles, e.g.,

the role of an *author ("by whom")* that results in reflections of the educational level, application of templates, pattern or reference models,
the role of an *addressee ("to whom")* that restricts the utilisation of the model or that supports the extended application beyond the purpose originally intended, and
the role of *broad public ("whichever")* that develops a common understanding of the model depending on the group or the culture of the public.

The Application Domain Dimension and the World of Origins. The application domain consists of people, organisational systems, and technical systems that interact to work towards a goal. This dimension clarifies

the *domain depending on models purpose ("for what")* such as an application domain, properties reflected or neglected,
the *scope to specific elements ("what")* that are considered to be typical and whose properties should be reflected,
the *attention within the domain depending on models purpose ("where")* that limits the model to the 'normal' aspects,
the *orientation of the domain ("wherefrom")* that restricts the attention and the issues for the current activities supported by the model,
the sources for origins or the infrastructure ("whence") considered for the model, and
the *restrictions of the world ("wherein")* associated with the model.

3 The World of Modelling Activities

3.1 Workflows Applied in the Model Development and Deployment Process

The purpose dimension governs the workflows applied in conceptual modelling. It also governs the kind of model application. We may distinguish a number of workflows in conceptual modelling such as the following ones:

Construction workflows are based on creation of models (as images, representations or portraits of the origin) that are used for production of systems (using as models as groundwork, background, pattern, standards, prototypes for the system). This kind of model exploitation uses the *dichotomy of models* as image of an origin and groundwork for a system.

Explanation workflows result in new insights into the world of the origins.

Optimisation-variation workflows result in an improvement and adaptation of the origins.

Verification-validation-testing workflows result in an improvement of the one of the subject, in most cases in an improvement of models.

Reflection-optimisation workflows are typical for mathematical modelling of the world of origins.

Explorative workflows are using models for learning about origins.

Hypothetical workflows are typical for discovery sciences, e.g., sciences used for climate research.

Documentation-visualisation workflows target on better understanding and comprehension of models.

These workflows can be intertwined or shuffled with each other. They may be performed one after another. In this paper we concentrate on the *construction* (or *creation-production*) workflow which seems to be central for information systems.

3.2 Conceptual Modelling Activities Governed by its Purpose

Models are developed with different goals, different scope, within different context, with different appeal to the receiver of the model, with different granularity, with different background, and with different realisation forms. Therefore we have to explicitly handle modelling purpose properties.

The *mission* of modelling is described by scope of the model, the users community, the tasks the model might support, the major and minor purposes satisfied by the model and the benefits obtained from the model for the given user community. The *goals* of a model are based on the impact of the model, restricted by the relationships among users and their roles they are playing. The *brand* of the model is given by the who-what-whom-action pattern. The *meta-model* can be used to provide information about the model such as the context of the model, the context in which the model might be useful to the auditory, the usage of the model, and the restrictions of the model.

3.3 Modelling Acts

It surprises that these model activities are not explicitly handled in most modelling approaches. The same observation can be observed for a declaration of the main goals of the modelling act. Main modelling acts which are the following ones:

construct a model, a part of the model, a concept or a judgement, etc. (describe, delineate, fabricate, master),

communicate the judgements, the observations, the concepts, etc. (explain, express, verbalise or display),

understand the application domain, the system opportunities, etc. (cognise, identify, recognise, percept),
discover the problems, the potential, the solutions, etc. (interact, identify),
indicate properties of importance, relevance, significance, etc. (visualise, measure, suggest, inform),
variate and *optimise* a solution, a judgement, a concept, a representation depending on some criteria,
verify or *validate* or *test* a model, a solution, a judgement, a representation or parts of those,
control the scope of modelling, the styles or pattern, parts of a model, judgements, etc. (rule, govern, proofread, confirm, restrain, administer, arrange, stratify, standardise),
alternate or *compensate* or *replace* or *substitute* or *surrogate* models or parts of them, judgements, concepts, etc. (transfer, reassign, evolve, migrate, balance, correct, novate, truncate, ersatz).

The first and last four goals lead to a *datalogical* model that is structured according to technology. The other goals result in an *infological* model that is delivered to the needs of the user. We thus use a different frame of reference. The application of the results may thus be descriptive or prescriptive, constitutive or prognosticating, categorical or exegetic or contemplative or formulaic.

4 The Theory of Conceptual Models and Conceptual Modelling

4.1 Conceptual Modelling: Modelling Enhanced by Concepts

An information systems model is typically a schematic description of a system, theory, or phenomenon of an origin that accounts for known or inferred properties of the origin and may be used for further study of characteristics of the origin. *Conceptual modelling* aims to create an abstract representation of the situation under investigation, or more precisely, the way users think about it. Conceptual models enhance models with concepts that are commonly shared within a community or at least between the stakeholders involved in the modelling process. A general definition of concepts is given in [37]. Concepts specify our knowledge what things are there and what properties things have. Concepts are used in everyday life as a communication vehicle and as a reasoning chunk. Concept definition can be given in a narrative informal form, in a formal way, by reference to some other definitions etc. We may use a large variety of semantics [28], e.g., lexical or ontological, logical, or reflective. [37] introduces a general theory of concepts that can be used for conceptualisation.

Conceptualisation aims at collection of objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them. It is thus an abstract, simplified view or description of the world that we wish to represent. Conceptualization extends the model by a number of concepts that are the basis for an understanding of the model and for the explanation of the model to the user.

4.2 Conceptual Models

The theory of conceptual models [37] extends the framework [11, 31, 32]. Models can be characterised by four main dimensions: (1) Models and conceptual models are governed by the purpose. The model preserves the purpose. (2) The model is a mapping of an origin. It reflects some of the properties observed or envisioned for the origin. (3) Models use languages and are thus restricted by the expressive power of these languages. (4) Models provide a value or benefit based on their utility, capability and quality characteristics.

The *purpose of a model* covers a variety of different intentions and aims such as *perception support* for understanding the application domain, *explanation and demonstration* for understanding an origin, *preparation* to management and handling of the origin, *optimisation* of the origin, *hypothesis verification* through the model, *construction* of an artifact or of a program, *control* of parts of the application, *simulation* of behaviour in certain situations, and *substitution* for a part of the application. Depending of the purpose we shall use different models.

Models are author-driven and addressee-oriented. They depend therefore on the culture, attitude, perceptions, education, viewpoints etc. of the stakeholders involved into the modelling process. Models are purposeful/situated/easily-modifiable/sharable/reusable/multi-disciplinary/multi-media chunks of knowledge about the application domain. They are both bigger and smaller than theories, i.e., bigger since they integrate ideas from different theories, since they use different representations, and since they are directed by their purpose; smaller since they are created for their purpose in a specific situation and since they are developed to be sharable and reusable. One of the most important quality characteristics of a model is that it should be easy to modify and to adapt.

4.3 Towards a Theory of Model Activities

Modelling activities are based on modelling acts. Modelling is a specific form and we may thus develop workflows of modelling activities. These workflows are based on work steps [33] such as ‘decompose’ or ‘extend’, abstraction and refinement acts, validation and verification, equivalences of concepts, transformation techniques, pragmatistic solutions and last but not least the domain-specific solutions and languages given by the application and implementation domains.

The act of modelling is based on an *activity* that is characterised by the *work products*, the *aspects* under consideration (scope), the *resources* used in an activity, and the *partners* involved into the activity. Additionally we might extend this characterisation by activity goals and intentions (for what), time span (when), and restrictions (normal, exception and forbidden cases) or obligations for later activities. We may distinguish a number of activities and acts, e.g., *understand, conceptualise, abstract, define, construct, refine, document* and *evaluate*. Model activities should be governed by *good practices* which can be partially derived from modelling as an apprenticeship or technology. A theory of modelling activities has been developed in [36] and therefore not within the scope of this paper.

5 Modelling of Information Systems as Engineering

5.1 Models Serving Both as a Description of an Application (Domain and Problem) and as a Prescription for Construction (of Systems)

By taking a leaf out of D. Bjorner [1] book we divide information systems engineering into five main phases: (1) *application domain description* with properties that are of interest and that are of relevance, (2) requirements or *objectives prescription for a model*, (3) *model development* with a statement of properties that are obeyed by the model, (4) requirements of *objectives prescription for the construction* of an information system, and (5) *information systems construction* and coding with properties that are obeyed by the information system. Therefore, a model is used as a mediator between the application world and the systems world. The (application, model, system)-triple is reflected by the *information system development triptych* consisting of description of application world, prescription for construction and specification of systems.

Conceptualisation is an orthogonal phase that aims at a theoretical underpinning of models. It is used for *semantification* of models and for improvement of comprehensibility of models and explicit reasoning on elements used in models.

The application domain description is mapped to a model describing the application domain, its entities, functions, events, and behaviour. It is based on a formal, semi-formal or natural language which allows to formulate a set of theorems or postulates or properties that are claimed to behold of the domain model. The information system itself is an artifact too. The model mediates between this final artifact and the application. Models describe the problem to be solved for the application and which are used as starting point for implementation. They are also used for documentation of the system, for migration and evolution processes, for optimisation of systems, for control of parts of systems, and for simulation of systems. Models must reflect the structure of a system, the functionality of a system, the support facilities of a system and the collaboration environment of a system.

Therefore we concentrate on one of the workflows: the prescription of systems imposed by the description of an application domain and of the problems under solution. This workflow is often considered to be one of the main workflows. We may also use other workflows. The construction workflow is however a typical example of an engineering workflow[6]. Engineering is nowadays performed in a systematic and well-understood form.

5.2 The Construction Workflow Based on Information Systems Models

Modelling is based on an evolutionary process and thus consists of at least three sub-processes:

[6] The difference between scientific exploration and engineering is characterised by [24] as follows: “Scientists look at things that are and ask ‘why’; engineers dream of things that never were and ask ‘why not’. Engineers use materials, whose properties they do not properly understand, to form them into shapes, whose geometries they cannot properly analyse, to resist forces they cannot properly assess, in such a way that the public at large has no reason to suspect the extent of their ignorance.” Modelling incorporates both engineering and science. It is thus considered to be an engineering science.

- *selection* including rigorous testing against the origin,
- *communication* for generation of a common understanding and a productive way of thinking within a community, and
- *accumulation* of results and integration of these results into future developments.

The construction workflow is one of the most prominent workflows in information systems modelling. Methodologies developed for software engineering can be directly applied to this workflow. They are however mainly oriented towards system construction. The systems description dimension is not as well explored. The combination of these two sub-workflows is shown in Figure 2. We need to include into this combination also the quality dimension. The body of knowledge of software engineering includes also a large set of quality characteristics. [10] develops an approach to systematic quality development. We integrate this systematic quality management.

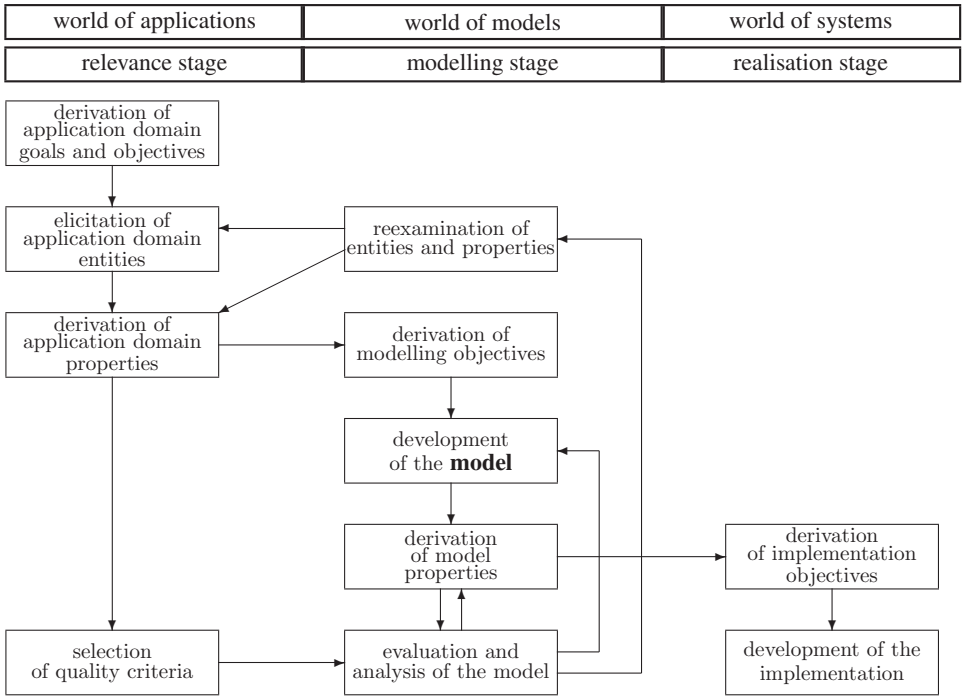


Fig. 2. The construction workflow that includes quality assurance

We can also develop other workflows such as agile modelling, spiral modelling and incremental modelling workflows. We restrict our attention in the sequel to the workflow in Figure 2 due to the length of this paper. This workflow separates three different worlds: the world of applications, e.g., the application domain in dependence on the purpose; the world of models, e.g. conceptual models used for information systems development; the world of systems, e.g., information systems combined with presentation

systems. Based on this separation we can distinguish three stages: the relevance, modelling and realisation stages. This workflow reflects the separation into objectives, the artifact and the properties within the language dimension.

5.3 Modelling by Generalising Engineering Approaches

The development of models results in many challenges. Modelling is essentially synthetic rather than analytic in substance. Identifying the real task of the modelling problem is probably the greatest challenge. Models can be based on building blocks. Another challenge is to find the right modelling method. Engineering targets at capacity of products to withstand service load both effectively and efficiently during their service life [24]. Efficiency also considers performance of the system. Engineering is also concerned with avoidance of technical, operational or unpredictable failures, i.e. to develop a system that deflect all service loads.

Engineering science for modelling is based on many different supporting sciences and technologies: industrial design, ergonomics, aesthetics, environments, life sciences, economics, mathematics, marketing, and manufactoring and forming processes. Engineering design includes five facets: design for effective function, design for manufacture, design for human users, socially responsible design, and economically responsible design.

Engineering distinguishes three dimensions: the stakeholder dimension, the procedure or process dimension, and the product dimension. It uses many techniques such as enformulation for structuring the purposes and objectives, problem decomposition together with component engineering, problem evolution, organising the engineering process, result evaluation, and result management. It also considers economic, social and environmental issues.

Therefore, it seems to be natural to use achievements of engineering for understanding modelling. This similarity is not only applicable to the description-prescription workflow but also for all the other workflows.

6 Modelling of Information Systems as a Science

6.1 Properties of Activities To Model

Activities to model form a process and can be characterised by a number of (ideal) properties:

Monotonicity: Activities are monotone if any change to be applied to one specification leads to a refinement. It thus reflects requirements in a better form.

Incrementality: Activities are iterative or incremental if any step applied to a specification is only based on new requirements or obligations and on the current specification.

Finiteness: Activities are finite if any quality criteria can be checked in finite time applying a finite number of checks.

Application domain consistency: Any specification developed corresponds to the requirements and the obligations of the application domain. The appropriateness can be validated in the application domain.

Conservativeness: Activities are conservative if any model revision that cannot be reflected already in the current specification is entirely based on changes in the requirements.

Typical matured activities to model are at least conservative and application domain consistent. Any finite sequence of activities can be transformed into a process that is application domain consistent. The inversion is not valid but depends on quality criteria we apply additionally. If the modelling process is application domain consistent then it can be transformed in an incremental one if we can extract such area of change in which consistency must be enforced.

6.2 Towards Modelling Principles

A theory of conceptual modelling can be based on a system of guiding principles. We conclude that at least three guiding principles must be explored in detail:

Internal principles are based on a set of ground entities and ground processes.

Bridge principles explain the results of conceptual modelling in the context of their usage, for instance for explanation, verification/validation, and prognosis.

Engineering principles provide a framework for mastering the modelling process, for reasoning on the quality of a model, and for termination of a modelling process within a certain level of disturbance tolerance (error, incompleteness, open issues to be settled later, evolution).

Information systems modelling principles are specialisations of design principles [3]. They are conventions for planning and building correct, fast or high performance, fault tolerant, and fit information systems. Conceptual modelling is based on architecture of a system into components, uses their interactions, and pictures their layout. Modelling is the process of producing models. It is thus adapted from engineering and may thus use the separation of activities into requirements, specification, development, and testing.

Depending on the purpose of the model several quality criteria may be preferred. For instance, construction models should fulfill criteria for good models such as correctness of models, refinement to highly effective systems, fault tolerance of systems, ubiquity of systems, and fitness of systems.

Modelling principles are not laws of nature, but rather conventions that have been developed by modellers to the most success when pursuing quality properties. Therefore, various sets of principles might be developed depending on the community. For instance, modelling based on extended ER models is based on compositionality, incrementality, structure-orientation, and conservativeness. Modelling principles for sets of models such as UML are far more difficult to develop and to maintain.

6.3 The Design Science Approach

MIS design science aims at the development of a general theory for models, model activities and modelling. We shall use the approach for a deeper insight into modelling. Models are called ‘design’ in [7].

The management information system community characterises the modelling process by seven guidelines [7]:

(1) model are purposeful IT artifacts created to address a problem;
(2) models are solutions to relevant and important problems;
(3) the utility, quality, and efficacy of models must be evaluated by quality assessment;
(4) modelling research must contribute to the state of the art;
(5) modelling research relies upon the application of rigorous methods;
(6) modelling is a search process and use termination conditions;
(7) models must be communicated both to technology-oriented as well as to management audiences.

We observe that guidelines (1), (2), and (7) are characterising the model. Guidelines (3), (6) characterise model activities. Guideline (3), (5) is related to modelling as a technology. Guideline (4) is a general statement that relates modelling to a science.

Main ingredients of modelling can be derived from these guidelines [4, 16]. Core components are purpose and scope (causa finalis), artifacts (causa materialis), the oneness of form and function (causa formalis), artifacts mutability, testable propositions about the model, and theoretical underpinning. Additional requests are the potential implementation (causa efficiens) and utility for exposition and testing [4].

Design science separates three cycles [38]: the relevance (or description) cycle, the design (or modelling) cycle, and the rigor (or conceptualisation) cycle.

6.4 Reasoning Support for Modelling

Properties and objectives are used as a glue between the three sections or panels of the triptych. We distinguish between properties of the application, properties of the model as an artifact, and properties of the information system as a final artifact. Since we typically use different languages the (property-objectives)-pair is used as an hinge in the triptych. Design science separates three cycles [38] within modelling workflows. It distinguishes the relevance cycle as the iterative process that re-inspects the application and the model, the design cycle as the iterative model development process, and the rigor cycle that aims in grounding and adding concepts developed to the knowledge base. This separation of concern into requirements engineering, model development and conceptualization is the starting point for the development of a reasoning support for modelling.

This reasoning support includes an explicit consideration of the quality of the model, of the quality of the modelling process, and of the quality of supporting theories. We may combine the informal discussions with our approach and separate the modelling acts by the things that are under consideration. Figure 3 displays the different ways of working during a database systems development. We may distinguish the *description-prescription-specification triptych* and the *engineering diptych*. The first consists of three phases:

Description of problems, phenomena and demands for system support in the application domain is based on the actual/goal analysis. It starts with the description of origins O and targets at an understanding of properties $\Phi(O)$ that are of relevance. It may be used for derivation of objectives $\Psi(M)$ for model development.

Prescription uses the objectives $\Psi(M)$ derived during description for model M development and derivation of properties $\Phi(M)$ that should reflect properties $\Phi(O)$ of the origin.

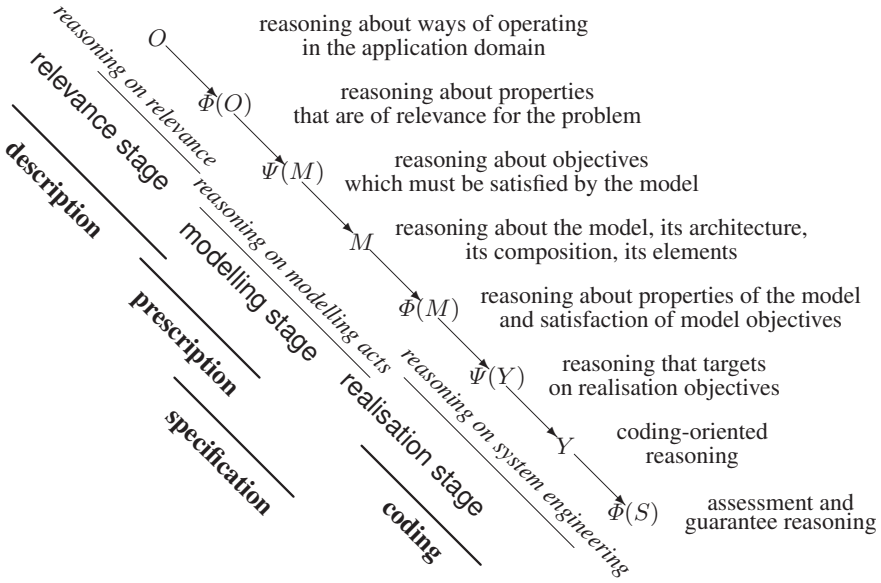


Fig. 3. Reasoning processes and reasoning support for description followed by prescription

Specification is based on a model M and its properties $\Phi(M)$. It may be used for derivation of objectives $\Psi(S)$ for system development.

The engineering diptych consists of

specification of the system that should solve problems in the application domain and *coding* of the systems that augment the reality.

The relevance stage consists of O and $\Phi(O)$. The modelling stage consists of $\Psi(M)$, M and $\Psi(M)$. The realisation stage consists of $\Psi(S)$ and Y. It could consider $\Phi(S)$.

We observe that support for modelling results in a wide variety of reasoning. For instance, reasoning about properties of a model is also based on an explicit consideration of the notion of an *analogy* between the model and the application domain things. or the model and its reflection in theories and constructions. Reasoning on objectives of realisations includes detection of requirements a system must satisfy.

A general theory of reasoning must therefore cover many different aspects. We may structure these aspects by a pattern for specification of reasoning support for modelling acts or steps as follows:

- the *modelling acts* with its specifics; [36]
- the *foundation for the modelling acts* with the theory that is going to support this act, the technics that can be used for the start, completion and for the support of the modelling act, and the reasoning techniques that can be applied for each step;
- the *partner* involved with their obligations, permissions, and restrictions, with their roles and rights, and with their play;
- the *aspects* that are under consideration for the current modelling acts;
- the *consumed and produced elements of the artifact* that are under consideration during work;

- the *resources* that must be obtained, that can be used or that are going to be modified during a modelling act.

Consider, for instance, the reasoning that targets on realisation objectives. It includes specific facets such as

- to command, to require, to compel, and to make someone do something with supporting acts such as communicating, requesting, bespeaking, ordering, forbidding, prohibiting, interdicting, proscribing;
- to ask, to expect, to consider obligatory, to request and expect with specific supporting acts such as transmitting, communicating, calling for, demanding;
- to want, to need, to require, to have need of with supporting acts of wanting, needing, requiring;
- to necessitate, to ask, to postulate, to need, to take, to involve, to call for, to demand to require as useful, to just, or to proper.

The reasoning on operating, on relevant properties, on model objectives, on the model itself, on construction and assessment and guarantees can be characterised in a similar form.

The realisation stage may be replaced by other stage that support different purposes. We concentrated on prescription and construction of new systems. Another application is *model refinement*.

Design science aims at another kind of model refinement by adding more rigor after evaluation of a model. This refinement is essentially *model evolution* and *model evaluation*. Another refinement is the enhancement of models by concepts. This refinement is essentially a 'semantification' or *model conceptualisation* of the model.

Experimentation and justification of models is a third kind of adding rigor to (conceptual) models.

6.5 The Model of a Model

Models are often only considered with their intext, i.e., their structures und behaviour. Context is either neglected or taken for granted. We must however relate a model to the context dimension if we want to understand, to deploy or to modify the model.

Models follow typically some modelling schemata or pattern [5]. They are based on conceptions (concepts, theoretical statements (axioms, laws, theorems, definitions), models, theories, and tools). Conceptual processes include procedures, conceptual (knowledge) tools and associated norms and rules. Conceptions and conceptual processes are based on paradigms which are corroborated.

Models support interaction, understanding, sharing, and collaboration among people. They depend on existing knowledge, the actual (ontological) state of the reality, the condition of the person's senses and state of mind, and the state of employed instruments. Therefore, models depend on the background concepts that are accepted in a community.

We can summarise the considerations so far and develop a *general model frame*, i.e., a model of the model. It consists of four main components

Founding concepts: A model is based on paradigms, background theories, assumptions and guiding principles. It is composed of base conceptions/concepts with a certain scope, expressions, concept space organisation, and some quantification/measurement). Language-based models use a namespace or ontology as a carrier. This namespace is based on definitions made, i.e., the cargo in the sense of [15].

Structure and behaviour: A model is often build incrementally. Models can be multi-faceted (with a specific topology/geometry, with states, with interactions, with causal associations) or monolithic.

Application domain context: A model corresponds to a part of the reality, i.e. the application domain. The domain forms the empirical scope of the model. General or application-specific correspondence rules guide the association between the origin and the model. Each application domain is based on general laws one might have to consider for the model as well.

Meta-model: Models are developed within a theory and have a status within it. These theories provide the content of paradigms. Concepts are the most elementary building blocks. The construction process of a model is guided by the laws applicable to such theory. We may use basic models, emergent models, and subsidiary models.

Reusing the theories of concepts, content and topics [34], we shape the *general concept frame*. A concept is given by the scope, by at least one expression, by its association to other concepts and its media type [27] for the content. The application domain and potential functions constitute the scope of a concept. A concept can be defined by one or more partially synonymous expressions in a definition frame [36]. The concept space must follow some internal organisation. Concepts are interdependent and associated with each other. A concept must be underpinned and quantified by some data which use a certain format. We assume that the formatting can be given by a media type.

6.6 Model Fitness

[5] introduces *model viability* We extend this approach and consider fitness of a model. Fitness (or superior quality) of a model is given by

(a) *usability* of the model for its purpose, i.e., for resolving the questions, e.g., *validity* of the model;
(b) *potential* of the model for the purpose, i.e., for the goals that are satisfied by the model, e.g., *reliability* and *degree of precision* of the the model;
(c) *efficiency* of the model for the function of the model within the application, i.e., the practice [39] of deployment of the model;
(d) *generality* of the model beside its direct intention of construction of the model, i.e., for applying the model to other goals or purposes, within another function or with some modification or extension, e.g., the extend of *coverage* in the real world.

These four criteria form main quality characterisations of a model. Viability is defined through validity, reliability for the model purpose and function, extent of coverage in dependence on context such as space and time, and efficiency of the model. Viability

thus can be used to evaluate how well the model represents the reality for a given scope and how suitable or instrumental is the model for its purpose and function.

The *potential* of a model is defined by its strengthes, weaknesses, opportunities and threats. The potential can be assessed within a SWOT analysis. A model must be empirically corroborated in dependence on the objective. The abstraction property [31] determines the degree of corroboration. A model must be consistent with the context and the background and coherent in its construction. Models are parsimonious reductions of their origins. Due to this reduction models must be revisable for changes in reality. At the same time models must be relatively stable and robust against minor changes.

6.7 Observations for Information Systems Model Engineering

Engineering of conceptual models inherits both facets of didactically ruled learning [30] and of engineering [24]. The following characteristics of engineering sciences are observed also for conceptual modelling:

(α) The origin of a model is partly a product of creativity. Systems developed in our field are a product of developers and thus dependent on these stakeholders. They must be understood, well-explained and used with a purpose.

(β) The origin of a model is a complex systems. The attention focuses both on the creation of complex artifacts and on conceptualisiations of the application world. They are typically modularly constructed. Modularisation is only one of the underlying design principles. Conceptual modelling targets at useful concepts. It goes through a series of iterative design cycles in dependence on its purpose.

(γ) Models satisfy the purpose, are sharable, useful and reusable. Models are not developed just as an intermediate result of the implementation process and for satisfaction of purpose. They are shared within a community and are reusable in other situations. Moreover, models support a better understanding of the origins.

(δ) The origins are continuously changing and thus the models too. The application domain is continuously evolving. Models must correspondingly evolve too. Significant changes tend to be applied to the starting model so that the original concepts become unrecognisable after model evolution. Models are also used for changing the application world. This change must again be reflected within the model.

(ϵ) Origins being modelled are influenced by social constraints and affordances. Models are influenced as much by purposes as by physical and economical aspects of the contexts in which they are used. These influences are changing and evolving as well. Therefore, models are going to be used in ways their stakeholders did not imagine. Models are influenced as much by socially generated capital, constraints, and affordances as by the capabilities of stakeholders who created them.

(ζ) No single "grand theory" is likely to provide realistic solutions to realistic complex application problems. In realistic modelling situations that involve information systems, there almost never exist unlimited resources. Relevant stakeholders have typically conflicting goals. Therefore, ways of working displayed in Figure 3 usually need to integrate approaches drawn from different disciplines.

(η) Development of a model usually involves a series of iterative modelling cycles. Artifacts that serve as models are developed through a series of model activities and are iteratively tested and revised in dependence on the purpose.

Consequences for model engineering: The modelling process itself also changes the application domain and the understanding of the origin. Therefore, modelling is not reducible to condition-action rules. Modelling is a matter of engineering. Experienced modellers not only right develop a model but they also develop the right model - by developing models at the right time, with the right background and context, and for the right purpose. Model engineering is therefore based on advanced skills of handcrafting, i.e., making substitutions and adaptations depending on purpose and application situation, understanding which compositions perform best, continuously adapt the result of the process, and understand difficult-to-control things in their handcraft environment. The same situation is valid for information systems development if performance of the system becomes crucial and heavily depends on the DBMS.

7 Techniques for Modelling of Information Systems

7.1 Separation and Decomposition of the Workflows

Modelling integrates the classical problem solving four-phase cycle [22]:

1. Developing an understanding of the task: The task is analysed within its context and is compared with the goal for completion of the task. The initial situation is characterised. in the case of modelling processes the understanding is based on objectives derived from the purpose or from previous steps.
2. Development of a plan for the solution of a task: The instruments and tools for the solution of the task are reconsidered. The plan consists of heuristic forward and backtracking steps, steps for problem restructuring and for quality control.
3. Application of the plan for the development of the solution: The plan is consecutively applied for the generation of the solution. If certain steps are considered to be inappropriate then the plan is revised as well.
4. Development of an understanding of the solution: The result is evaluated based on criteria that either follow from the purpose or from the problem. Properties of the solution are derived.

This approach is based on four components:

The *state space* consists of the collection of all those states that are reachable from the *initial state*. Some of the states are considered to be desirable, i.e. are *goal states*. States can be modelled through languages such as ER. State may have properties such as suitability for certain purposes.

The *actions* allow to move from one state to another state under certain conditions. We may assume that the effect of the actions is observable to a certain extent by the user. User may use several actions in parallel. Actions may be blocked or enabled depending on conditions. Actions may be used at some cost.

The *goal test* determines whether a given state or state set satisfies the goals. The goal test may be defined through a set of states or through properties. The goal test may also allow to state which quality has the state set for the problem solution.

The *controller* evaluates the actions undertaken by the stakeholder. Some actions may be preferred over other, e.g. have less costs, or are optimal according to some optimality criterion. Controllers can be based on evaluators of the paths from the initial state to the current state.

Creation steps are the most complex steps in modelling. They typically consist of an orientation or review substep, of a development step performed in teams, and of a finalisation substep . Creation steps are composed of a number of substeps that can be classified into:

- Review of the state-of-affairs: The state of the development is reviewed, evaluated, and analysed. Obligations are derived. Open development tasks can be closed, rephrased or prioritised.
- Study of documents and resources: Available documents and resources are checked whether they are available, adequate and relevant for the current step, and form a basis for the successful completion of the step.
- Discussions and elicitation with other partners: Discussions may be informal, interview-based, or systematic. The result of such discussions is measured by some quality criteria such as trust or confidence. They are spread among the partners with some intention such as asking for revision, confirmation, or extension of the discussion.
- Recording and documentation of concepts: The result of the step is usually recorded in one work product or consistently recorded in number of work products.
- Classification of concepts, requirements, results: Each result developed is briefly examined individually and in dependence of other results from which it depends and to which it has an impact.
- Review of the development process: Once the result to be achieved is going to be recorded the work product is examined whether is has the necessary and sufficient quality, whether is must be revised, updated or rejected, whether there are conflicts, inconsistencies or incompleteness or whether more may be needed. If the evaluation results in requiring additional steps or substeps then the step or the substep is going to be extended by them.

7.2 Maieutics for Mastering Iterations

Modelling of information systems is not only aiming at achieving a nominal system but aims too at satisfaction of real interests of all stakeholders involved into modelling. It must consider all relevant aspects of an application and thus results in co-design of structuring, functionality, and supporting systems such as view and interaction support [33]. Stakeholders (or users) iteratively obtain a deeper insight and understanding about the necessities and conditions of the problem and the strengths, weaknesses, opportunities and threats of the solution depending of the purpose of the modelling within a modelling process. Therefore, modelling integrates ideas developed for maieutics [14, 17].

The maieutics frame [19] is essentially a specific form of a dialogue. In conceptual modelling, it consists (1) of an open-ended process, (2) of the elaboration of ideas that

are grounded in references to the application domain, to the users, prior knowledge and experience, and to the languages as carriers, and (3) of the discussion (in form of conceptualisation, interpretation, explanation, diverging ideas, and new understandings) that is inductive and exploratory rather than deductive and conclusive.

Modelling requires to utilise the knowledge in dependence on the purpose of the model. Answers found during modelling may not be evident in the material on hand; modellers may have to delve into subtleties or ambiguities they had not thought of. Information systems modelling is based on elaboration and conceptualisation of model elements. The inductive and exploratory discussion facilitates the development of argumentation by fostering the (re)consideration of alternatives and versions.

Conceptual modelling is based on references to the application domain, connections across the model, elaboration based on prior knowledge and/or experience, interpretations, explanations and conceptualisations, diverging ideas, and new understandings. Therefore the modelling process is highly iterative and revising/remastering decisions that have already been made.

7.3 Management and Support for Sub-workflows

For the modelling stage we derive the following general approach based on problem solving cycles:

initiation	differentiation and understanding	evaluation and selection by relevancy	model	justification and consensus	experimentation and field exploration	application
O	$\Phi(O)$	$\Psi(M)$	M	$\Phi(M)$	$\Psi(Y)$	Y
origin	origin properties	modelling objectives	artifact	properties	objectives	implementation

We therefore arrive at a modelling process in Figure 4 that refines the general workflow in Figure 2. We may zoom-in into these sub-workflows. For instance, one of the most interesting step is step (3) in the modelling activities. This step consists of a number of substeps. The conceptualisation stage is orthogonal within the database design framework in Figure 5.

Conceptualisation is based on the notion of concepts introduced in [12, 37]. Design science [4, 7, 16, 38] uses the rigor cycle as one of its three cycles aiming at model development. The rigor cycle has not yet been defined. We also arrive at a sub-workflow for conceptualisation in Figure 6.

8 Duties and the Task Spectrum in Conceptual Modelling

8.1 CMM and SPICE for Conceptual Modelling

A software process is considered to be the set of activities, methods, and practices used in the production and evolution of software [8] and the associated products [21]. For improving of a software process there are four main approaches: modelling, assessment, measurement, and technology adoption [25]. The approaches supplement each other, but one of them is usually in a dominating position. CMMI and SPICE (Software Process Improvement and Capability dEtermination) [9] are the two most widely

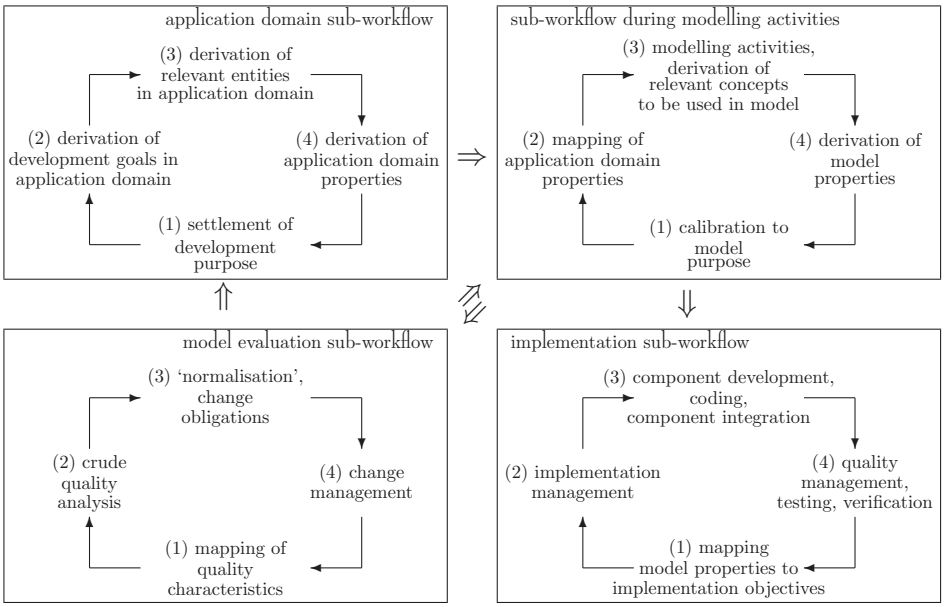


Fig. 4. The sub-workflows for construction modelling processes

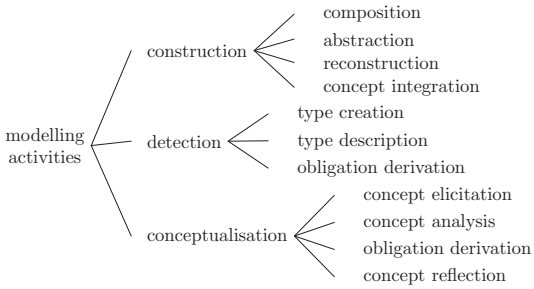


Fig. 5. Sub-steps of the modelling and concept derivation step in the modelling sub-workflow

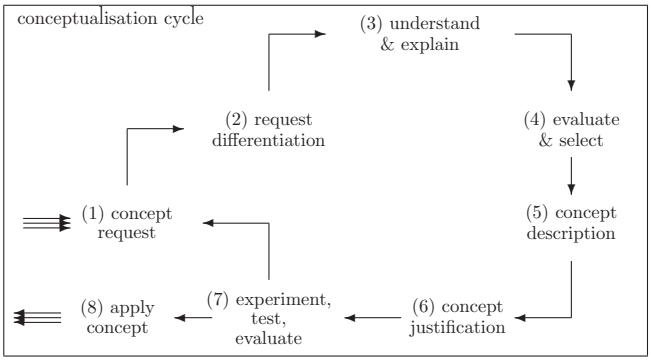


Fig. 6. The sub-workflow for conceptualisation steps within the modelling step

used software assessment models in software process improvement work today. The capability dimension consists of six *capability levels*: incomplete, performed, managed, established, predictable and optimizing.

Information system development is a specific software development process. Therefore, the SPICE characterisations are applicable as well. We may therefore distinguish different levels of the IS development *capability*:

1. *Performed and executed*: The goals of the application domain are satisfied. The information system development process is set out.
2. *Managed and defined*: Additionally, the application domain and scope are imaged by a model that allows to derive components of the system by means of model elements.
3. *Established and controlled*: The model is well-documented and allows to understand its design decisions. The model is used as a background and groundwork for the system.
4. *Understood, predictable and performed with sense*: The elements of the model are based on concepts that describe their semantics and meaning. The impact of languages as a model carrier, the assumptions made during design, the paradigms used during and the scope of the model are given in an explicit form.
5. *Optimised*: The model is developed with a number of alternatives. There are quantitative methods that support reasoning on quality of the model. Model alternatives can be given in a form that is the most adequate for the auditory. They can be used for deriving the best realisation.

8.2 The Duty Portfolio of Modelling

Following [6] we distinguish four main duties in conceptual modelling:

(1) Description: The application domain is described in a way that allows to comprehend the actual state, the necessities for system development and deployment, and the specifics and phenomena of the application.

(2) Explanation: The understanding of reality, of the processes and data in the application world and of the context of the application supports the creation of systems that effectively and efficiently support users. This understanding can be based on explication of concepts behind the application. It can also be based on behavioural pattern, on general laws and regulations and on user profiles and stories [26, 27].

(3) Creation: The system creation includes coding of the system, embedding the system into a systems context, developing supporting means for users, and supporting a new behaviour of users of a system. It uses the demands stated for the application, the analysis of the current state, and the requests for change by the system. Creation includes elements of SWOT analysis (*s*trengthes, *w*eaknesses, *o*pportunities, *t*hreats) and evaluation of the quality of the system.

(4) Prognosis: The behaviour of the augmented system, the opportunities of changes and evolution and the restrictions of the augmented reality are predicted. The user expectations and the reality of system exploitation are compared on the basis of main storyboards observed for the applications.

We may now combine these approaches into a process survey in Figure 7. The relevance cycle is based on observation of the state of affairs, scoping of the demands for system development, and describing a view of the application domain. These cycles form the y-dimension. We also use the x-dimension for explicit display of the changes imposed to the reality. Typically, information systems augment the reality. Figure 7 combines the approaches of design science (research) [7, 16] with those based on main duties for system development [6] and those typically used for conceptual modelling [33].

The design or modelling cycle uses the scoped application domain description for the development of a model. The rigor cycle adds semantics, meaning and context to the model. The description of the scoped application domain may directly be used for system development. For instance, agile development is typically following this direct approach. The model may also be directly used for system development. The advantage of such approach is that all relevant elements are supported by a model and that the model may be used for understanding the system. The system is therefore defined. We may also use the model for development of a behaviour description, guidelines (e.g., for system deployment) and documentation. In this case modelling is established.

Furthermore, we might background the model by concepts. In this case, users of the model may perform system construction with a sense of groundwork behind the model and the description of the application domain. Models may also be a part of a knowledge base. In this case we integrate, generalise and found the model through concepts in the knowledge base.

The relevance, design and rigor cycles are based on comprehension of the application domain, perception of the relevant elements and knowledge or understanding development for those elements. During system development models are used as a mediating artifact. They describe and image the problems, phenomena and demand form one side and serve as a prescription for systems development from the other side. Models may also serve as a background and foundation of the system if they are integrated with concepts.

9 Conclusion

Models are artifacts that can be specified within a (W^4+$W^{17}H$)-frame based on the classical rhetorical frame introduced by Hermagoras of Temnos[7].

1. They are primarily characterised by W^4: wherefore (purpose), whereof (origin), wherewith (carrier, e.g., language), and worthiness ((surplus) value).
2. Secondary characterisation $W^{17}H$ is given by:
   - user or stakeholder characteristics: by whom, to whom, whichever;
   - characteristics imposed by the application domain: wherein, where, for what, wherefrom, whence, what;

[7] Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (W^7: Who, what, when, where, why, in what way, by what means). The Zachman frame uses a simplification of this frame.

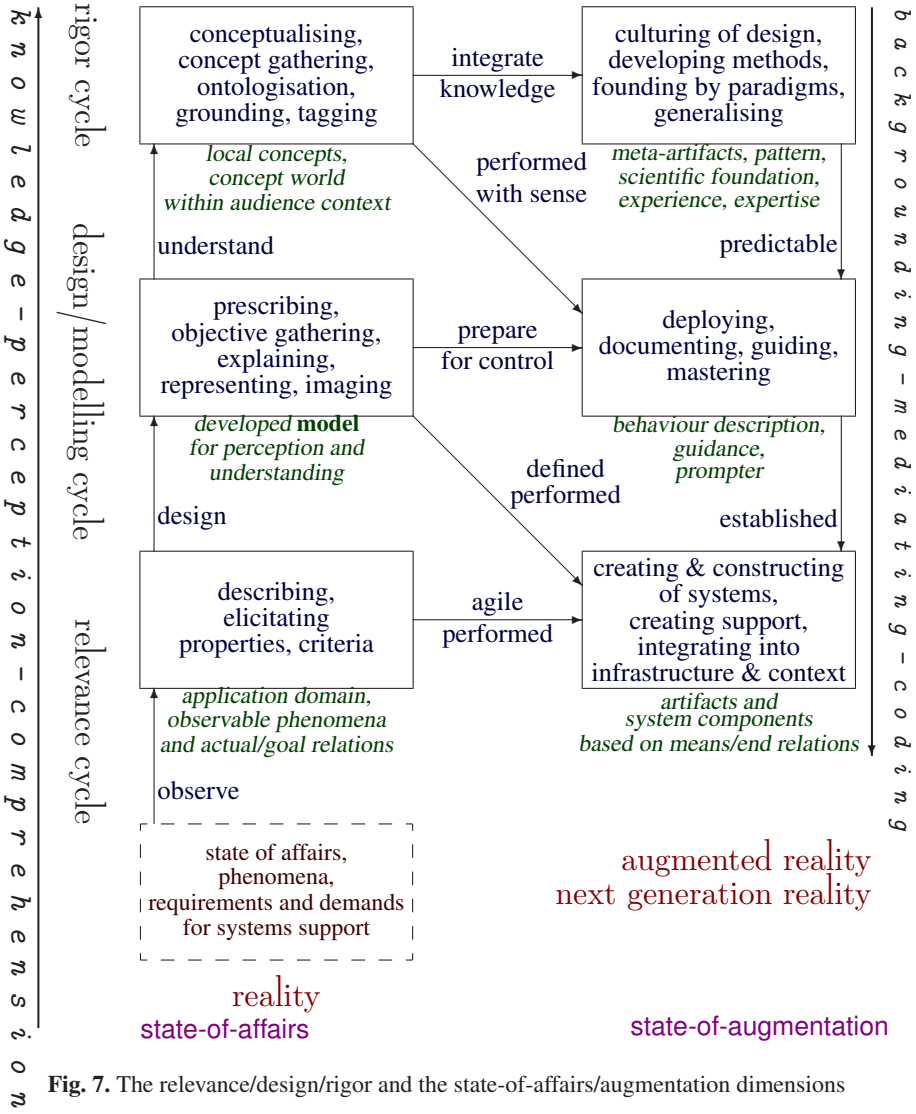


Fig. 7. The relevance/design/rigor and the state-of-affairs/augmentation dimensions

- purpose characteristics characterising the solution: how, why, whereto, when, for which reason; and
- additional context characteristics: whereat, whereabout, whither, when.

Modelling combines at the same time and systematically different aspects: culture, art, systematics and technology of model (re)development and model application. It uses modelling activities and techniques.

Conceptual modelling is biased through a pragmatical culture. It uses languages as a sophisticated medium of expression. It defines its specific arts and sciences. It reflects thoughts, i.e. perception, interpretation and understanding of people involved. It

is implicitly based on value systems transmitted through communities of practice based on some commonsense and consensus. Conceptual modelling is also at the same time a social activity, i.e. a shared pursuit within a community, demonstrated in a variety of textbooks, publications and conferences. Conceptual models are used for social aspects, i.e. include the give-and-take of socialisation, negotiation, protocol, and conventions within the community of their users. These aspects of models and of modelling activities collectively redefine conceptual modelling as a culture.

A general theory of models and modelling contains also other considerations: models as constructs that use signs, practices of model deployment, conditions for model functioning, cognitive and epistemic functions, normative functions, status and role of modelling, boundaries and reach of models, reflections on model deployment, etc. In Computer Science more specific questions must be taken into account such as the following ones: elements of model purposes, functioning of models, genesis, model capacity, added value of models, analysis of the general and epistemic role of a model, dependence of models in the context of other artifacts, functional reach of a model, etc. Models are elements of the Computer Science culture. As such we need to consider problems of model quality characteristics, of difference development and discovery, of integration into knowledge, of capability for system performance and prognosis, of the social impact of models within communities, of integration of intuition and vision, of parallelization and coexistence of models, of importance for feedback, of dependence of models on the carrier (language), of approximation and reduction, of abstraction, etc. These questions are problems for future research.

References

1. D. Bjørner. *Software Engineering 3: Domains, requirements, and software design*. Springer, Berlin, 2006.
2. D. Bjørner. *Domain engineering*, volume 4 of *COE Research Monographs*. Japan Advanced Institute of Science and Technolgy Press, Ishikawa, 2009.
3. P.J. Denning. Great principles of computing. http://cs.gmu.edu/pjd/GP/, 2007.
4. S. Gregor and D. Jones. The anatomy of a design theory. *Journal of Association for Information Systems*, 8(5):312–335, 2007.
5. I.A. Halloun. *Modeling Theory in Science Education*. Springer, Berlin, 2006.
6. L.J. Heinrich, A. Heinzl, and R. Riedl. *Wirtschaftsinformatik: Einführung und Grundlegung*. Springer, Berlin, 4 edition, 2011.
7. A. Hevner, S. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quaterly*, 28(1):75–105, 2004.
8. W.S. Humphrey. *Managing the Software Process*. Addison-Wesley, 1989.
9. ISO/IEC. Information technology - process assessment - part 2: Performing an assessment. IS 15504-2:2003, 2003.
10. H. Jaakkola and B. Thalheim. Framework for high-quality software design and development: a systematic approach. *IET Software*, 4(2):105–118, 2010.
11. R. Kaschek. *Konzeptionelle Modellierung*. PhD thesis, University Klagenfurt, 2003. Habilitationsschrift.
12. Y. Kidawara, K. Zettsu, Y. Kiyoki, K. Jannaschk, B. Thalheim, P. Linna, H. Jaakkola, and M. Duzí. Knowledge modeling, management and utilization towards next generation web. In

Information Modelling and Knowledge Bases XXI, volume 206, pages 387–402. IOS Press, 2010.
13. G. Klaus and M. Buhr, editors. *Philosophisches Wörterbuch*. VEB Bibliographisches Institut, Leipzig, 1971.
14. H. Krauch. System analysis. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon zur Wissenschaftstheorie*, pages 338–344. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.
15. B. Mahr. Information science and the logic of models. *Softw. Syst. Model.*, 8:365–383, 2009.
16. S.T. March and V.C. Storey. Design science in the information systems discipline: An introduction to the special issue on design science research. *MIS Quarterly*, 4:725–730, 2008.
17. M.D. Mesarovic and Y. Takahara. *General systems theory: Mathematical foundations*. Academic Press, New York, 1975.
18. J. Mittelstraß, editor. *Enzyklopädie Philosophie und Wissenschaftstheorie*. J.B. Metzler, Stuttgart, 2004.
19. P. Orellana. *Maieutic frame presense and quantity and quality of argumentation in a Paideia seminar*. Doctor of philosophy, University of North Carolina at Chapel Hill, 2008.
20. E. Ortner and B. Schienmann. Normative language approach - a framework for understanding. In *Proc. 15th Int. ER Conf., Conceptual Modeling - ER'96*, LNCS 1157, pages 261–276. Springer, Berlin, 1996.
21. M.C. Paulk, B. Curtis, M.B. Chrissis, and C.V. Weber. Capability maturity model for software, version 1.1. Technical Report CMU/SEI-93-TR-024, Software Engineering Institute, February 1993.
22. G. Polya. *How to solve it: A new aspect of mathematical method*. Princeton University Press, Princeton, 1945.
23. J.E. Safra, I. Yeshua, and et. al. *Encyclopædia Britannica*. Merriam-Webster, 2003.
24. A. Samuel and J. Weir. *Introduction to Engineering: Modelling, Synthesis and Problem Solving Strategies*. Elsevier, Amsterdam, 2000.
25. S. Saukkonen and M.Oivo. Six step software process improvement method (in finnish; teollinen ohjelmistoprosessi. ohjelmistoprosessin parantaminen SIPI-menetelmällä). Tekes 64/98, Teknologiakatsaus, October 1998.
26. K.-D. Schewe and B. Thalheim. Reasoning about web information systems using story algebra. In *ADBIS'2004*, LNCS 3255, pages 54–66, 2004.
27. K.-D. Schewe and B. Thalheim. Usage-based storyboarding for web information systems. Technical Report 2006-13, Christian Albrechts University Kiel, Institute of Computer Science and Applied Mathematics, Kiel, 2006.
28. K.-D. Schewe and B. Thalheim. Semantics in data and knowledge bases. In *SDKB 2008*, LNCS 4925, pages 1–25, Berlin, 2008. Springer.
29. G. Simsion. *Data modeling - Theory and practice*. Technics Publications, LLC, New Jersey, 2007.
30. B. Sriraman and L. English. *Theories about mathematics education*. Springer, Berlin, 2010.
31. H. Stachowiak. Modell. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon zur Wissenschaftstheorie*, pages 219–222. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.
32. W. Steinmüller. *Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993.
33. B. Thalheim. *Entity-relationship modeling – Foundations of database technology*. Springer, Berlin, 2000.
34. B. Thalheim. The conceptual framework to user-oriented content management. *Series Frontiers in Arificial Intelligence, 154, Information Modelling and Knowledge Bases*, XVII:30–49, 2007.

Syntax, Semantics and Pragmatics of Conceptual Modelling

Bernhard Thalheim

Computer Science Institute, Christian-Albrechts-University Kiel,
Olshausenstrasse 40, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract

Models, modelling languages, modelling frameworks and their background have dominated conceptual modelling research and information systems engineering for last four decades. Conceptual models are mediators between the application world and the implementation or system world. Currently conceptual modelling is rather a craft and at the best an art. We target on a science and culture of conceptual modelling.

Models are governed by their purpose. They are used by a community of practice and have a function within application cases. Language-based models use a language as a carrier. Therefore, semiotics of models must be systematically developed. This paper thus concentrates on the linguistic foundation of conceptual modelling.

1 Introductory Notions for Conceptual Modelling

Conceptual modelling is a widely applied practice in Computer Science and has led to a large body of knowledge on constructs that might be used for modelling and on methods that might be useful for modelling. It is commonly accepted that database application development is based on conceptual modelling. It is however surprising that only very few publications have been published on a *theory of conceptual modelling*. We continue our research [2, 3, 15, 16, 17, 18] and aim in a theory of linguistic foundations for modelling within this paper.

1.1 Goals, Purposes and Deployment Functions of Conceptual Models

Purpose is often defined via intention and mixed with function. *Goal* (or intention or target or aim) is a ternary relation between a current state, envisioned states, and people (community of practice). Typical - sometimes rather abstract - intentions

are perception support, explanation and demonstration, preparation to an activity, optimisation, hypothesis verification, construction, control, and substitution.

The *purpose* is a binary relation between intentions and means or instruments for realisation of the intention. The main mean we use is the language. Semiotics is widely intentionally used for modelling; however without paying attention to it.

The *deployment function* of a model relates the model purpose to a practice or application cases or application 'game' similar to Wittgenstein's language game (we call it better *deployment case* and is characterised by answering the classical W-questions: how, when, for which/what or why, at what/which (business use case), etc. We add to purpose: application, conventions, custom, exertion, habit, handling, deployment, service, usage, use, and way of using. The model has a role and plays its behaviour within this application game.

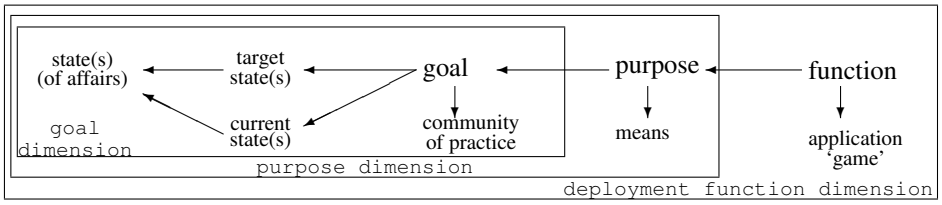


Figure 1: Distinction of Intention, Purpose,and Deployment Function of a Model

1.2 Abstraction and Conceptual Modelling

Abstraction is one of the most overloaded conceptions in Computer Science and at the same time one of the most under-specified. Abstraction means development of general concepts by abstracting common properties of specific concepts. Using the approach in [14] we develop three dimensions of abstraction:

Structural abstraction is used for highlighting essential, necessary, and general structural elements of the origin. Structural abstraction has three main constituents: *combining structural abstraction* (often called) *classifying structural abstraction* (often called meta(-meta(-meta)) abstraction) combines things of interest into collections that contain these things as elements; *generalising structural abstraction* (often called pattern or templates).

Context abstraction "factors out" repeating, shared or local patterns of things and functionality from individual things. Context abstraction assumes that the surroundings of a things under consideration are commonly assumed by a community of practice or within a culture and focuses on the concept, turning away attention from its surroundings such as the environment and setting. Models use ambiguities, ellipses, metaphors and commonly assumed conceptions.

Behavioural abstraction is used for concentrating of essential and general behavioural elements. We may distinguish between *combining*, *classifying*, and *generalising behavioural abstraction*. Aspect separation, encapsulation, and modularisation are specific techniques.

The opposite of abstracting is detailing. *Refinement* is a specific kind of faithful detailing. Refinement uses the principle of modularisation and information hiding. Developers typically use conceptual models or languages for representing and conceptualising abstractions. Classical forms of abstraction are generalisation, isolation, and idealisation.

We thus may concentrate on three main tasks for abstraction within a community of practice:

- *Choose the right scope* within the application area in dependence on goals.
- *Choose the right focus* at the right level and in the right granularity .
- *Choose the right observation* with the right behaviour and right properties.

2 The Notion of the Model

It is noted (e.g. [1]) (it can be observed indeed also in almost all textbooks on Computer Science) that there is **no** commonly accepted adequate definition of the concept of the model. This claim is considered as one of the *big lacuna* of the science and art of modelling.

A model is simply a material or virtual *artifact* (1) which is called model within a community of practice (2) based on a judgement (3) [7] of appropriateness for representation of other artifacts (things in reality, systems, ...) and serving a *purpose* (4) within this community. We distill thus criteria for artifacts to become a model. We can use on two approaches: abstract properties or we criteria for artifacts.

2.1 Stachowiak, Aristoteles, Galilei and Mahr Properties of Models

Models are often defined through abstract properties they must satisfy [15, 18].

(1) *Mapping* property: Each model has an origin and is based on a mapping from the origin to the artifact.
(2) *Truncation* property: The model lacks some of the ascriptions made to the original and thus functions as an Aristolean model by abstraction of irrelevant.
(3) *Pragmatic* property: The model use is only justified for particular model users, tools of investigation, and period of time.
(4) *Amplification* property: Models use specific extensions which are not observed for the original,

(5) *Distortion* property: Models are developed for improving the physical world or for inclusion of visions of better reality, e.g. for construction via transformation or in Galilean models.
(6) *Idealisation* property: Modelling abstracts from reality by scoping the model to the ideal state of affairs.
(7) *Carrier* property: Models use languages and are thus restricted by the expressive power of these languages.
(8)*Added value* property: Models provide a value or benefit based on their utility, capability and quality characteristics.
(9) *Purpose* property: Models and conceptual models are governed by the purpose. The model preserves the purpose.

The first three properties have been introduced by Stachowiaks [12]. The fourth and fifth property have been introduced by Steinmüller [13]. The seventh property is discussed by Mahr [10]. The sixth, eight and nine properties [18] are however equally if not the most important ones.

2.2 Criteria for Appropriateness of an Artifact to Become a Model

The separation into goal, purpose and deployment function for models provides three main appropriateness criteria:
(1) The *adequacy* of a model defines its *potential* for the goals. Adequacy is given by the similarity of the model with its origin in dependence on its goal, the regularity for the application (within a well-founded system that uses rules for derivation of conclusions), the fruitfulness (or capacity) for goals, and the simplicity of the model through the reduction to the essential and relevant properties in dependence on the goal.
(2) A model is *fit* for its purpose if it is *usable* for the purpose, *suitable* within the given context and for the prescribed purposes, *robust* against small changes in the parameters, *accurate* to the level of precision that is necessary for the purpose, and *compliant* with the funding concepts, application context and meta-model.
The model must be *testable* and, if false, it can be disconfirmed by a finite set of observations (finitely testable) and by any of superset of these observation (irrevocably testable).
(3) The *usefulness* for deploying is given by *effectiveness* for complete and accurate satisfaction of the goal, *understandability* for purposeful deployment of the model by users, *learnability* of the model within the deployment stories, *reliability* and a high *degree of precision* of the the model, and *efficiency* of the model for the function of the model within the application. *testability*

Additional criteria are *generality* of the model beside its direct goals and intentions and the extend of *coverage* in the real world for other goals.

2.3 The Hidden Background of Models

The structure and function of a model are based on a correspondence relation between the reality or the augmented and the model. The model can be constructed incrementally. It represents a number of facets of the origin (topology/geometry, state, interaction, causal). The *model pragmatism* is however hidden. It consists of at least three background dimensions:

Founding concepts: A model uses the cultural background within the application area and within a community of practice. Base conceptions (scope, expressions, concept space organisation, quantification/measurement), a namespace/ontology/carrier, a number of definitions (state, intrinsic, object, interaction descriptors and depictors), and a language as cargo [10] characterise these founding concepts.

Application context: The application domain binds the model to some common understanding that is not explicitly defined in the model. Each model has an empirical scope of the model, has specific application-domain driven correspondences, and must satisfy a number of laws and regulations.

Meta-model: Each model has a basement, is restricted by paradigms and theories, has a status in the application; context, displays elements on certain abstraction level and granularity, and uses a scale. It is also prone for paradigmatic evolution within the epistemological profile of community of practice.

These dimensions are partially known in didactics (of modelling) [5]. The three background dimensions drive however the model deployment and development.

3 Language-Backed Modelling

3.1 Language Selection Matters

Languages may however also restrict modelling. This restriction may either be compensated by over-development of language components or by multi-models. The relational database modelling language uses integrity constraint as compensation component for the inadequate expressibility of the language. The Sapir-Whorf hypothesis [19] results in the following principle:

***Principle of linguistic relativity*:** Actors skilled in a language may not have a (deep) understanding of some concepts of other languages. This restriction leads to problematic or inadequate models or limits the representation of things and is not well understood.

The principle of linguistic relativity is not well understood. In [15] we demonstrated via a crossroad example that Petri nets are often not the right tool for representation of behaviour. A similar observation on UML is made by Krogstie [9].

3.2 The Cognitive Insufficiency of the Entity-Relationship Modelling Language

Lakoff introduces six basic schemata of cognitive semantics without stating that this list of schemata is complete.

- The container schema define the distinction between in and out. They have an interior, a boundary and an exterior.
- The part-whole schema define an internal structuring and uses whole, part and configuration as construction units.
- The link schema connects thing of interest. It uses various kinds of links for associating or un-associating things.
- The center-periphery schema is based on some notion of a center. Peripherical elements are not as important than those in the center.
- The source-path-goal schema uses source (or starting point), destination, path, and direction. It allows also to discuss main and side tracks.
- Typical ordering schemata are the up-down, front-back and the linear ordering schema. They use spatial and temporal associations.

We call a modelling language *cognition-complete* if these six schemata can be represented.

The classical ER modelling language suffers from a number of restrictions. It uses the container and the link schemata. It allows to mimic the part-whole schema via special links (called IsA). This work-around is however badly misunderstood. In order to become cognition-complete integrity constraints must be used. Their cognitive complexity is however beyond surveyability of humans. A typical flaw of the classical ER model is the use of monster types that integrate stabile - almost not changing - properties and transient - often changing - properties. Objects are then taken as a whole. Unary relationship types easily resolve this problem if higher-order types are permitted.

Extended ER modelling languages are however also not cognitive complete. The center-periphery schema can only be emulated. The source-path-goal schema can be represented by higher-order relationship types. The part-whole schema is supported by the specialisation via unary relationship types and by generalisation via cluster types. Ordering schemata can be defined using the order types and bulk types.

4 Syntax of Conceptual Models: Structuring and 'Functioning'

Syntax of models is build on deictic context-based rules [20] for construction of complex expressions using a domain-dependent vocabulary and governed by a set of meta-rules for construction (styles, pattern, abstraction).

4.1 Morphology of Conceptual Models and the Form of Elements

Morphology is the science of word form structure. The part of syntax is completely neglected in conceptual modelling. It is however equally important. Elements of a modelling language can be classified according to their categories and roles within a model and according to their specific expression within a model. Expression might similarly ruled by inflection, deviation, and composition. Therefore, techniques like lemmatisation (reduction of words to their base form) and characterisation by the (morpho-syntactic) role within a model.

Based on [11] we distinguish five morphological features: full or partial specification, layering within a model, integrity constraints, cyclic or acyclic structuring, complete set of schemata for cognitive semantics, open or closed context, and kind of data types.

Syntax for models is context-dependent. Constructs are bound by an implicit construction semantics [14] that is an integral component of any language. Models are governed by syntactic rules or explicit and implicit social norms. They are constructed with implicit styles and architectures.

4.2 The Lexicography and the Namespace of Models

Lexicography has developed a number of principles for coding and structuring lexical elements based on the lexicon on the application domain. Most ontological research in Computer Science does not got beyond lexicography and uses a topical annotation of model elements while hoping that every stakeholder has the same interpretation for words such as 'name', 'description', 'identifier' etc. If we consider however more complex entries such as 'address' then we detect that such kind of annotation does not work even for one language. The situation becomes far worse if we consider different languages, cultures, or application domains. Then the nightmare of "integration" becomes a challenge.

Models typically use a general and an application-dependent namespace. Moreover, the model is a product of a community of practice with its needs, its commonspeak, its specific functions of words, its specific phrases and abbreviations, and its specific vocabulary.

5 Semantics of Conceptual Models

5.1 Kinds of Semantics

Semantics is the study of meaning, i.e. how meaning is constructed, interpreted, clarified, obscured, illustrated, simplified, negotiated, contradicted and paraphrased. It has been treated differently in the scientific community, e.g., in the area of knowledge bases and by database users.

- The scientific community prefers the treatment of 'always valid' semantics based on the mathematical logic. A constraint is valid if this is the case in any correct database.
- Database modellers often use a 'strong' semantics for several classes of constraints. Cardinality constraints are based on the requirement that databases exist for both cases, for the minimal and for the maximal case.
- Database mining is based on a 'may be valid' semantics. A constraint is considered to be a candidate for a valid formula.
- Users usually use a weak 'in most cases valid' semantics. They consider a constraint to be valid if this is the usual case.
- Different groups of users use an 'epistemic' semantics. For each of the group its set of constraints is valid in their data. Different sets of constraints can even contradict.

Semantics is currently one of the most overused notions in modern computer science literature. Its understanding spans from synonyms for structuring or synonyms for structuring on the basis of words to precise defined semantics. This partial misuse results in a mismatch of languages, in neglecting formal foundations, and in brute-force definitions of the meaning of syntactic constructions.

Semantics of models uses also commonsense, intended and acceptable meanings, various kinds of quantifications, (logical) entailment, deduction, induction, and abduction.

5.2 The Lexicology and the Namespace of Models

Ontologies are becoming very popular in Computer Science research. Philosophy developed now a rather restrained usage of ontologies. Lexicology [4] - as a part of philology - or semasiology is based on semantic relations in the vocabulary of a language. Lexicology of models studies elements of models and their meaning, relations between these elements, sub-models and the namespace in the application

domain. Classical linguistic relations such as homonym, antonym, paronym, synonym, polysemy, hyponym, etc. are used for stereotyped semantics in the namespace.

Models combine two different kinds of meaning in the namespace: referential meaning establishes an interdependence between elements and the origin ('what'); functional meaning is based on the function of an element in the model ('how'). The referential meaning is well investigated and uses the triangle between element, concept and referent. The functional meaning relates elements in a model to the model context, to the application context, and to the function of this element within the model. It thus complements the referential meaning. Additionally model lexicology use the intext (within the model), the general, the part-of-model, and the differential (homonym-separating) meaning. Further, we need to handle the change of meaning for legacy models.

6 Pragmatics of Conceptual Models

6.1 General Pragmatics of Modelling in a Community of Practice

Pragmatics for modelling is the study how languages are used for intended deployment functions in dependence on the purposes and goals within a community of practice. Functions, purposes and goals are ruling the structure and function of the model. We distinguish the descriptive-explanatory and persuasive-normative functions of a model. Models are used for (1) acting (2) within a community, especially the modeller and have (3) different truth or more generally quality [8]. We may distinguish between *far-side* and *near-side* pragmatics separating the 'why' from the 'what' side of a model. General pragmatics allows to describe the overall intentions within the community and strategies for intention discovery.

We may distinguish between the development and deployment modes. The first mode starts with abstraction and mapping and then turns to the representation. The second mode inverses the first mode. Based on this distinction we infer a number of basic principles of modelling pragmatics similar to Hausser [6]: model surface compositionality (methodological principle), model presentation order's strict linearity relative to space (empirical principle), model interpretation and production analysed as cognitive processes (ontological principle), reference modelled in terms of matching an model's meaning with context (functional principle).

Models must be methodologically valid, support subjective deductive (paradeductive) inference with an open world interpretation, allow context-dependent reasoning (implicature), provide means for collaborative interaction, weaken connectives and quantifications, and integrate deductive, inductive, abductive and paradeductive reasoning.

6.2 Visualisation or the 'Phonetics' of Conceptual Models

Phonology is the science of language sounds. *Phonetics* investigates the articulatory, acoustic, and auditive process of speech. It is not traditionally considered not considered to be a part of a grammar. But it is equally important for practical language deployment. Phonology of models is concerned with the ways in which intentions can be conveyed using conventional and non-conventional resources. The modeller uses reference for transferring the specific point of view that has been used for modelling.

Visualisation is the 'phonetics' in modelling. It is based on three principles:

Principles of visual communication are based on three constituents: *Vision*, *cognition*, and *processing and memorizing characteristics*. We may use specific visual features such as contrast, visual analogies, presentation dramaturgy, reading direction, visual closeness, symmetric presentation and space and movement.

Principles of visual cognition refer to *ordering*, *effect* delivery, and *visualisation*. We base those on *model organisation*, *model economy*, *skills of users*, and *standards*.

Principles of visual design are based on *optical vicinity* , *similarity* , *closeness*, *symmetry*, *conciseness*, *reading direction*.

These principles help to organise the model in a way that correspond to human perception.

7 Summary

Models are artifacts that can be specified within a (W^4+$W^{17}H$)-frame that is based on the classical rhetorical frame introduced by Hermagoras of Temnos[1]. Models are primarily characterised by W^4: wherefore (purpose), whereof (origin), wherewith (carrier, e.g., language), and worthiness ((surplus) value). Secondary characterisation $W^{17}H$ is given by:

- user or stakeholder characteristics: by whom, to whom, whichever;
- characteristics imposed by the application domain: wherein, where, for what, wherefrom, whence, what;

[1]The rhetor Hermagoras of Temnos, as quoted in pseudo-Augustine's De Rhetorica defined seven "circumstances" as the loci of an issue: Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis(W^7: Who, what, when, where, why, in what way, by what means). See also Cicero, Thomas Aquinas, and Qunitillian's loci argumentorum as a frame without quesrioning. The Zachman frame uses an over-simplification of this frame.

- purpose characteristics characterising the solution: how, why, whereto, when, for which reason; and
- additional context characteristics: whereat, whereabout, whither, when.

Modelling is the art, the systematics and the technology of model (re)development and model application. It uses model activities and techniques. This paper is going to be extended by more specific aspects of the modelling art in the context of semiotics and linguistics.

References

[1] J. Agassi. Why there is no theory of models. In I. Niiniluoto W.E. Herfel, W. Krajewsky and R. Wojcicki, editors, *Theories and Models in Scientific Processes*, pages 17–26, Amsterdam-Atlanta, 1995.

[2] A. Dahanayake and B. Thalheim. Towards a framework for emergent modeling. In *ER Workshops*, volume 6413 of *Lecture Notes in Computer Science*, pages 128–137. Springer, 2010.

[3] A. Dahanayake and B. Thalheim. Enriching conceptual modelling practices through design science. In *BMMDS/EMMSAD*, volume 81 of *Lecture Notes in Business Information Processing*, pages 497–510. Springer, 2011.

[4] R.S. Ginsburg, s.S. Khidekei, G.Y. Knyazeva, and A.A. Sankin. *A course in modern English lexicology*. Vysschaja Schkola, Moscov, 2nd edition, 1979. (In Russian).

[5] I.A. Halloun. *Modeling Theory in Science Education*. Springer, Berlin, 2006.

[6] R. Hausser. *Foundations of computational linguistics - human-computer communication in natural language (2. ed.)*. Springer, 2001.

[7] R. Kaschek. *Konzeptionelle Modellierung*. PhD thesis, University Klagenfurt, 2003. Habilitationsschrift.

[8] K. Korta and J. Perry. *Critical Pragmatics*. Cambridge University Press, Cambridge, 2011.

[9] J. Krogstie. Quality of UML. In *Encyclopedia of Information Science and Technology (IV)*, pages 2387–2391. 2005.

[10] B. Mahr. Information science and the logic of models. *Software and System Modeling*, 8(3):365–383, 2009.

[11] T. Ritchey. Outline for a morphology of modelling methods -Contribution to a general theory of modelling. *Acta Morphologica Generalis*, 1(1):1–20, 2012.

[12] H. Stachowiak. Modell. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon zur Wissenschaftstheorie*, pages 219–222. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.

[13] W. Steinmüller. *Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993.

[14] B. Thalheim. *Entity-relationship modeling – Foundations of database technology*. Springer, Berlin, 2000.

[15] B. Thalheim. Towards a theory of conceptual modelling. *Journal of Universal Computer Science*, 16(20):3102–3137, 2010. http://www.jucs.org/jucs_16_20/towards_a_theory_of.

[16] B. Thalheim. The art of conceptual modelling. In *Proc. EJC 2011*, pages 203–222, Tallinn, 2011.

[17] B. Thalheim. The science of conceptual modelling. In *Proc. DEXA 2011*, volume 6860 of *LNCS*, pages 12–26, Berlin, 2011. Springer.

[18] B. Thalheim. The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 12, pages 543–578. Springer, Berlin, 2011.

[19] B.L. Whorf. *Lost generation theories of mind, language, and religion*. Popular Culture Association, University Microfilms International, Ann Arbor, Mich., 1980.

[20] R. Wojcicki. *Wyklady z logiki z elementami teorii wiedzy*. Wyd. Naukowe SCHOLAR, 2003. (In Polish).

The Conceptual Model $\equiv$ An Adequate and Dependable Artifact Enhanced by Concepts

Bernhard THALHEIM [1]
Christian-Albrechts University Kiel, Computer Science Institute, 24098 Kiel, Germany

Abstract. Conceptional modelling is one of the central activities in Computer Science. Conceptual models are mainly used as intermediate artifact for system construction. The notion of the conceptual model is at present rather informal. Conceptional modelling is performed by a modeller who directs the process based on his/her experience, education, understanding, intention, and attitude.

This paper develops a definition of the model and the conceptual model that encompasses model notions used in Computer Science, Mathematics, Natural and Social Sciences, and Humanities. Models are artifacts which have a background, a basis, and a context. They are products that are used by community of practice such as programmers, learners, business users, and evaluators. Models must be adequate for the representation of origins, dependable for their signification, functional; thus providing the necessary capability and be able to provide effective deployment.

Keywords. model, models in science, model theory, modelling.

1. Introduction

It is noted (e.g. [1]) (it can be observed indeed also in almost all textbooks on Computer Science) that there is **no** commonly accepted adequate definition of the concept of the model. This claim is considered as one of the *big lacuna* of the science and art of modelling.

As a starting point, a model can be simply considered to be a material or virtual *artifact* (1) which is called model within a community of practice (2) based on a judgement (3) [7] of appropriateness for representation of other artifacts (things in reality, systems, ...) and serving a *purpose* (4) within this community. We distill in the sequel criteria for an artifact to become a model. We can use two approaches: abstract properties and criteria for artifacts. We can observe, however, that most properties of models are hidden or implicit, and thus not given. The user must be a member of a community and base his/her understanding on the background accepted in this community. This implicitness is the main source of misunderstandings concerning models.

Additionally, models are used in many different deployment scenarios and stories (*Gebrauchsspiel* [22]). They are used for certain purposes and have a function within their deployment. Often they should not or cannot be used outside of their purpose.

[1] thalheim@is.informatik.uni-kiel.de http://www.is.informatik.uni-kiel.de/∼ thalheim

1.1. Stachowiak, Aristoteles, Galilei and Mahr Properties of Models

Models are often defined through abstract properties that they must satisfy [14,15].

(1) *Mapping* property: each model has an origin and is based on a mapping from the origin to the artifact.
(2) *Truncation* property: the model lacks some of the ascriptions made to the original and thus functions as an Aristolean model by abstraction by disregarding the irrelevant.
(3) *Pragmatic* property: the model use is only justified for particular model users, the tools of investigation, and the period of time.
(4) *Amplification* property: models use specific extensions which are not observed in the original.
(5) *Distortion* property: models are developed for improving the physical world or for inclusion of visions of better reality, e.g. for construction via transformation or in Galilean models.
(6) *Idealisation* property: modelling abstracts from reality by scoping the model to the ideal state of affairs.
(7) *Carrier* property: models use languages and are thus restricted by the expressive capacity of these languages.
(8) *Added value* property: models provide a value or benefit based on their utility, capability and quality characteristics.
(9) *Purpose* property: models and conceptual models are governed by the purpose. The model preserves the purpose.

The first three properties were introduced by Stachowiak [12]. The fourth and fifth property were introduced by Steinmüller [13]. The seventh property is discussed by Mahr [9]. The sixth, eight and ninth properties [15] are, however, equally if not the most important ones.

1.2. The Manifold of Meaning for the Word "Model"

The notion 'model' carries many meanings: Encyclopedia Britannica [10] contains two main meanings:

1. A models is a miniature representation of something (a model of the dam that was accurate down to the last detail). Synonyms are miniature, and pocket edition. Related words are copy, mock-up, replica, reproduction; dummy, and effigy.
2. A model is something set or held before one for guidance or imitation (Samuel Johnson's literary style is often used as a model for writers seeking precision and clarity). Synonyms are in this case archetype, beau ideal, ensemble, example, exemplar, ideal, mirror, paradigm, pattern, standard. We may compare this meaning with the word paragon. Related words are (2.1) apotheosis, nonesuch, nonpareil, paragon; (2.2) emblem, symbol, type; (2.3) embodiment, epitome, quintessence; and (2.4) criterion, gauge, touchstone.

The synonyms.org dictionary presents a more detailed picture for the German word 'Modell'. It consists of 111 synonyms that can be arranged into 10 groups[2].

[2]The groups are split along rather generalised lines. We have used the most characteristic synonyms for the presentation of the categories.

(1) Material representation object: image, statuary, artwork, sculpture, figure, illustration, gestalt, marionette, ornament;

(2) Derived object: derivative, derivable, fork, spinoff, descendant;

(3) produced object: artifact, product, article, result, creation, commodity, achievement, master piece, oeuvre, opus, works;

(4) Completed object: completion, achievement, design, accomplishment, implementation, conversion, variant;

(5) Constructed object: construction, style of construction, pattern, sample;

(6) Class of construction: construction, pattern, mock-up, archetype, type, kind;

(7) Typical object: sample, paradigm, probe, prototype, schema;

(8) Starting object: initial release, original (edition), draft, guideline, oddball, fogy;

(9) Ideal object: ideal, overall concept, antetype, paragon;

(10) Generalised object: concept, pattern, plan, principle, schema, standard.

This variety of the definition can, however, be represented by pairs *(meaning, purpose)*. The definition of the model depends on the use of models. The use can be characterised by processes, i.e. the *Gebrauchsspiel* in the sense of Wittgenstein. We relate the definitions of 'model' to deployment or exploitation functions and corresponding workflows [16]:

Description-prescription function: models as images, figures, standard, opus, exposition, representation, composition, realisation;

Explanation function: models 'gestalt', pattern, guidance, typ, family or species, original, concept, principle, form, workout;

Optimisation-variation function: models as creation, ideal, achievement, probe, article, plan, variant, substitute;

Verification-validation-testing function: models as sample, schema, specimen, pattern;

Reflection-optimisation function: models as creation, design, construction, type, derivative, master piece, product;

Explorative function: models as result, product, work, art piece, metaphor, paradigm, first edition, style, realisation, artefact;

Hypothetical function: models as copy, release, original form, offshoot, simulation or experiment product;

Documentation-visualisation function: models as presentation, figure, illustration, demonstration, explanation, adornment, plastic, structure.

1.3. The Story of the Paper

Models are working instruments which are widely used in Computer Science, Mathematics, Natural and Social Sciences, and the Humanities. For instance, there are many different notions in Business Informatics, e.g., [19]. However, most of them are rather informal statements and cannot be considered to be definitions. This paper aims at providing a definition of the notion of a conceptual model. We start first with a case study and then introduce a general notion of a model in sequel.

2. Models, Models, Models in Computer Science

2.1. The Variety of Models Used in Computer Science

Computer Science uses more than 50 different kinds of models in all of its sub-disciplines [18]. It seems to be difficult to survey all of these approaches. We may, however, classify models based on their orientation. Models are used for understanding, for a description of the general architecture of a system, for investigation, and for construction of systems. Additionally, models are also used as a guidance for the development process of software and hardware systems. Therefore, we may visualise the range of models in Figure 1.

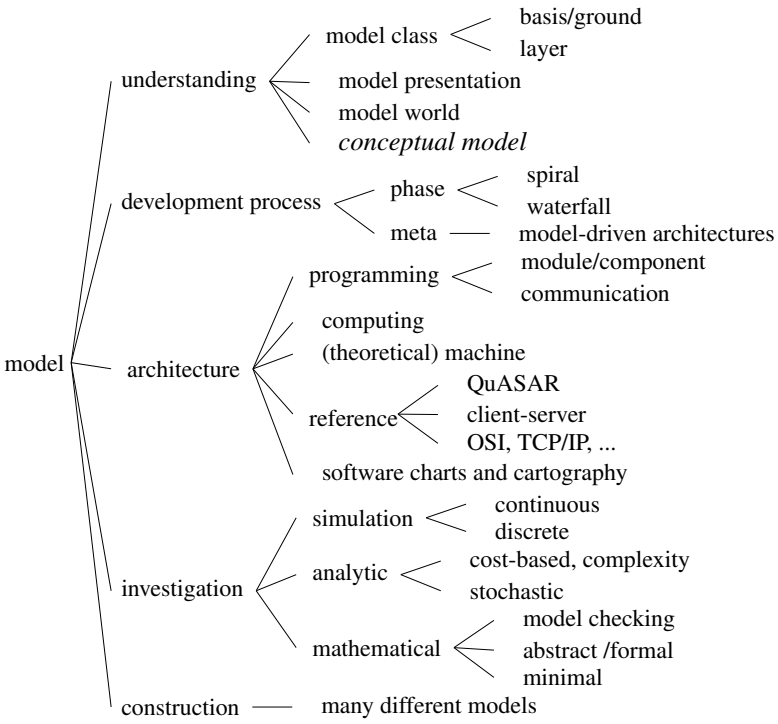


Figure 1. A general classification of models in computer science

The main purpose of models developed and deployed in Computer Science is the construction of systems. Therefore, the function of a model is to give a description of the part of the application world, to use this description as a prescription for requirements, to base on the model the specification of the system, and finally to realise the system [16]. Therefore, we have to distinguish the relevance stage, the modelling stage, and the realisation stage and may associate the model to its functions within these stages. The variety of construction modelling languages is very large. The UML uses more than 140 different kinds of diagrams for different aspects. We can recognise a good number of system modelling languages and kinds of system models. We may concentrate on different aspects of a system such as task models, usage models, object models, component

models, process models, functional models, state models, event models, and information system models. The latter kind of models can also be separated according to a variety of aspects. We thus arrive at an overview in Figure 2.

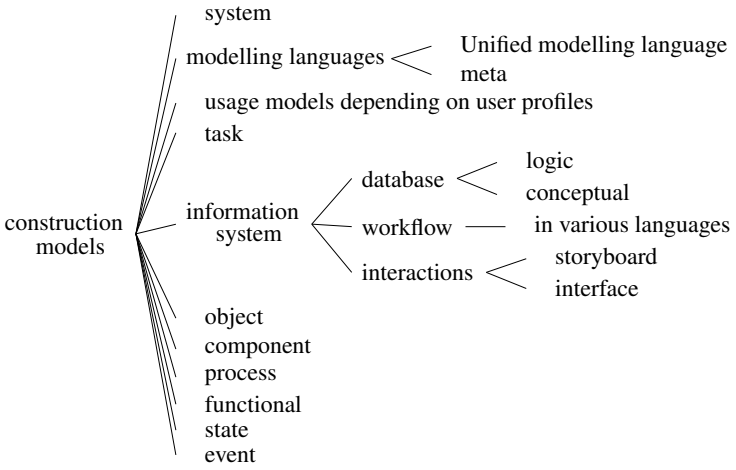
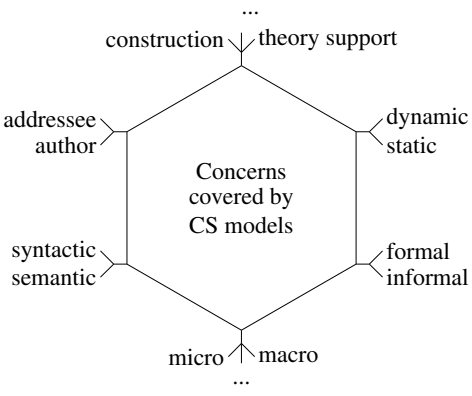


Figure 2. The variety of models used for construction of computer systems

This variety of models in courses of Computer Science [18] has a tendency to confuse. We target however on a general view of the notion of model. We observe that these models are oriented towards their specific purpose. As already observed by [13], the main purposes are: (a) construction of systems, (b) explanation of structure, computation and control of systems, and (c) quality assurance of software and hardware systems. Models should not be used outside their purpose.



The variety of models and conceptual models in Computer Science is overwhelming. But at the same time we observe a number of commonalities. Models are driven by their purpose and thus incorporate thus methods for their language-based development. The main purpose is construction of systems. There are, however, also other purposes of interest, such as theoretical investigation. The community of practice consists typically of authors and addressees. Models my also be distinguished by their aspects such as static or dynamic, by their abstraction level, such as micro or macro, by their level of formality and by their capability, such as syntactic or semantic concerns.

We may learn from these models. There are general rules for (α) combination of models, (β) separation of concern, (γ) model construction, (δ) theoretical foundations etc.

2.2. A Case Study for one of the Models: The Turing Machine

The Turing Machine is one of the most widely used models for computation in Computer Science.
A Turing Machine is given, for instance, as a 7-tuple $(Q, \Sigma, \Gamma, B, q_0, F, \delta)$ with a finite, non-empty set of states Q, a finite, non-empty input alphabet Σ, a finite tape alphabet Γ with $\Gamma \supseteq \Sigma$ and the blank symbol B, a start state $q_0 \in Q$, a set of final states $F \subseteq Q$, and a partial state transformation function $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{R, N, L\}$.

The Turing Machine is often defined using some kind of Von Neuman architecture similar to the one in Figure 3. Instead of a state transformation function we could use a

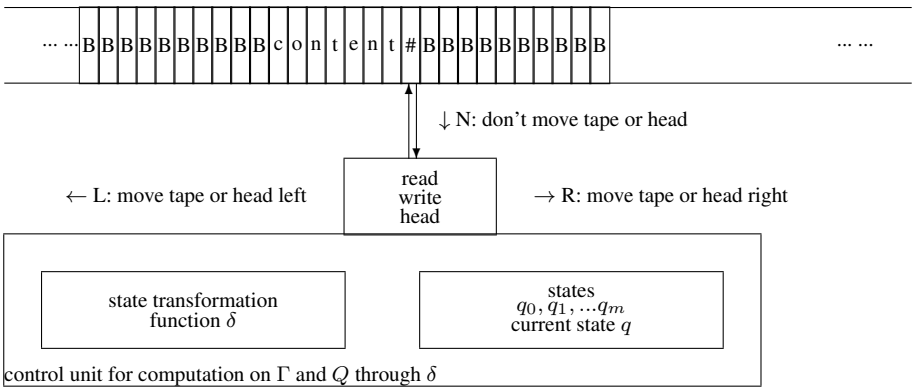


Figure 3. Components of a Turing Machine with a control unit and a storage tape

state transformation relation. Instead of one tape we could use a finite set of tapes. It is however proven that any general Turing Machine can be represented by a machine in the form introduced here.

The second main element of the model of a Turing Machine is its computation process for a Turing Machine. It uses the concept of configuration. A (finite) configuration wqv is given by two words $w, v \in \Gamma^+ \cup \{\lambda\}$ (for the empty word λ) and a state $q \in Q$. It defines that the tape of the Turing Machine is empty (populated by B) except the word wv, that the head is on the first letter of v and that the state of the Turing Machine is q. Given a configuration wqv then the next configuration is determined by δ. A configuration wqv is final if $q \in F$. The result of computation is then v. The initial configuration is $q_0 v_0$.

There is a multiplicity of books which introduces this model for computation. It is commonly believed that any computational process can be described through a computation of Turing Machines.

Turing Machines represent a general notion of algorithm or computer from one side and are a representation of a computation from the other side. The representation of computation serves at least the two purposes in Figure 4.

Turing Machine represent a variety of computational systems such as Von Neuman computers or 1-address machines. They do, however, not represent interactive machines (HCI machines), analog computers or parallel computer systems. We observe, however, four properties: (a) they are well-formed (coherent, consistent, non-ambiguous) in the

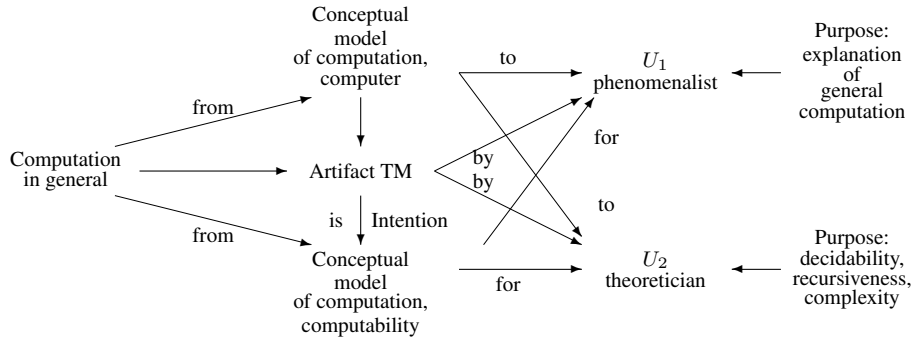


Figure 4. Turing Machines are artifacts that are representations of computations in general for either a conceptual model of computation or a theoretical model of computation with two different purposes for two types U_1, U_2 of users

sense of mathematics; (b) they behave analogously to computers considered; (c) they are far simpler than those computers; (d) they can be used for understanding computation in the sense of Von Neuman, for proof of computability or non-computability, for proof of expressive power of other computation models or other variants of Turing Machines, for consideration of complexity of problems which are computable or decidable. They are therefore *adequate*.

Turing Machines are *justified*: (α) they are corroborated by accepted understanding of state-transforming computations; (β) they are rationally coherent and conform in semiotics given by first-order predicate logics; (γ) they support exclusion of non-intended next world transformations and allow to falsify which kind of transformation is not considered; (δ) they are stable and plastic, e.g., robust since adding tapes or control units, coding of alphabets, restricting to (one-sided) tapes do not change the notion of computable functions. Additionally, they are *sufficient* due to firm quality for other models of computation in general due to their (i) internal quality (e.g., simplicity, minimality, robustness), (ii) external quality (e.g., correctness, comprehensibility, generality), and (iii) quality in use (e.g., parsimony). Due to viability and firm quality Turing Machines are *plausible* devices for computation in general. This judgement of plausibility is based on the evaluation of these machines as precise, rigid and non-tolerant mechanisms without any modality and with full confidence of the computational process.

There are, however, many hidden assumptions, paradigms, postulates, and conventions which are intentionally used but not defined in an explicit form.

Paradigms of computing: we typically assume a Von-Neuman architecture of machines and thus bound computation to determinism and sequentiality and use a separation into storage, computation and control. Computation is based on first-order predicate logics in its canonical form. Results of computation are well-determined and do not depend on context. Computation is performed by next-step transformations (sequential time postulate [2]) and thus uses a special time model.

$\mathrm{Observation}\ (1)$: *Paradigms may impose a number of restrictions to the model.*

Postulates of computing: computations are state transformations (abstract state postulate [3,5]). Each computational step uses a finite set of resources (bounded exploration postulate [3,5]). State transformations are carried out if they do not result in update conflicts (update postulate [3,5]). Coding of the alphabets does not influence computation (isomorphism postulate [2]). The finite result function entirely depends on the input (interaction postulate [2]). Parallelity can be bounded (bounded choice non-determinism postulate [3,5,20]). The alphabets might be defined on infinite resources (background

postulate [20]).

Observation (2): *Postulates restrict the model to a specific form.*

Assumptions and principles for Turing Machines: a machine has exclusive rights on the data space. The abstraction is not changed for different time or steps. The result of a computation step is the final result and cannot be negotiated. The final result is achieved if we reach a final state. The meaning is definable on top of the (first-order predicate) syntax. Input and output are finite. The tape can only be changed at the place of the read/write head. Implicit principles such as compositionality, functional or relational state transformations, step-wise computation, and context-freeness.

Observation (3): *Assumptions and principles might be negotiated or changed.*

Background models, languages and theories and resulting restrictions: Turing Machine computation is based on the sequential time model. Each step is performed sequentially. Languages used are logical ones, functions or relations, and graphs. Background theories are, for instance, logic and canonical set theory. Non-trivial semantical properties of Turing Machines are undecidable (Rice Theorem).

Observation (4): *Background heavily impacts on the model.*

Thought community and thought style: the thought community uses a tape that is infinite, a finite set of states, unchangeability of the transformation function, and a separation into control and storage units. Therefore, these machines cannot be certified (Second Rice Theorem).

Observation (5): *The though approach is an implicit and authority-driven background.*

Profile, goal, purpose and function of the model: the profile of a model is defined through its goal, purpose or function [14,15,17]. The profile of the Turing Machine is oriented on a proof of computability and the corresponding computational complexity. However, it does not consider many aspects, e.g., efficient programs. Construction of real machines is beyond its purpose.

Observation (6): *The model should not be used beyond its profile.*

Concepts and cargo used within the model: the transformation is based on the notion of a function, uses a coded alphabet and recursion. The cargo of the Turing Model is based on the definition of this model as step-wise computation and potential alternatives. It uses time or space complexity measures, movement of either head or tape, termination through reaching a final state, and language and word algebras.

Observation (7): *Concept enrichment depends on the application domain.*

CS approaches do not try to get to the bottom of paradigms, postulates, foundations, theories, assumptions, principles, concepts or cargo, thought community or the background. We must, however, consider this whenever we want to understand and to properly use a model. We call these elements of an artifact *fundament*.

However, unless we do not have *methods* for its utilisation within its deployment, the Turing Machine itself is useless. Most methods used originate from Mathematical Logics and Algebra. Complexity is based on tape or transformation step complexity. These complexity measures allow to use methods from Mathematical Analysis. Therefore, we can prognose the time or space consumed for certain computations. Furthermore, graph theory is used for visualisation through finite labelled graphs. The development of a state transformation function, the integration of machines, the exploration and interpretation is not supported. Turing Machine models are considered to be models that are not to be developed beside the development of a state transformation function. The *deploy-*

ment recount is mainly oriented towards a description of computation and prognosis of computability with or within bounded resources. Construction, definition, exploration, communication, documentation of real computers and other functions of artifacts are not supported by Turing Machines.

2.3. The Informal Notion of a Model

To summarise: a model can be any real or virtual artifact that represents other artifacts. The artifact must, however, be adequate and plausible. Each artifact introduces its fundament. An artifact is deployed as an instrument in science and engineering. This deployment is evolving and might incorporate other artifacts. Therefore, we might use *model suites* [4] instead of a singleton model. This deployment is based on the profile or capacity of the artifact. Artifacts are used in a context by some users with their own intentions, goals and tasks. Therefore, an artifact serves their goals, purposes or functions within the given portfolio.

The example in the previous subsection allows us to gain an insight into the properties of a model. The mapping property cannot be defined through some kind of homomorphism of structure and/or behaviour. It is far more complex. Models are, however, abstractions of some origins. Therefore, any model supports truncation. Models are used within their profile and for the tasks defined for the deployment portfolio. Therefore, the pragmatic property is essential. Models might augment reality. Turing Machines show how amplifications might work. They also have properties of Galilean models. They idealise the computational mechanism. The languages used may be far more strict and rigid than those that may be be used for the origin. Models are used as instruments and bring in some added value. Models serve a purpose based on their profile.

The deployment 'game' of a model is based on an embedding into the story space of the specific discipline. Models such as Turing Machines are neither a good means for construction of an architecture of a computer nor for well-designed programs. They do not allow to reason on information systems beside the general notion of computation. The Turing Machine deployment is a typical explorative function.

3. The Notion of a Model

3.1. The Formal Notion of a Model

Any artifact can be used as a model. It faithfully or dependable represents other artifacts and must provide facilities or features for its use. Based on our observations (1) - (8) we conclude that a model is implicitly based on the *background of a model* consisting of a

basis $\mathcal{B}$ from one side, i.e., basement, paradigms, postulates, restrictions, theories, culture, foundations, conventions, commonsense and
grounding $\mathcal{G}$ from other side, i.e., concepts, foundations, language as carrier, and the cargo,

context $\mathcal{C}$ such as application domain or discipline, school of thought, time, space, granularity and scope, and

community of practice $\mathcal{P}$ with corresponding roles and potentially with specific plays of these roles, and rights, obligations, and the practice commonly accepted within this CoP $\mathcal{P}$.

Given now a community of practice $\mathfrak{P}$ ($\subseteq$ $\mathcal{P}$), a grounding $\mathfrak{G}$ ($\subseteq$ $\mathcal{G}$), bases $\mathfrak{B}$ ($\subseteq$ $\mathcal{B}$), and a context $\mathfrak{C}$ ($\subseteq$ $\mathcal{C}$). Artifacts are explicitly or mostly implicitly given together with some grounding $\mathfrak{G}$, with bases $\mathfrak{B}$ and within some context $\mathfrak{C}$ by a community of practice $\mathfrak{P}$. We call these artifacts $(\mathfrak{G}, \mathfrak{B}, \mathfrak{C}, \mathfrak{P})$ artifacts.

An artifact is called well-formed if it satisfies a well-formedness criterion γ_{form}. An artifact has a profile. The profile is based on the goal or purpose or function of the artifact. If the artifact is devoted to its profile $\mathfrak{D}$ the artifact is called purposeful, given the following measures and thresholds: an *analogy criterion* or an analogy measure $analogy(\mathcal{A}^*; \mathcal{A})$ for artifacts (or collections of artifacts), a analogy threshold $\Theta_{analogy}$; a *complexity measure* for artifacts $complex(\mathcal{M})$ and a simplification threshold $\Theta_{complex}$.

A $(\mathfrak{G}, \mathfrak{B}, \mathfrak{C}, \mathfrak{P})$ artifact $\mathcal{A}^*$ is called $\mathfrak{A}$ *adequate* for a collection of artifacts $\mathfrak{A}$ if

- it is well-formed according to γ_{form} and governed by a $(\mathfrak{C}, \mathfrak{P})$ convention,
- it is analogous to the artifacts $\mathfrak{A}$ to be represented according to some analogy criterion, e.g., $analogy(\mathcal{A}^*, \mathfrak{A}) > \Theta_{analogy}$,
- it is simpler than the artifacts $\mathfrak{A}$ according to some complexity criterion, e.g., $complex(\mathcal{A}^*) < \Theta_{complex} \cdot complex(\mathfrak{A})$, and
- it is purposeful for the profile $\mathfrak{D}$.

An artifact is *justified* by a justification $\mathcal{J}$, i.e. [6], by empirical corroboration (according to purpose, background, etc.) for the representation of the artifacts $\mathfrak{A}$ that is supported by some argument calculus $(\mathrm{BasicArguments}, \Gamma_{\mathrm{Corroboration}})$, by rational coherence and conformity explicitly stated through formulas $(\gamma_{\mathrm{coherence}}, \gamma_{\mathrm{conform}})$, by falsifiability that can be given by a an abductive and/or inductive logical system $(\mathrm{BasicExclusion}, \Gamma_{\mathrm{Falsifiable}})$ (with tests, reduction, parsimony), and by stability and plasticity (depending on the scope, grounding, basis, context and quality) explicitly given through formulas $(\gamma_{\mathrm{stability}}, \gamma_{\mathrm{plasticity}})$. The artifact is *sufficient* by its *quality* characterisation $\mathcal{Q}$ for internal quality, external quality and quality in use or through quality characteristics [14] such as correctness, generality, usefulness, comprehensibility, parsimony, robustness, novelty etc. Justification and sufficiency characterise the signification of an artifact for deployment, reliability and degree of precision efficiency for satisfying the deployment necessities, and extent of coverage depending on deployment. It is typically combined with some assurance evaluation $\mathfrak{E}$ (tolerance, modality, confidence, and restrictions) for $\mathfrak{A}$.

An artifact $\mathcal{A}^*$ is called $(\mathfrak{J}, \mathfrak{Q}, \mathfrak{E})$ *dependable* for some of the justification properties $\mathfrak{J}$ ($\subseteq$ $\mathcal{J}$) and some of the sufficiency characteristics $\mathfrak{Q}$ ($\subseteq$ $\mathcal{Q}$) if

- the quality criteria $\mathfrak{Q}$ are satisfied through $\mathfrak{E}$, and
- it is justified by $\mathfrak{J}$ through $\mathfrak{E}$.

A $(\mathfrak{G}, \mathfrak{B}, \mathfrak{C}, \mathfrak{P})$ artifact $\mathcal{A}^*$ is called **model** of $\mathfrak{A}$ if it is $\mathfrak{A}$ *adequate* and $(\mathfrak{J}, \mathfrak{Q}, \mathfrak{E})$ *dependable*.

The model and the artifact are *functional* if there are methods $\mathfrak{M}_{\mathfrak{D}}$ for utilisation of the artifact in dependence on the profile $\mathfrak{D}$ ($\subseteq$ $\mathcal{D}$) of the artifact. In this case the artifact provides a service. Functional artifacts have their capability and capacity [14]. They are thus deployable. Additionally, development methods $\mathfrak{M}_{\mathfrak{E}}$ for development of artifacts might exist.

Artifacts are used for application cases. These cases are embedded into application stories such as description-prescription, explanation, optimisation-variation, verification-validation-testing, reflection-optimisation, exploration, hypothesis development, documentation-visualisation, or also for substitution. These application stories (or

'deployment games' [22]) are supported by the task portfolio [8] which an artifact might serve. Typical tasks include defining, constructing, exploring, communicating, understanding, replacing, substituting, documenting, negotiating, replacing, reporting, and accounting. We call an artifact and a model *effective* if it can be deployed according to its portfolio.

3.2. Facets of Models

A (*matured*) model is thus an artifact that uses

a fundament with

- the grounding, and
- the (meta-)basis,

four governing directives given by

- the artifacts to be represented by the model,
- the deployment or profile of the model such as goal, purpose or functions,
- the community of practice acting in different roles on certain rights through some obligations, and
- the context of time, discipline, application and scientific school,

two pillars which provide

- methods for development of the model, and
- methods for utilisation of the model,

and finally

the model portfolio and function for the deployment of the model in the given application.

The *model house* in Figure 5 displays these different facets of the model. The house consists of a cellar and a fundament, two pillars, four driving or governing forces, and finally the deployment roof. The *grounding* is typically implicitly assumed. It contains paradigms, the culture in the given application area, the background, foundations and theories in the discipline, postulates, (juristical and other) restrictions, conventions, and the commonsense. The *basis* is the main part of the background. It is typically given for modelling. The development uses a variety of methods for *description*, *construction*, *evolution*, *corroboration*, and *evaluation*. The utilisation is based on methods for *applying*, *prognosis*, *reasoning*, *explanation*, and *exploration*. We have used different verbs for classification of the activities. The model can be used for completion of certain tasks. These tasks may be combined into a *model portfolio*. The model is used for certain functions or deployment scenario (*Gebrauchsspiel*). Finally, the model is *governed* by four directives: *artifacts*, *profile*, *community of practice*, and *context*.

4. Conceptual Models and Conceptional Modelling

4.1. Differences between Conceptual Models and Conceptional Modelling

The words 'conceptual' and 'conceptional' are often considered to be synonyms. R.T. White [21] has already observed that concepts are not the same as conceptions. Concepts can be used in the meaning of classification and as an abstraction of a set of knowledge

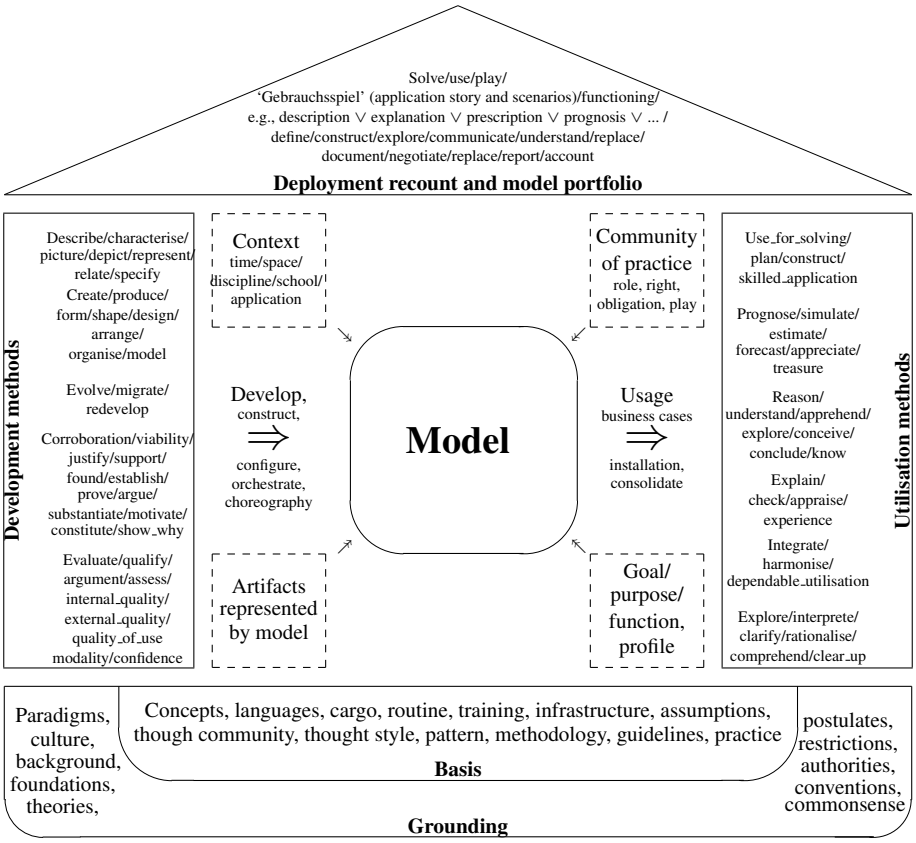


Figure 5. Facets of the model with grounding and basis as the fundament, with four governing directives, with technical and technological pillars for development and utilisation, and with the application roof

a person associates with the concept's name. However, conceptions are systems of explanation. Conceptions are thus far more complex and difficult to define than the either meanings of the concept.

The word 'conceptual' is linked to concepts and conceptions. Conceptual means that a thing, e.g. artifact is characterised by concepts or their conceptions. The word 'conceptional' associates a thing as being or being of the nature of a notion or concept. Therefore, we distinguish the 'conceptual model' from 'conceptional modelling'. Conceptional modelling is modelling with associations to concepts. A conceptual model incorporates concepts into the model.

4.2. Conceptional Modelling: Modelling Enhanced by Concepts

An information systems model is typically a schematic description of a system, theory, or phenomenon of an origin that accounts for known or inferred properties of the origin and may be used for further study of characteristics of the origin. *Conceptional modelling* aims to create an abstract representation of the situation under investigation, or more precisely, the way users think about it. Conceptual models enhance models with

concepts that are commonly shared within a community or at least between the stakeholders involved in the modelling process. A general definition of concepts is given in [15]. Concepts specify our knowledge what things are there and what properties things have. Concepts are used in everyday life as a communication vehicle and as a reasoning chunk. Concept definition can be given in a narrative informal form, in a formal way, by reference to some other definitions, etc. We may use a large variety of semantics [11], e.g., lexical or ontological, logical, or reflective.

Conceptualisation aims at collection of objects, concepts and other entities that are assumed to exist in some area of interest, and the relationships that exist amongst them. It is thus an abstract, simplified view or description of the world that we wish to represent.

4.3. Conceptualisation of Models

Conceptualisation extends the model by a number of concepts that are the basis for an understanding of the model and for the explanation of the model to the user. A general theory of concepts that has been used for conceptualisation was introduced by[15]. Concepts are used in everyday life as a communication vehicle and as a reasoning chunk.

Our revision of the design science and the conceptual modelling frameworks in [4] shows that concept enhancement is an orthogonal activity. It can be partially completed without having a negative impact on the realisation phase if stakeholders (i.e., the community of practice) involved have a common understanding of the model and its properties, and a commonsense about the objectives for realisation and a good mapping facility. Therefore, conceptualisation may also be implicit and may use some kind of lexical semantics, e.g. word semantics, within a commonly agreed name space.

5. Conclusion

In this paper we introduced a formal definition of the notion of a conceptual model. This notion has been applied and tested in Computer Science, in Philosophy, in Physics, and other sciences. As far as we discovered the notion is sufficient.

We have been aiming at development of a *formal* notion of a model. Such formal notion is necessary whenever we need a theory of conceptional modelling. It allows to exclude artifacts to become a model outside the judgement frame.

References

[1] J. Agassi. Why there is no theory of models. In I. Niiniluoto W.E. Herfel, W. Krajewsky and R. Wojcicki, editors, *Theories and Models in Scientific Processes*, pages 17–26, Amsterdam-Atlanta, 1995.

[2] Andreas Blass, Yuri Gurevich, Dean Rosenzweig, and Benjamin Rossman. Interactive small-step algorithms I: Axiomatization. *LMCS*, 3(4:3):1–29, 2007.

[3] E. Börger and R. Stärk. *Abstract state machines - A method for high-level system design and analysis*. Springer, Berlin, 2003.

[4] A. Dahanayake and B. Thalheim. Enriching conceptual modelling practices through design science. In *BMMDS/EMMSAD*, volume 81 of *Lecture Notes in Business Information Processing*, pages 497–510. Springer, 2011.

[5] Y. Gurevich. Sequential abstract-state machines capture sequential algorithms. *ACM TOCL*, 1(1):77–111, 2000.

[6] I.A. Halloun. *Modeling Theory in Science Education*. Springer, Berlin, 2006.
[7] R. Kaschek. *Konzeptionelle Modellierung*. PhD thesis, University Klagenfurt, 2003. Habilitationsschrift.
[8] R. Kaschek. Challenges to information systems development. Position paper given at. SIGPrag 2008 inauguration meeting (Paris), December 2008. SIGPrag 2008: ICIS 2008 affiliated workshop.
[9] B. Mahr. Information science and the logic of models. *Software and System Modeling*, 8(3):365–383, 2009.
[10] J.E. Safra, I. Yeshua, and et. al. *Encyclopædia Britannica*. Merriam-Webster, 2003.
[11] K.-D. Schewe and B. Thalheim. Semantics in data and knowledge bases. In *SDKB 2008*, LNCS 4925, pages 1–25, Berlin, 2008. Springer.
[12] H. Stachowiak. Modell. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon zur Wissenschaftstheorie*, pages 219–222. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.
[13] W. Steinmüller. *Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993.
[14] B. Thalheim. Towards a theory of conceptual modelling. *Journal of Universal Computer Science*, 16(20):3102–3137, 2010. `http://www.jucs.org/jucs_16_20/towards_a_theory_of`.
[15] B. Thalheim. The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 17, pages 547–580. Springer, Berlin, 2011.
[16] B. Thalheim. The art of conceptual modelling. In *Information Modelling and Knowledge Bases XXII*, volume 237 of *Frontiers in Artificial Intelligence and Applications*, pages 149–168. IOS Press, 2012.
[17] B. Thalheim. The science and art of conceptual modelling. In A. Hameurlain et al., editor, *TLDKS VI*, number 7600 in LNCS, pages 76–105. Springer, Heidelberg, 2012.
[18] M. Thomas. Modelle in der Fachsprache der Informatik. Untersuchung von Vorlesungsskripten aus der Kerninformatik. In *DDI*, volume 22 of *LNI*, pages 99–108. GI, 2002.
[19] O. Thomas. Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation. Technical Report Heft 184, Institut für Wirtschaftsinformatik, DFKI, Saarbrücken, Mai 2005.
[20] Qing Wang. *Logical Foundations of Database Transformations for Complex-Value Databases*. Logos Verlag Berlin, Germany, 2010.
[21] R.T. White. Commentary: Conceptual and conceptional change. *Learning and instruction*, 4:117–121, 1994.
[22] L. Wittgenstein. *Philosophical Investigations*. Basil Blackwell, Oxford, 1958.

Acknowledgment. I would like to thank my colleagues at the Christian-Albrechts University Kiel for the fruitful discussions on many facets of models within the context of the faculties and institutes of Archeology, Arts, Biology, Chemistry, Computer Science, Economics, Electrotechnics, Environmental Sciences, Farming and Agriculture, Geosciences, Historical Sciences, Humanities, Languages and Semiotics, Mathematics, Medicine, Ocean Sciences, Pedagogical Science, Philosophy, Physics, Political Sciences, Sociology, and Sport Science. We are thankful to the International Institute of Theoretical Cardiology (IIfTC) for the evaluation of our approach. These discussions lasted in weekly 'Tuesday' open-end evening seminars from 2009 until 2012 and in monthly seminars at the IIfTC. They resulted in a general understanding of the notion of a model in most sciences, engineering and technology.

Das Modell des Modelles

Bernhard Thalheim

October 5, 2014

Vorbemerkung

((1)) Bernd Mahr endet die Arbeit [4]((48)) mit einem *Forschungsauftrag* zur Entwicklung einer aussagenkräftigen allgemeinen Theorie der Modelle. Modelle sind mittlerweile zum "inter-galactic" Kommunikationsmittel von Wissenschaftlern geworden. Wir müssen deshalb den Forschungsauftrag erweitern zur Entwicklung eines Modelles eines Modelles.

((2)) Modelle werden in allen Wissenschaften verwendet. Sie sind Werkzeuge bzw. Arbeits*instrumente*. Ihre Verwendung folgt einem Ziel, meist auch einem *Zweck* (d.h. einem Ziel, wobei auch Methoden zum Erreichen eines Zieles bereitgestellt sind). Mit dem Zweck beantworten wir die rhetorischen Fragen: warum, wozu, womit, wobei, wofür, mit welchem Mehrwert. Sie haben als Instrumente in entsprechenden *Nutzungsszenarien* Funktionen wie z.B. eine Beschreibungs- und Vorbildfunktion für die Entwicklung eines (Informatik-)Systemes, eine Erklärungsfunktion für Phänomene, eine Optimierungs- und Variationsfunktion für existierende Lösungen, eine Stellvertreterfunktion für die Substitution von Systemen oder eine Dokumentations- und Visualisierungsfunktion für komplexe Systeme oder auch Systeme mathematischer Formeln.

((3)) Der Gebrauch von Modellen in entsprechenden Funktionen führt zu einer Vielfalt von unterschiedlichen Auffassungen, welches Instrument Modell ist. Wir kennen in den Wissenschaften u.a. Situationsmodelle, Perzeptionsmodelle, Realmodelle, Erklärungsmodelle, Experimentmodelle, formale Modelle, mathematische Modelle, Simulationsmodelle, Emulationsmodelle, Ersetzungsmodelle, Erklärungsmodelle und Repräsentationsmodelle [2]. Jedes dieser Modelle hat eine spezifische Form, eine spezifische Funktion und auch spezifische Zwecke.

Eingrenzung

((4)) Ein allgemeiner Modellbegriff wurde in einer Vielzahl von Arbeiten - moderne Arbeiten folgen meist den Ansätzen von H. Stachowiak [5, 1] - seit der En-

twicklung der Wissenschaften und der Technologie gesucht. Der hier vorzustellende Modellbegriff stellt eine Ergänzung des Modellbegriffes nach B. Mahr [4] ((24))-((29)) und folgt dem Modell der Auffassung [4]((29))-((30)) Er basiert auf einem Kompendium der Modellbegriffe [8] in den Agrar- und Ernährungswissenschaften, der Altertumskunde, der Informatik, den Ingenieurwissenschaften, den Lebenswissenschaften, der Mathematik, der Medizin, den Meereswissenschaften, der Ökologie, der Ökonomie, der Pädagogik, der Philologie, der Philosophie, der Physik, den Sportwissenschaften und den Werkstoffwissenschaften. Der im folgenden vorgestellte Modellbegriff kombiniert und verallgemeinert diese Ansätze. Er ist ausreichend für diese disziplinären Modellbegriffe.

Ein Modellbegriff

((5)) Ein Modell ist ein Instrument, das *adäquat* und *verläßlich* andere Originale *repräsentiert* je nach beabsichtigtem *Zweck* (oder Funktion in der Nutzung dieses Instrumentes oder auch damit verbundenem Ziel)[6, 7]. Das Modellsein hängt von der Nutzergemeinschaft und dem Kontext ab.

Instrumente bringen ihren eigenen *Inhalt* mit. Der Inhalt erlaubt eine epistemische Einordnung des Instrumentes in den Gebrauchskontext. Der Inhalt basiert auf einem *Hintergrund* des Instrumentes, wird innerhalb einer *Nutzergemeinschaft* akzeptiert und ist innerhalb eines *Kontextes* gültig. Der Inhalt wird im *Cargo* zusammengefaßt.

Zweckmäßige Instrumente sollen für den Zweck tauglich sein, d.h. ihnen sind Methoden zur Nutzung und zur Entwicklung zugeordnet.

Instrumente sollen als Modelle in Nutzungsszenario effektiv eingesetzt. Solche Instrumente sind dann *effektive* Modelle.

((6)) Der Inhalt eines Instrumentes wird explizit angegeben, bestimmt den Nutzen und den Mehrwert, den das Instrument als Modell beibringt. Der Hintergrund eines Instrumentes besteht zum einem (i) aus Paradigmen, Postulaten, Theorien, der Fundierung, Prinzipien, der Kultur, den Autoritäten, den Beschränkungen und dem commonsense (gesunder Menschenverstand) und zum anderen (ii) den Annahmen, dem Denkstil und dem Denkkollektiv, Konzepten und Konzeptionen, Konventionen, Sprachen, Methodiken, den gängigen and akzeptierten Praktiken inklusive ‘good practices’ und Anleitungen. Man kann sich oft auf wohldefinierte oder wohlgeformte Instrumente einschränken.

((7)) Ein (wohldefiniertes) Instrument heißt *adäquat* für die Repräsentation von Originalen, wenn es zum einem analog aufgrund eines Analogiemaßes, zum zweiten fokussierter aufgrund eines Komplexitätsmaßes und zielgerecht für entsprechende angegebene Ziele ist.

((8)) Ein (wohldefiniertes) Instrument wird *verläßlich* genannt, falls es gerechtfertigt mit einer Rechtfertigung ist und eine befriedigende Qualität anhand eines Qualitätspasses besitzt. Verläßlich- und Zuverläßigkeit haben mindestens zwei Aspekte: eine Nachvollziehbarkeit der Struktur und Funktionen des Instrumentes zum einem und eine ausreichende Qualität zum anderen. Eine Rechtfertigung umfaßt eine (empirische) Begründung, Aussagen zur Kohärenz und Konformität, eine Validierung gegen die Originale und Aussagen zur Nachhaltigkeit und zur Repräsentation.

Der Cargo

((9)) Der Cargo [3] stellt einen Abstrakt für ein Modell dar, mit dem das Modell eingeführt werden kann. Instrumente sind aufgrund ihrer Adäquatheit, ihrer Rechtfertigung und ihres Qualitätspasses Träger eines *Cargo*. Sie werden von einer Nutzergemeinschaft bewußt oder unbewußt innerhalb ihres Kontextes als Modell *aufgefaßt* [4]((31)) aufgrund von entsprechenden Urteilen, von subjektiven Vorstellungen und einer Qualifikation.

((10)) Ein Instrument, das als Modell genutzt wird, ist mit einem Cargo beladen. Die Beladung umfaßt eine Darstellung der *Bestimmung* des Instrumentes als Modell, des *Hauptgehaltes* des Modelles je nach *Gebrauchsmodell*, der *Bedeutung* des Modelles innerhalb eines Bedeutungsspielraumes und der *Identität* des Instrumentes als Modell. Der Cargo eines Modelles inkludiert auch die Darstellung der Kapazität und des Potentials eines Modelles.

References

[1] R. Frigg and S. Hartmann. Models in science. http://plato.stanford.edu/ entries/models-science/, retrieved Sept. 21, 2014. First published Mon Feb 27, 2006; substantive revision Mon Jun 25, 2012.

[2] G. Greefrath, G. Kaiser, W. Blum, and R. Borromeo Ferri. *Mathematisches Modellieren für Schule und Hochschule*, chapter Mathematisches Modellieren - Eine Einführung in theoretische und didaktische Hintergründe, pages 11–37. Springer, 2013.

[3] B. Mahr. Cargo. Zum Verhältnis von Bild und Modell. In I. Reichle, S. Siegel, and A. Spelten, editors, *Die Wirklichkeit visueller Modelle*, pages 17–40. Wilhelm Fink Verlag, München, 2008.

[4] B. Mahr. Modelle und ihre Befragbarkeit - Grundlagen einer allgemeinen Modelltheorie. *Erwägen-Wissen-Ethik*, forthcoming, 2015.

[5] H. Stachowiak. *Allgemeine Modelltheorie*. Springer, 1973.

[6] B. Thalheim. The science and art of conceptual modelling. In A. Hameurlain et al., editor, *TLDKS VI*, number 7600 in LNCS, pages 76–105. Springer, Heidelberg, 2012.

[7] B. Thalheim. The conceptual model $\equiv$ an adequate and dependable artifact enhanced by concepts. In *Information Modelling and Knowledge Bases*, volume XXV of *Frontiers in Artificial Intelligence and Applications, 260*, pages 241–254. IOS Press, 2014.

[8] B. Thalheim and I. Nissen, editors. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*. De Gryuter, Boston, 2015.

Bernhard Thalheim
Institut für Informatik
Christian-Albrechts-Universität zu Kiel, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

27 The Notion of a Model

Definition: A model is a well-formed, adequate, and dependable instrument that represents origins. Its criteria of well-formedness, adequacy, and dependability must be commonly accepted by its community of practice within some context and correspond to the functions that a model fulfills in utilisation scenarios and use spectra. As an instrument, a model is grounded in its community's sub-discipline and is based on elements chosen from the sub-discipline.

27.1 The Conception of the Model

Science and technology widely uses models in a variety of in utilisation scenarios. Models function as an instrument in some utilization scenarios and a use spectrum. Their function in these scenarios is a combination of functions such as explanation, optimization-variation, validation-verification-testing, reflection-optimization, exploration, hypothetical investigation, documentation-visualization, and description-prescription as a mediator between a reality and an abstract reality that developers of a system intend to build. The model functions determine the *purposes* of the deployment of the model.

Models have several *essential properties* that qualify an instrument as a model (Tha12; Tha14):

- An instrument is *well-formed* if it satisfies a well-formedness criterion.
- A well-formed instrument is *adequate* for a collection of origins if (i) it is analogous to the origins to be represented according to some analogy criterion, (ii) it is more focused (e.g. simpler, truncated, more abstract or reduced) than the origins being modelled, and (iii) it sufficient to satisfy its purpose.
- Well-formedness enables an instrument to be *justified*: (i) by an empirical corroboration according to its objectives, supported by some argument calculus, (ii) by rational coherence and conformity explicitly stated through formulas, (iii) by falsifiability that can be given by an abductive or inductive logic, and (iv) by stability and plasticity explicitly given through formulas.
- The instrument is *sufficient* by a *quality* characterisation for internal quality, external quality and quality in use or through quality characteristics (Tha10) such as correctness, generality, usefulness, comprehensibility, parsimony, robustness, novelty etc. Sufficiency is typically combined with some assurance evaluation (tolerance, modality, confidence, and restrictions).

- A well-formed instrument is called *dependable* if it is sufficient and is justified for some of the justification properties and some of the sufficiency characteristics.
- An instrument is called **model** if it is *adequate* and *dependable*. The adequacy and dependability of an instrument is based on a *judgement* made by the community of practice.
- An instrument has a *background* consisting of an undisputable grounding from one side (paradigms, postulates, restrictions, theories, culture, foundations, conventions, authorities) and of a disputable and adjustable basis from other side (assumptions, concepts, practices, language as carrier, thought community and thought style, methodology, pattern, routines, commonsense).
- A model is used in a *context* such as discipline, a time, an infrastructure, and an application.

Not only should a model faithfully represent a collection of origins by being well-formed, adequate, and dependable, it should also provide facilities or methods for its use. A model is *functional* if there are methods for utilization of the instrument to achieve the objectives for which an instrument might serve. Typical task objectives include defining, constructing, exploring, communicating, understanding, replacing, substituting, documenting, negotiating, replacing, optimizing, validating, verifying, testing, reporting, and accounting. We call a model *effective* if it can be deployed according to its objectives.

27.2 Properties of Models

Models satisfy several properties that make them functional and effective (Mah08; Mah15; Sta73; Tha10; Tha11; Tha12; Tha14):

(1) *Mapping* property: the model has an origin and can be based on a mapping from the origin to the instrument.
(1') *Analogy* property: the model is analogous to the origins based on some analogy criterion.
(2) *Truncation* (*reduction*) property: the model lacks some of the ascriptions made to the origin and thus functions as an Aristotelian model by abstraction by disregarding the irrelevant.
(3) *Pragmatic* property: the model use is only justified for particular model users, the tools of investigation, and the period of time.
(4) *Amplification* property: models use specific extensions which are not observed in the original.

(5) *Idealisation* property: modelling abstracts from reality by scoping the model to the ideal state of affairs.
(6) *Carrier* (*cargo*) property: models reflect a conception on origins based on the capacity of a language and are filled with anticipation. They carry a cargo(Mah08).
(6') *Utilisation* property: the model functions well within its intended scenarios of usage according to its capacity and potential.
(7) *Divergence* property: models (e.g. Galilean models) are developed for improving divergence, deviation, discrepancy the physical world or for inclusion of visions of better reality, e.g. for construction via transformation.
(8) *Added value* property: models provide a value or benefit based on their utility, capability and quality characteristics.
(9) *Purpose* property: models are governed by the purpose. The model preserves the purpose.

Literatur

[Mah08] B. Mahr. Cargo. Zum Verhältnis vo n Bild und Modell. In I. Reichle, S. Siegel, and A. Spelten, editors, *Die Wirklichkeit visueller Modelle*, pages 17–40. Wilhelm Fink Verlag, München, 2008.

[Mah15] B. Mahr. Modelle und ihre Befragbarkeit - Grundlagen einer allgemeinen Modelltheorie. *Erwägen-Wissen-Ethik*, forthcoming, 2015.

[Sta73] H. Stachowiak. *Allgemeine Modelltheorie*. Springer, 1973.

[Tha10] B. Thalheim. Towards a theory of conceptual modelling. *Journal of Universal Computer Science*, 16(20):3102–3137, 2010. http://www.jucs.org/jucs_16_20/towards_a_theory_of.

[Tha11] B. Thalheim. The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 17, pages 547–580. Springer, Berlin, 2011.

[Tha12] B. Thalheim. The science and art of conceptual modelling. In A. Hameurlain et al., editor, *TLDKS VI*, number 7600 in LNCS, pages 76–105. Springer, Heidelberg, 2012.

[Tha14] B. Thalheim. The conceptual model ≡ an adequate and dependable artifact enhanced by concepts. In *Information Modelling and Knowledge Bases*, volume XXV of *Frontiers in Artificial Intelligence and Applications, 260*, pages 241–254. IOS Press, 2014.

Enhancing Entity-Relationship Schemata for Conceptual Database Structure Models

Bernhard Thalheim and Marina Tropmann-Frick

Department of Computer Science, Christian-Albrechts-University Kiel, 24098 Kiel, Germany

Abstract. The paper aims at development of well-founded notions of database structure models that are specified in entity-relationship modelling languages. These notions reflect the functions a model fulfills in utilisation scenarios.

1 Utilisation Scenarios of Conceptual Models

Conceptual models are used as an artifact in many utilisation scenarios. Design science research [4] and ER schema development methodologies (e.g. [3, 8, 11]) developed so far a good number of such scenarios.

Communication and negotiation scenario: The conceptual model is used for exchange of meanings through a common understanding of notations, signs and symbols within an application area. It can also be used in a back-and-forth process in which interested parties with different interests find a way to reconcile or compromise to come up with an agreement. The schema provides negotiable and debatable propositions about the understanding of the part of the reality but does not have well-developed justificatory explanations.

Conceptualisation scenario: The main application area for extended entity-relationship models is the conceptualisation of database applications. Conceptualisation is typically shuffled with discovery of phenomena of interest, analysis of main constructs and focus on relevant aspects within the application area. The specification incorporates concepts injected from the application domain.

Description scenario: In a description scenario, the model provides a specification how the part of the reality that is of interest is perceived and in which way augmentations of current reality are targeted. The model says what the structure of an envisioned database is and what it will be.

Prescription scenario: The conceptual model is used as a blueprint for or prescription of a database application, especially for prescribing the structures and constraints in such applications. The schema proposes what the structure of a database is on the one hand and how and where to construct the realisation on the other hand. ER schemata can be translated to relational, XML or other schemata based on transformation profiles [11] that incorporate properties of the target systems.

These scenarios are typically bundled into *use spectra*. For instance, design science uses three cycles: the relevance cycle based on the design cycle based on description and communication and negotiation scenario, and the rigor cycle based on a knowledge discovery and experience propagation scenario. Database development is mainly based on description, conceptualisation, and construction scenarios. The re-engineering and system maintenance use spectrum is based on combination of documentation scenarios with an explanation and discovery scenario from one side and communication and negotiation scenario from the other side. Models are also used for documentation scenarios, explanation and discovery scenarios for applications or systems, and for knowledge experience scenario. We concentrate here on the four scenarios.

Contribution of the Paper

The first main contribution of this paper is an analysis whether an entity-relationship schema suffices as a model for database structures. We realise that the four scenarios require additional elements for the ER schema in order to become a model. The second main contribution of this paper is a proposal for an enhancement of ER schemata which allows to consider the artifact as a model within the given four scenarios. The paper partially presupposes our research (esp. [15], see also other papers in [16]).

2 The General Notion of a Model

Science and technology widely use models in a variety of utilisation scenarios. Models function as an artifact in some utilization scenario. Their function in these scenarios is a combination of functions such as explanation, optimization-variation, validation-verification-testing, reflection-optimization, exploration, hypothetical investigation, documentation-visualization, and description-prescription as a mediator between a reality and an abstract reality that developers of a system intend to build. The model functions determine the *purposes* of the deployment of the model.

The following notion of the model has been developed [17] after an intensive discussion in workshops with researchers from disciplines such as Archeology, Arts, Biology, Chemistry, Computer Science, Economics, Electrotechnics, Environmental Sciences, Farming and Agriculture, Geosciences, Historical Sciences, Humanities, Languages and Semiotics, Mathematics, Medicine, Ocean Sciences, Pedagogical Science, Philosophy, Physics, Political Sciences, Sociology, and Sport Science.

Definition 1. *A model is a well-formed, adequate, and dependable artifact that represents origins. Its criteria of well-formedness, adequacy, and dependability must be commonly accepted by its community of practice within some context and correspond to the functions that a model fulfills in utilisation scenarios. As an artifact, a model is grounded in its community's sub-discipline and is based on elements chosen from the sub-discipline.*

This notion has been tested against the notions of a model that are typically used in these disciplines. We could state that these notions are covered by our notion. Origins of a model [7] are artifacts the model reflects. Adequacy of models has often been discussed, e.g. [6, 9, 10]. Dependability is only partially covered in research, e.g. [5].

Models have several *essential properties* that qualify an artifact as a model [15, 16]:

- An artifact is *well-formed* if it satisfies a well-formedness criterion.
- A well-formed artifact is *adequate* for a collection of origins if (i) it is analogous to the origins to be represented according to some analogy criterion, (ii) it is more focused (e.g. simpler, truncated, more abstract or reduced) than the origins being modelled, and (iii) it sufficient satisfies its purpose.
- Well-formedness enables an artifact to be *justified*: (i) by an empirical corroboration according to its objectives, supported by some argument calculus, (ii) by rational coherence and conformity explicitly stated through formulas, (iii) by falsifiability that can be given by an abductive or inductive logic, and (iv) by stability and plasticity explicitly given through formulas.
- The artifact is *sufficient* by its *quality* characterisation for internal quality, external quality and quality in use or through quality characteristics [13] such as correctness, generality, usefulness, comprehensibility, parsimony, robustness, novelty etc. Sufficiency is typically combined with some assurance evaluation (tolerance, modality, confidence, and restrictions).
- A well-formed artifact is called *dependable* if it is sufficient and is justified for some of the justification properties and some of the sufficiency characteristics.
- An artifact is called **model** if it is *adequate* and *dependable*. The adequacy and dependability of an artifact is based on a *judgement* made by the community of practice.
- An artifact has a *background* consisting of an undisputable grounding from one side (paradigms, postulates, restrictions, theories, culture, foundations, conventions, authorities) and of a disputable and adjustable basis from other side (assumptions, concepts, practices, language as carrier, thought community and thought style, methodology, pattern, routines, commonsense).
- A model is used in a *context* such as discipline, a time, an infrastructure, and an application.

The Taxonomy of Conceptual Models

The starting point in our investigation was the observation that there is no unique and commonly agreeable notion of the conceptual database structure model as such. The model supports different purposes and has different functions in utilisation scenarios. Therefore, we must have different notion of the conceptual database structure model.

The conceptual model functions within the utilisation scenarios in different roles with different rigidity, modality and confidence. Models are used as *perception* models (reflection of one partys current understanding of world; for

understanding the application domain), *situation* models (reflection of a given state of affairs), *conceptual* models (based on formal concepts and conceptions) *experimentation* models (as a guideline and basis for experimentation), *formal* models (based some formalism within a well-based formal language), *mathematical* models (in the language of mathematics), *computational* models (based on some (semi-)algorithm), *physical* models (as physical artifact), *visualisation* models (for representation using some visualisation), *representation* models (for representation of some other notion), *diagrammatic* models (using a specific language, e.g. UML), *exploration* models (for property discovery), and *heuristic* models (based on some Fuzzyness, probability, plausibility, correlation), etc. The large variety of known and used model notions (e.g. see [14, 15, 17] and the collection [16]) mainly reflects these different kinds. The situation model might use a rigorous structured English (OMG proposal) and represents the nature of the business within the language of the business. In this case it is also called *reality* model .

Due to space limitations we concentrate here on the four utilisation scenarios described in section 1. The other scenarios, such as documentation scenario, explanation and discovery scenario for applications, explanation and discovery scenario for systems and knowledge discovery and experience propagation scenario, are supported by specific conceptual models in a similar form.

3 Conceptual Database Structure Models for Communication and Negotiation

Communication aims at exchange of meanings among interested parties. The model is used as a means for communication. It truly represents some aspects of the real world. It enables clearer communication and negotiation about those aspects of the real world. It has therefore potentially several meanings in dependence on the parties. Communication acts essentially follow rhetoric frames[1], i.e. they are characterised through "who says what, when, where, why, in what way, by what means" (Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis). In our case, the model ("what") incorporates the meaning of parties (semantical space; "who") during a discourse ('when') within some application with some purpose ("why") based on some modelling language.

Typically, artifacts used for communication and negotiation follow additional principles: Viewpoints and specific semantics of users are explicitly given. The artifact is completely logically independent from the platform for realisation. The name space is rather flexible. The model is functioning and effective if methods for reasoning, understanding, presentation, exploration, explanation, validation, appraisal and experimenting are attached.

Conceptual model for communication: *The conceptual database structure model comprises the database schema, reflects viewpoints and perspectives of different involved parties $\mathcal{U}$ and their perception models, and implicitly links to (namespaces or) concept fields of parties. Adequacy and dependability are based on the*

[1] It relates back to Hermagoras of Temnos or Cicero more than 2000 years ago.

association of the perception models to viewpoints and of the viewpoints with the schema..

A partial communication model does not use a schema and does not associate viewpoints to schema elements.

Therefore, the model can be formally defined as a quintuple

$$(\mathcal{S},\, \{\,(\mathcal{V}_i, \Phi_i)\,|\,i \in \mathcal{U}\},\ \{\,(\mathcal{P}_i, \Psi_i)\,|\,i \in \mathcal{U}\},\, \mathcal{A},\, \mathcal{D})$$

that relates elements of the conceptual schema $\mathcal{S}$ to the perception model $\mathcal{P}_i$ of the given party i. The perception model is reflected in the schema via viewpoints $\mathcal{V}_i$. It implicitly uses concept fields $\mathfrak{C}_i$ of parties i. The mapping $\Psi_i : \mathcal{P}_i \rightarrow \mathcal{V}_i$ associates the perception model of a given party i to the agreed viewpoint. In the global-as-design approach, the viewpoint $\mathcal{V}_i$ is definable by some constructor Φ_i defined on $\mathcal{S}$. The adequacy $\mathcal{A}$ is directly given by the second and third parts of the model. The justification $\mathcal{J}$ and the dependability $\mathcal{D}$ are extracted from the properties of Φ_i and Ψ_i.

The negotiation scenario can thus be understood as stepwise construction of the mappings, stepwise revision of the schema and the viewpoints, and analysis whether the schema represents the corresponding perception model.

4 Conceptual Database Structure Models for Conceptualisation

Conceptualisation is based on one or more concept or conception spaces of business users. Given a business user community $\mathcal{U}$ with their specific concept fields $\{\mathfrak{C}_i |\, i \in \mathcal{U}\}$. Let us assume that the concept fields can be harmonised or at least partially integrated into a common concept field of users $\mathfrak{C}^{\mathcal{U}}$ similar to construction approaches used for ontologies.

Conceptual model for conceptualisation: *The conceptual database structure model comprises the database schema, a collection of views for support of business users, and a mapping for schema elements that associates these elements to the common concept field.*.

Therefore, the model can be formally defined as a quintuple

$$(\mathcal{S},\, \mathfrak{V},\, \mathcal{M},\, \mathcal{A},\, \mathcal{D})$$

consisting of the conceptual schema $\mathcal{S}$ and a mapping $\mathcal{M} : \mathcal{S} \rightarrow \mathfrak{C}^{\mathcal{U}}$. The adequacy $\mathcal{A}$ is based on the mapping. The justification $\mathcal{J}$ and the dependability $\mathcal{D}$ are derived from the concept fields.

5 Conceptual Database Structure Models for Description and Prescription

An artifact that is used as a conceptual model for database system description can be either understood as a representation, refinement and amplification [13, 16] of situation or reality models or as a refinement and extension of the

communication model. The main approach to conceptual modelling for system construction follows the first option. The second option would however be more effective but requires a harmonisation of the perception models. The first option may start with reality models that reflect the nature of the business in terms and in the language of the business. It includes also the top management view, a corporate overview, and a sketch of the environment. The reality models are reasonable complete, are described in terms of the business and use general categories that are convergent.

Conceptual model for description: *The conceptual database structure model comprises the database schema, a collection of views for support of business users, a collection of a commonly accepted reality models that reflects perception or situation models with explicit association to views, and the declaration of model adequacy and dependability..*

Therefore, the model can be formally defined as a quintuple

$$(\mathcal{S},\, \mathfrak{V},\, (\mathfrak{R}, \Psi),\, \mathcal{A},\, \mathcal{D})\ .$$

The conceptual model may be enhanced by an association Φ of views to the schema. This enhancement is however optional.

Descriptive models adequately explicate main concepts [12] from the reality models and combine them into views. The descriptive model reflects the origins and abstracts from reality by scoping the model to the ideal state of affairs.

Prescriptive models that are used for system construction are filled with anticipation of the envisioned system. They deliberately diverge from reality in order to simplify salient properties of interest, transforming them into artifacts that are easier to work with. They may follow also additional paradigms and assumption beyond the classical background of conceptual database structure models: Salami slicing of the schema by rigid separation of concern for all types; conformance to methods for simple (homomorphic) transformation; adequateness for direct incorporation; hierarchical architecture within the schema, e.g. for specialisation and generalisation of types; partial separation of syntax and semantics; tools with well-defined semantics; viewpoint derivation; componentisation and modularisation; integrity constraint formulation support; conformance to methods for integration; variations for the same schema for more flexible realisation etc.

Directives (or pragmas) [1] prescriptively specify properties for the realisation. Transformation parameters [11] for database realisation are, for instance, treatment of hierarchies, controlled redundancy, NULL marker support, constraint treatment, naming conventions, abbreviation rules, set or pointer semantics, handling of weak types, and translation options for complex attributes. Based on [2] we give an explicit specification of directives for the realisation. The prescription model also consists of a general description of a realisation style and tactics, of configuration parameters (coding, services, policies, handlers), of generic operations, of hints for realisation of the database, of performance expectations, of constraint enforcement policies, and of support features for the system realisation. These parameters are combined to the realisation template

$\mathcal{T}$. The realisation template can be extended by quantity matrix for database classes $\mathfrak{Q}$ and other performance constraints $\mathfrak{C}$ and by business tasks and their reflections through business data units $\mathfrak{B}$. Directives can be bound to one kind of platform and represent in this case a technological twist, e.g. by stating how data is layered out. They are typically however bound to several platforms in order to avoid evolution-proneness of models.

Conceptual model for prescription: *The conceptual database structure model comprises the database schema, a collection of views for both support of business users and system operating, a realisation template, and the declaration of model adequacy and dependability..*

Therefore, the model can be formally defined as a quintuple

$$(\mathcal{S},\, \mathfrak{V},\, \mathcal{T},\, \mathcal{A},\, \mathcal{D})\ .$$

6 Conclusion

This paper shows that the ER schema is a central unit in a conceptual database structure model. The conceptual database structure model contains also other elements in dependence on its function in utilisation scenarios. As long as we use a global-as-design approach, the ER schema is essential and the kernel of such database structure models.

We may combine the conceptual models to description/prescription models

$$(\mathcal{S},\, \mathfrak{V},\, (\mathfrak{R}, \Psi),\, \mathcal{T},\, \mathcal{A},\, \mathcal{D})\ .$$

and to description/prescription models with conceptualisation

$$(\mathcal{S},\, \mathfrak{V},\, (\mathfrak{R}, \Psi),\, \mathcal{M},\, \mathcal{T},\, \mathcal{A},\, \mathcal{D})\ .$$

The combination with communication/negotiation is more problematic since the corresponding models are based on divergent perception models that might represent the very personal viewpoint of business users in different context and work organisation.

The notion of the model for conceptual database structure models can be summarised in dependence on their utilisation scenario:

Table 1. Conceptual database structure models that extend the conceptual database schema in dependence on utilisation scenarios

Scenario	Model origin	Add-ons to the conceptual database schema
Communication and negotiation	Perception (and situation) models	Views representing the viewpoint variety and associated with the perception models
Conceptualisation	Perception and reality models	Associations to concepts and conceptions, semantics and meanings, namespaces
Description	Reality model	View collection, associations to origins
Prescription	Reality (and situation) models	View collection, realisation template

References

1. ISO/IEC JTC 1/SC 22. Information technology – Programming languages – C, ISO/IEC 9899:2011.
2. Bader AlBdaiwi, René Noack, and Bernhard Thalheim. Pattern-based conceptual data modelling. In *Information Modelling and Knowledge Bases XXVI, 24th International Conference on Information Modelling and Knowledge Bases (EJC 2014), Kiel, Germany, June 3-6, 2014*, pages 1–20, 2014.
3. Carlo Batini, Stefano Ceri, and Shamkant B. Navathe. *Conceptual Database Design: An Entity-relationship Approach*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1992.
4. Shirley Gregor. The nature of theory in information systems. *MIS Q.*, 30(3):611–642, September 2006.
5. Ibrahim A. Halloun. *Modeling Theory in Science Education*. Springer, Berlin, 2006.
6. Roland Kaschek. *Konzeptionelle Modellierung*. habilitation, University Klagenfurt, 2003.
7. Bernd Mahr. Information science and the logic of models. *Software and System Modeling*, 8(3):365–383, 2009.
8. H.A. Mannila and K.J. Räihä. *The Design of Relational Databases*. Addison-Wesley, Wokingham, England, 1992.
9. Herbert Stachowiak. Modell. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon zur Wissenschaftstheorie*, pages 219–222. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.
10. W. Steinmüller. *Informationstechnologie und Gesellschaft: Einführung in die angewandte Informatik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993.
11. Bernhard Thalheim. *Entity-Relationship Modeling: Foundations of Database Technology*. Springer-Verlag, Berlin, 2000.
12. Bernhard Thalheim. The conceptual framework to user-oriented content management. In Marie Duz, Hannu Jaakkola, Yasushi Kiyoki, and Hannu Kangassalo, editors, *Information Modelling and Knowledge Bases, volume XVIII*, Frontiers in Artificial Intelligence and Applications, pages 30–49. IOS Press, 2007.
13. Bernhard Thalheim. Towards a theory of conceptual modelling. *Journal of Universal Computer Science*, 16(20):3102–3137, 2010.
14. Bernhard Thalheim. *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges*. Springer-Verlag, Berlin, 2011.
15. Bernhard Thalheim. The conceptual model ≡ an adequate and faithful artifact enhanced by concepts. In *Information Modelling and Knowledge Bases XXV, 23rd European-Japanese Conference on Information Modelling and Knowledge Bases (EJC 2013), Nara, Japan, June 3-7, 2013*, pages 241–254, 2013.
16. Bernhard Thalheim. Models, to model, and modelling - Towards a theory of conceptual models and modelling - Towards a notion of the model, collection of recent papers, http://www.is.informatik.uni-kiel.de/∼thalheim/indexkollektionen.htm, 2014.
17. Bernhard Thalheim and Ivor Nissen, editors. *Wissenschaft und Kunst der Modellierung*. De Gruyter, Ontos Verlag, Berlin, 2015.

A Conceptual Model for Services

Bernhard Thalheim[1] and Ajantha Dahanayake[2]

[1] Department of Computer Science, Christian Albrechts University Kiel, D-24098 Kiel ***
[2] Prince Sultan University, Dept. of Computer Information Science, Riyadh, Kingdom of Saudi Arabia †

Abstract. Models are a mainstay of every scientific and engineering discipline. Models are typically more accessible to study than the systems. Models are instruments that are effectively functioning within a scenario. The effectiveness is based on an associated set of methods and satisfies requirements of usage of the model. A typical usage of a model is explanation, informed selection, and appropriation of an opportunity. This usage is declared through information and directions for usage or more specifically through an informative model in the case of a service model.

1 Services and The Conception of a Service

Today, the service has gained recognition as the more realistic concept for dealing with complexities of cross-disciplinary systems engineering extending its validity beyond the classical information systems design and development realm [4]. In this respect the service concept combines and integrates the value created in different design contexts such as person-to-person encounters, technology enabled self-service, computational services, multi-channel, multi-device, location-based and context-aware, and smart services [13]. Therefore, the service concept reveals the intrinsic design challenges of the information required to perform a service, and emphasizes the design choices that allocate the responsibility to provide this information between the service provider and service consumer.

1.1 Some Well-Known Service Notions

The service is being defined using different abstraction models with varying applications representing multitude of definitions of the service concept [7]. The increasing interests in services have introduced service concept's abstraction into levels such as; business services, web services, software-as-a-service (SaaS), platform-as-a-services, and infrastructure-as-a-service [2]. Service architectures are proposed as means to methodically structure systems [1, 5, 16].

There are number of service notations available in the in the literature, and research has looked into the service mainly from two perspectives, (a) from the

*** thalheim@is.informatik.uni-kiel.de
† adahanayake@pscw.psu.edu.sa

low-level technological point of view and (b) from the higher abstract business point of view. These two categories of service descriptions have derived number of service notations. Some of those main stream service notations are:

The *REA (Resource-Event-Agent) ontology* [8, 11] uses as core concepts resources, economic event, and agent. The *RSS (Resource-Service-Systems) model* [12] is an adaptation of REA ontology stressing that REA is a conceptual model of economic exchange and uses a Service-Dominant Logic (SDL) [21]. The *model of the three perspectives of services* uses abstraction, restriction, and co-creation. It concentrates on the use and offering of resources [2]. The perspectives addressed by this model are: service as a means for abstraction; service as means for providing restricted access to resources; and service as a means for co-creation of value. The logics behind is the Goods Dominant Logic (GDL) model [22]. *Web service description languages* concentrate on Service-Orientated Architectures (SOAs) for web service domain. Software systems are decomposed into independent collaborating units [14]. Named services interact with one another through message exchanges. The *seven contexts of service design* [6, 9, 13] combine person-to-person encounters, technology-enhanced encounters, self-service, computational services, multi-channel, multi-device, and location-based and context-aware services description.

1.2 The Explanation, Selection, and Appropriation

Explanation, understanding and informed selection of a tool is one of the main usage scenarios for a software models. People want to solve some problems. Services provide solutions to these problems and require a context, e.g. skills of people, an infrastructure, a specific way of work, a specific background, and a specific kind of collaboration. In order to select the right service, a model of the service is used as an *instrument for explanation and quick shallow understanding* which service might be a good candidate, what are the strengths and weaknesses of the service under consideration, which service meets the needs, and what are the opportunities and risks while deploying such a service.

The best and simplest instrument in such usage scenario is the *instruction leaflet* or more generally as a specification of the information and directions on the basis of the *informative model*. We shall show in the sequel that this model of a service extends the cargo dimension [10] to the general notion of the informative model. Such models of a service enable people in directed, purposeful, rewarding, realistic, and trackable deployment of a service within a given usage scenario, i.e. use according to the qualities of the model [4]. After informed selection of a service, it might be used in the creation of new work order based on the assimilation of the service into the given context, i.e. appropriation of the service.

1.3 Developing a Service Model based on the W*H Frame

Systems are typically characterised by a combination of large information content with the need of different stakeholders to understand at least some system aspects. People need models to assist them in understanding the context of their

own work and the requirements on it. We concentrate in this paper on the support provided by models to understand how a system works, how it can be used or should not be used, and what would be the benefit of such a model. We illustrate this utilisation of models for services.

We develop a novel service model based on the W*H specification frame [4]. The W*H model [4] provides a high-level and conceptual reflection and reflects on the variety of aspects that separates concerns such as service as a product, service as an offer, service request, service delivery, service application, service record, service log or archive and also service exception, which allows and supports a general characterization of services by their ends, their stakeholders, their application domain, their purpose and their context.

2 The Notion of a Model

The theory of models is the body of knowledge that concerns with the fundamental nature, function, development and utilisation of models in science and engineering, e.g. in Computer Science. In its most general sense, a model is a proxy and is used to represent some system for a well-defined purpose. Changes in the structure and behaviour of a model are easier to implement, to isolate, to understand and to communicate to others. In this section we review the notion of the model that has been developed in [18–20].

2.1 Artifacts that are Models

A model *is a well-formed, adequate, and dependable artifact that represents origins.* Its criteria of well-formedness, adequacy, and dependability must be commonly accepted by its community of practice within some context and correspond to the functions that a model fulfills in utilisation scenarios.

The model should be well-formed according to some well-formedness criterion. As an instrument or more specifically an artefact a model comes with its *background*, e.g. paradigms, assumptions, postulates, language, thought community, etc. The background its often given only in an implicit form. A model is used in a *context* such as discipline, a time, an infrastructure, and an application.

Models function as an instrument in some usage scenarios and a given usage spectrum. Their function in these scenarios is a combination of functions such as explanation, optimization-variation, validation-verification-testing, reflection-optimization, exploration, hypothetical investigation, documentation-visualisation, and description-prescription functions. The model functions effectively in some of the scenarios and less effectively in others. The function determines the *purpose* and the *objective* (or goal) of the model. Functioning of models is supported by methods. Such methods support tasks such as defining, constructing, exploring, communicating, understanding, replacing, substituting, documenting, negotiating, replacing, optimizing, validating, verifying, testing, reporting, and accounting. A model is *effective* if it can be deployed according to its objectives.

Models have several *essential properties* that qualify an artifact as a model. An well-formed artifact is *adequate* for a collection of origins if it is *analogous* to the origins to be represented according to some analogy criterion, it is more *focused* (e.g. simpler, truncated, more abstract or reduced) than the origins being modelled, and it sufficiently satisfies its *purpose*.

Well-formedness enables an artifact to be *justified* by an *empirical corroboration* according to its objectives, by rational coherence and conformity explicitly stated through formulas, by falsifiability, and by stability and plasticity.

The artifact is *sufficient* by its *quality* characterisation for internal quality, external quality and quality in use or through quality characteristics [17] such as correctness, generality, usefulness, comprehensibility, parsimony, robustness, novelty etc. Sufficiency is typically combined with some assurance evaluation (tolerance, modality, confidence, and restrictions).

A well-formed artifact is called *dependable* if it is sufficient and is justified for some of the justification properties and some of the sufficiency characteristics.

2.2 Artifacts as Instruments in Some Usage Scenario

Models will be used, i.e. there is some usage scenario, some reason for its use, some goal and purpose for its usage and deployment, and finally some function that the model has to play in a given usage scenario. A typical usage scenario is problem solving. We first describe a problem, then specify the requirements for its solutions, focus on a context, describe the community of practices and more specifically the skills needed for the collaborative solution of the problem, and scope on those origins that must be considered. Next we develop a model and use this model as an instrument in the problem solving process. This instrument provides a utility for the solution of the problem. The solution developed within the model setting is then used for derivation of a solution for the given problem in the origin setting.

A similar use of models is given for models of services. Service models might be used for the development of a service system. They might be used for assessment of services, for optimisation and variation of services, for validation-verification-testing, for investigation, and for documentation-visualization. In this paper we concentrate on the *explanation, informed selection, and appropriation* use of a service model. It must provide a high level description of the service itself. This usage is typical for a process of determining whether a service is of high utility in an application. Such usage is based on specific usage pattern or more specifically on a special model that is the *usage model of an instrument as a model*.

2.3 Conceptional Modelling: Modelling Enhanced by Concepts

An information systems model is typically a schematic description of a system, theory, or phenomenon of an origin that accounts for known or inferred properties of the origin and may be used for further study of characteristics of the

origin. *Conceptional modelling* [1] aims to create an abstract representation of the situation under investigation, or more precisely, the way users think about it. *Conceptual models* enhance models with concepts that are commonly shared within a community or at least within the community of practice in a given usage scenario. Concepts specify our knowledge what things are there and what properties things have. Their definition can be given in a narrative informal form, in a formal way, by reference to some other definitions, etc. We may use a large variety of semantics [15], e.g., lexical or ontological, logical, or reflective.

2.4 Adequacy and Dependability of Informative Models

Models are used in *explanation, informed selection, and appropriation* scenarios. We call such models *informative models*. Their main aim of is to inform the user according to his/her information demand and according to the profile and portfolio. The instrument steers and directs its users which are typically proactive. It supplies information that is desired or needed. Users may examine and check the content provided. Typical methods of such instruments are communication, orientation, combination, survey, and feedback methods.

Users have to get informed what is the issue that can be solved with the instrument, what are the main ingredients of the instrument and how they are used, what is the main background behind this instrument, and why they should use this instrument. They need a quick shallow understanding how simple, how meaningful, how adequate, how realistic, and how trackable is the instrument (*SMART*). They must be enabled to select the most appropriate instrument, i.e. they should know the strengthes, weaknesses, opportunities, and threats of the given instrument (*SWOT*).

The SWOT and SMART evaluation is the basis for adequateness and dependability of informative models. The informative model must be analogous in structure and function to its origins. It is far simpler than the origin and thus more focussed. Its purpose is to explain the origin in such a way than a user can choose this instrument because of its properties since all demanded properties are satisfied. The selection and appropriation of an instrument by the user depends on the explanatory statement on the profile and the portfolio of the given instrument, on coherence to the typical norms and standards accepted by the community of practice, on a statement on applicability and added value of the instrument, and the relative stability of the description given. The instrument usage becomes then justified. Furthermore, the instrument must suffice the demands of such scenarios. The quality in use depends on understandability and parsimony of description, worthiness and eligibility of presented origins, and

[1] The words 'conceptual' and 'conceptional' are often considered to be synonyms. The word 'conceptual' is linked to concepts and conceptions. 'Conceptual' means that a thing - e.g. an instrument or artifact - is characterised by concepts or conceptions. The word 'conceptional' associates a thing as being or of the nature of a notion or concept. Conceptional modelling is modelling with associations to concepts. A conceptual model incorporates concepts into the model.

the added value it has for the given utilisation scenarios. The external quality is mainly based on its required exactness and validation. The internal quality must support these qualities. The quality evaluation and the quality safeguard is an explicit statement of these qualities according to the usage scenarios, to the context, to the origins that are represented, and to community of practice.

2.5 The Cargo of a Model

The cargo of any instrument is typically a very general instrument insert like the package insert in pharmacy or an enclosed label. It describes the instrument, the main functions, the forbidden usages, the specific values of the instrument, and the context for the usage model. Following [10, 20] we describe the cargo by a description of the *mission* of the instrument in the usage scenarios, the *determination* of the instrument, an *abstract declaration of the meaning* of the instrument, and a narrative explanation of the *identity* of the instrument.

The mission of a model consists of functions (and anti-functions or forbidden ones) that the model has in different usage scenarios, the purposes of the usages of the model, and a description of the potential and of the capacity of the model. The determination contains the basic ideas, features, particularities, and the usage model of the given instrument. The meaning contains the main semantic and pragmatic statements about the model and describes the value of the instrument according to its functions in the usage scenarios, and the importance within the given settings. Each instrument has its identity, i.e. the actual or obvious identity, the communicated identity, the identity accepted in the community of practice, the ideal identity as a promise, and the desired identity in the eyes of the users of the instrument.

2.6 The Informative Model

The *informative model* consists of the cargo, the description of its adequacy and dependability, and the SMART and SWOT statements. It informs a potential users through bringing facts to somebody's attention, provides these facts in an appropriate form according their information demand, guides them by steering and directing, and leads them by changing the information stage and level. Based on the informative model, the user selects the origin for usage with full informed consent or refuses to use it. It is similar to an instruction leaflet provided with instruments we use. The informative model is semantically characterized by: objectivity; functional information; official information; explanation; association to something in future; different representational media and presenters; degree of extraction from open to hidden; variety of styles such as short content description, long pertinent explanation, or long event-based description.

In the case of a service model, the informative model must state positively and in an understandable form what is the service, must describe what is the reward of a service, and must allow to reason about the rewards of the service, i.e. put the functions and purposes in a wider context ($PURE$). Informative models of a service are based on a presentation that is easy-to-survey and to understand,

that is given in the right formatting and form, that supports elaboration and surveying, that avoids learning efforts for their users, that provides the inner content semantics and its inner associations, that might be based on icons and pictographs, and that presents the annotation and meta-information including also ownership and usability.

We shall now explore in the sequel what are the ingredients of such informative instruments in the case of a service model.

3 Service Specifications

3.1 Scenarios and Functions of Service Specifications

To capture the scenarios and functions of service specification we introduce *W*H model* in Figure 1 that is a novel conceptual model for service modelling.

Service	Service Name			
Concept	Ends	*Wherefore?*		
		Purpose	*Why?*	
			Where to?	
			For When?	
			For Which reason?	
Content	Supporting means	*Wherewith?*		
		Application Domain	Application are	*Wherein?*
			Application case	*Wherefrom?*
			Problem	*For What*
			Organizational unit	*Where*
			Triggering Event	*Whence*
			IT	*What* *How*
Annotation	Source	*Where of?*		
		Party	Supplier	*By whom?*
			Consumer	*To whom?*
			Producer	*Whichever?*
		Activity	Input	*What in?*
			Output	*What out?*
Added Value	Surplus Value	*Worthiness?*		
		Context	Systems Context	*Where at?*
			Story Context	*Where about?*
			Coexistence Context	*Wither?*
			Time Context	*When?*

Fig. 1. The W*H Specification Frame for the Conceptual Model of a Service

The W*H model in Figure 1 fulfills the conceptual definition of the service concept composing the need to serve the following purposes:

- The composition of the W*H model consisting of *content space*, *concept space*, *annotation space*, and *add value space* as orthogonal dimensions that captures the fundamental elements for developing services.
- It reflects number of aspects neglected in other service models, such as the handling of the service as a collection of offering, a proper annotation facility, a model to describe the service concept, and the specification of added value. It handles those requirements at the same time.
- It helps capturing and organizing the discrete functions contained in (business) applications comprised of underlying business process or workflows into interoperable, (standards-based) services.
- The model accommodates the services to be abstracted from implementations representing natural fundamental building blocks that can synchronize the functional requirements and IT implementations perspective.
- It considers by definition that the services to be combined, evolved and/or reused quickly to meet business needs.
- Finally, it represents an abstraction level independent of underlying technology.

In addition, the W*H model in 1 also serves the following purposes:

- The inquiry through simple and structured questions according to the primary dimension on wherefore, whereof, wherewith, and worthiness further leading to secondary and additional questions along the concept, annotation, content, add value or surplus value space that covers usefulness, usage, and usability requirements in totality.
- The powerful inquiring questions are a product of the conceptual underpinning of W*H grounded within the conceptional modelling tradition in the Concept-Content-Annotation triptych extended with the Added Value dimension and further integration and extension with the inquiry system of Hermagoras of Temnos frames.
- The W*H model is comprise of 24 questions in total that cover the complete spectrum of questions addressing the service description; (W5 + W4 + W10H +W4) and H stands for how.
- The models compactness helps to validate domain knowledge during solution modelling discussions with the stakeholders with high demanding work schedules.
- The comprehensibility of the W*H model became the main contributor to the understanding of the domain's services and requirements.
- The model contributes as the primary input model leading to the IT-service systems projection on solution modelling.
- It contribute as the primary input model leading to the IT-service systems projection on the evaluations criteria of systems functioning on its trustworthiness, flexibility to change, and efficient manageability and maintainability.

3.2 Dimensions of Service Specification

The Content Dimension: Services as a Collection of Offerings. The service defines the what, how, and who on what basis of service innovation, design, and

development, and helps mediate between customer or consumer needs and an organizations strategic intent. When extended above the generalized business and technological abstraction levels, the content of the service concept composes the need to serve the following purposes:

- Fundamental elements for developing applications;
- Organizing the discrete functions contained in (business) applications comprised of underlying business process or workflows into inter operable, (standards-based) services;
- Services abstracted from implementations representing natural fundamental building blocks that can synchronize the functional requirements and IT implementations perspective;
- Services to be combined, evolved and/or reused quickly to meet business needs; Represent an abstraction level independent of underlying technology.

The abstraction of the notion of a service system within an organizations strategic intent emphasized by those purposes given above allow us to define the content description of services as a collection of offers that are given by companies, by vendors, by people and by automatic software tools [3]. Thus the content of a service system is a collection of service offerings.

The service offering reflects the supporting means in terms of with what means the service's content is represented in the application domain. It corresponds to identification and specification of the problem within an application area. The problem is a specific application case that resides with an organizational unit. Those problems are subject to events that produce triggers needing attention. Those triggering events have an enormous importance for service descriptions. They couple to the solution at hand that is associated with how and what of a required IT solution.

The Annotation Dimension. According to [14], annotation with respect to arbitrary ontologies implies general purpose reasoning supported by the system. Their reasoning approaches suffer from high computational complexities. As a solution for dealing with high worst-case complexities the solution recommends a small size input data. Unfortunately, it is contradicting the impressibility of ontologies and define content as complex structured macro data. It is therefore, necessary to concentrate on the conceptualisation of content for a given context considering annotations with respect to organizations intentions, motivations, profiles and tasks, thus we need at the same time sophisticated annotation facilities far beyond ontologies. Annotation thus must link the stakeholders or parties involved and activities; the sources to the content and concept.

The Concept Dimension. Conceptional modelling aims at creation of an abstract representation of the situation under investigation, or more precisely, the way users think about it. Conceptual models enhance models with concepts that are commonly shared within a community or at least between the stakeholders involved in the modelling process.

According to the general definition of concept as given in [19], *Concepts* specify our knowledge what things are there and what properties things have. Concepts are used in everyday life as a communication vehicle and as a reasoning chunk. Concept definition can be given in a narrative informal form, in a formal way, by reference to some other definitions etc. We may use a large variety of semantics, e.g., lexical or ontological, logical, or reflective.

Conceptualisation aims at collection of concepts that are assumed to exist in some area of interest and the relationships that hold them together. It is thus an abstract, simplified view or description of the world that we wish to represent. Conceptualisation extends the model by a number of concepts that are the basis for an understanding of the model and for the explanation of the model to the user.

The definition of the ends or purpose of the service is represented by the concept dimension. It is the curial part that governs the service's characterization. The purpose defines in which cases a service has a usefulness, usage, and usability. They define the potential and the capability of the service.

The Added Value Dimension. The added value of a service to a business user or stakeholder is in the definition of surplus value during the service execution. It defines the context in which the service systems exists, the story line associated within the context, which systems must coexist under which context definitions prevailing to time. Surplus value defines the worthiness of the service in terms of time and labor that provide the Return of Investment (ROI).

4 Conclusion

There are many other usage models for services. This paper elaborated the *explanation, informed selection, and appropriation* usage model for a service. Other usage models of an instrument as a model are, for instance, optimization-variation, validation-verification-testing, understanding, extension and evolution, reflection-optimization, exploration, documentation-visualization, integration, hypothetical investigation, and description-prescription usage models. We introduced in this paper a general notion of the model and showed what makes description or specification a service to be become a model of the service.

References

1. A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, SOMA: A method for developing service-oriented solutions. *IBM Systems Journal* 47(3), 2008: 377-396.
2. M. Bergholtz, B. Andersson, and P. Johannesson, *Abstraction, Restriction, and Co-creation: Three Perspectives on Services.* ER 2010 Workshops of Conceptual Modeling of Services, LNCS 6413, 107-116, Springer, Berlin, 2010.
3. A. Dahanayake. *CAME: An Environment for Flexible Information Modeling.* PhD Dissertation, Delft University of Technology, the Netherlands, 1997.

4. A. Dahanayake and B. Thalheim. Co-Design of Web Information Systems. *Correct Software in Web Applications and Web Services*, 145-176, Springer, Wien, 2015.
5. T. Erl. *SOA: principles of service design*. Prentice-Hall, Englewood Cliffs, 2007.
6. R.J. Glushko. *Seven Contexts for Service System Design*. P.P. Maglio et al. (eds.), Handbook of Service Science, Service Science: Research and Innovations in the Service Economy, DOI 10.1007/978-1-4419-1628-0_11, Springer Science+Business Media, LLC 2010.
7. S.M. Goldstein, R. Johnston, J.-A. Duffy, J. and Rao. The service concept: the missing link in service design research?. *Journal of Operations Management* 2002, 20, 212-134, Elsevier.
8. P. Hurby. *Model-Driven Design of Software Applications with Business Patterns*. Springer, Heidelberg, 2006.
9. P. Maglio, S. Srinivasan, J. Kreulen, and J. Spohrer. Service Systems, Service Scientists, SSME, and Innovation. *Communications of the ACM*, 2006, 49(7), 81-85.
10. B. Mahr. Zum Verhältnis von Bild und Modell. *Visuelle Modelle*, 17–40. Wilhelm Fink Verlag, Mnchen, 2008.
11. W.E. McCarthy. The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *Accounting Review* 57, 1982, 554-578.
12. G. Poels. *The Resource-Service-System Model for Service Science*. ER2010 Workshops, LNCS 6413, Springer, 117-126, 2010.
13. J. Spohrer, P.P. Maglio, J. Bailey, and D. Gruhl. Steps Towards a Science of Service Systems. *IEEE Computer* 40, 2007, 71-77.
14. K.-D. Schewe and B. Thalheim. Development of collaboration frameworks for web information systems. In *IJCAI'07* (20th Int. Joint Conf on Artificial Intelligence, Section EMC'07 (Evolutionary models of collaboration)), 27-32, Hyderabad, 2007.
15. K.-D. Schewe and B. Thalheim. Semantics in data and knowledge bases. *SDKB 2008*, LNCS 4925, 1-25, Springer, Berlin, 2008.
16. Z. Stojanovic and A. Dahanayake. *Service - Oriented Software Systems Engineering: Challenges and Practices*. Idea Group Publishing, PA, USA, 2004.
17. B. Thalheim. Towards a theory of conceptual modelling. *Journal of Universal Computer Science*, 16(20):3102–3137, 2010. http://www.jucs.org/jucs_16_20/towards_a_theory_of.
18. B. Thalheim. The conceptual model ≡ an adequate and dependable artifact enhanced by concepts. *Information Modelling and Knowledge Bases*, vol. XXV, *Frontiers in Artificial Intelligence and Applications, 260*, 241-254. IOS Press, 2014.
19. B. Thalheim. Models, to model, and modelling - Towards a theory of conceptual models and modelling - Towards a notion of the model. *Collection of recent papers*, http://www.is.informatik.uni-kiel.de/∼thalheim/indexkollektionen.htm, 2014.
20. B. Thalheim and I. Nissen, editors. *Wissenschaft und Kunst der Modellierung*. De Gruyter, Ontos Verlag, Berlin, 2015.
21. S.L. Vargo and R.F. Lusch. Evolving to a New Dominant Logic for Marketing. *Journal of Marketing* 68, 2004, 1-17.
22. S.L. Vargo, P.P. Maglio, and M.A. Akaka. On value and value co-creation: A service systems and service logic perspective. *European Management Journal* 26, 2008, 145-152.

Models and their Capability

Bernhard Thalheim and Marina Tropmann-Frick

Department of Computer Science, Christian-Albrechts-University Kiel, 24098 Kiel, Germany

Abstract. Models are one of the main instruments in Computer Science. The notion of model is however not commonly agreed due to the wide usage of models. It is challenging to find an acceptable and sufficiently general notion of model due to the large variety of known notations. Such notion should incorporate all of the different notations and at the same time should allow to derive the specific notation from the general notion. We introduce a universal parameterised notion of the model. The parameters in this notion support adaptation of the universal notion of the model to the specific notation of interest. We finally apply this notion and this adaptation to development of business process models that are specified in BPMN.

1 The Model - an Artifact and an Instrument

Classical Computer Science research considers models as *artifacts*[1] that are constructed in certain way and prepared for their utilisation according to the purpose under consideration such as construction of systems, verification, optimization, explanation, and documentation.

Creation for a practical purpose means that the main target of model development is its application in utilisation scenarios. Models are considered to be artifacts in a stronger sense. We observe however that models are developed for their utilisation within some scenario. They are functioning in this scenario. That means models are instruments in these scenarios. The notion of an instrument[2] concentrates on this utilisation of models. Models are therefore mainly *instruments* that are effectively functioning within a scenario. The effectiveness is based on an associated set of methods and satisfies requirements of usage of the model.

1.1 Models - The Third Dimension of Science

Models are used as perception models, experimentation models, formal models, conceptual models, mathematical models, computational models, physical

[1] An artifact is "something that is created by humans usually for a practical purpose" or "something characteristic of or resulting from a particular human institution, period, trend, or individual" or "a product of artificial character due usually to extraneous (as human) agency" [16]. The last meaning of the notion of an artifact is not taken into consideration for models in most sciences and also in Computer Science.

[2] An instrument is among others (1) a means whereby something is achieved, performed, or furthered; (2) one used by another as a means or aid or tool [16].

models, visualisation models, representation models, diagrammatic models, exploration models, heuristic models, etc. Experimental and observational data are assembled and incorporated into models and are used for further improvement and adaptation of those models. Models are used for theory formation, concept formation, and conceptual analysis. Models are used for a variety of purposes such as perception support for understanding the application domain, for shaping causal relations, for prognosis of future situations and of evolution, for planning, for retrospection of previous situations, for explanation and demonstration, for preparation of management, for optimisation, for construction, for hypothesis verification, and for control of certain environments.

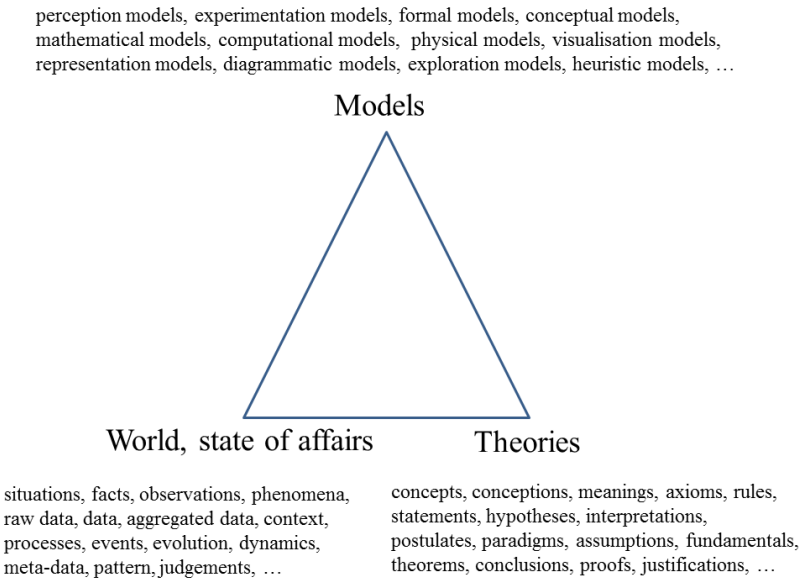


Fig. 1. Models - The third dimension of science

Models are one of the main instruments in scientific research. They are considered to be the *third dimension of science* [26][3] (Figure 1). They provide a tool for research and have an added value in research, e.g. for construction of systems, for education, for the research process itself. Their added value is implicit but can be estimated based on the capability, potential and capacity of the model. Models are common culture and practice in sciences. However, each discipline has developed its own modelling expertise and practice.

Models are often language based. Their syntax uses the namespace and the lexicography from the application domain. Semantics is often implicit. The lexicology can be inherited from the application domain and from the discipline. Models do not need the full freedom for interpretation. The interpretation is governed by the purpose of the model within the research scenario,

[3] The title of the book [4] has inspired this observation.

is based on disciplinary concerns (postulates, paradigms, foundations, commonsense, culture, authorities, etc.) and is restricted by disciplinary practices (concepts, conceptions, conventions, thought style and community [6], good practices, methodology, guidelines, etc.). Models combine at least two different kinds of meaning in the namespace: referential meaning establishes an interdependence between elements and the origin ('what'); functional meaning is based on the function of an element in the model ('how'). The pragmatics of a model depends on the community of practice, on the context of the research task and especially on the purpose or function of the model.

A model can be used for different purposes and various usage scenarios. Therefore, a model is typically also extended by views or viewpoints that reflect certain parts of the model and that hide details which are not necessary. This reflection is often only provided in a non-systematic or implicit way. Additionally, we need a refinement notion, methods for combination and for evaluation of models.

1.2 Scenarios of Model Utilisation

Models are used as an instrument in some utilisation scenario. At the same time, the model might be useless and not productive in other scenarios. Their function in these scenarios is a combination of functions such as explanation, optimization-variation, validation-verification-testing, reflection-optimization, exploration, hypothetical investigation, documentation-visualization, and description-prescription as a mediator between a reality and an abstract reality that developers of a system intend to build.

Traditionally, purposes or goals are considered first. The purposes and the goals are used to determine the functions of a model. This approach is centered around the purpose or goal and requires a definite understanding of the purpose and goal. Purposes and goals are often underspecified or blurry at the beginning. They become more clear after the model is being used. Compared to this approach, it is simpler to understand the application cases of a model and thus the utilisation scenarios. In this case we may derive the functions that a model has in these scenarios. Therefore, we use the approach that the functions of the model determine the *purposes* of the deployment of the model.

1.3 The Storyline of the Paper

The large variety of model notations (see, for instance, [13, 23, 30]) does not allow to transfer experience gained with one notation to other notations. Methods for utilisation or development are therefore mainly bound to one notation. Each subdiscipline has therefore its own understanding of modelling. It would however be beneficial to have a general notion of model that can be adapted to the specific notations of interest.

We introduce in Section 2 a universal notion of a model. This notion is based on the understanding of a model as an instrument in some utilisation scenarios. We only consider well-formed instruments since models must be intuitive and easy to understand. The model definition is based on two general

parameter sets, adequateness and dependability. Each of the parameters can be instantiated in dependence of the function that the model should have in a given utilisation scenario within the sub-discipline. This instantiation facility is based on a conception frame for the model notion.

The approach is applied to BPMN modelling in Section 3. We describe the business process modelling approach and derive the capability of this modelling technique. We can now also explicitly describe the obstacles of BPMN modelling. Furthermore, we derive the evaluation procedure for the BPMN approach in Section 4.

This approach to modelling in Computer Science can now be used as a starting point of a theory of modelling (Section 5). We start with some, often implicitly given restrictions that a model has, esp. its burden by the background and by the directives. The evaluation of models also supports a statement on not-supported utilisations, called anti-profile. Finally, the conception frame can also be used for development of question forms that support model specification.

2 The Universal Notion of the Model

There are many notions of models. Each of them covers some aspects and concentrates on some properties such as the mapping, analogy, truncation, pragmatism, amplification, distortion, idealisation, carrier, added value, and purpose properties [11, 17, 18, 21]. The main property is however the *function property: The model suffices in its function in the utilisation scenarios that are requested.* This property results in the following notion of the model [25, 27, 29].

2.1 The Model Notion

Models have several *essential properties* that qualify an instrument as a model [22, 24]:

Definition 1. *An instrument is* well-formed *if it satisfies a well-formedness criterion.*

Definition 2. *A well-formed instrument is* adequate *for a collection of origins if (i) it is analogous to the origins to be represented according to some analogy criterion, (ii) it is more focused (e.g. simpler, truncated, more abstract or reduced) than the origins being modelled, and (iii) it is sufficient to satisfy its purpose.*

Definition 3. *Well-formedness enables an instrument to be* justified*: (i) by an empirical corroboration according to its objectives, supported by some argument calculus, (ii) by rational coherence and conformity explicitly stated through formulas, (iii) by falsifiability that can be given by an abductive or inductive logic, and (iv) by stability and plasticity explicitly given through formulas.*

Definition 4. *An instrument is* sufficient *by a* quality *characterisation for internal quality, external quality and quality in use or through quality characteristics [20] such as correctness, generality, usefulness, comprehensibility, parsimony, robustness, novelty etc. Sufficiency is typically combined with some assurance evaluation (tolerance, modality, confidence, and restrictions).*

Definition 5. *A well-formed instrument is called* dependable *if it is sufficient and is justified for some of the justification properties and some of the sufficiency characteristics.*

Definition 6. *An instrument is called* **model** *if it is* adequate *and* dependable. *The adequacy and dependability of an instrument is based on a* judgement *made by the community of practice.*

Definition 7. *An instrument has a* background *consisting of an undisputable grounding from one side (paradigms, postulates, restrictions, theories, culture, foundations, conventions, authorities) and of a disputable and adjustable basis from other side (assumptions, concepts, practices, language as carrier, thought community and thought style, methodology, pattern, routines, commonsense).*

Definition 8. *A model is used in a* context *such as discipline, a time, an infrastructure, and an application.*

The model notion can be depicted in Figure 2 based on the following conceptions:

a fundament or ***background*** with
- the grounding, and
- the (meta-)basis,

four governing directives given by
- the artifacts or better origins to be represented by the model,
- the deployment or profile of the model such as goal, purpose or functions,
- the community of practice (CoP) acting in different roles on certain rights through some obligations, and
- the context of time, discipline, application and scientific school,

two pillars which provide
- methods for development of the model, and
- methods for utilisation of the model,

and finally

the model utilisation scenario for the deployment of the model in the given application.

The *model house* in Figure 2 abstracted from its full version [24, 27] displays these different facets of the model. The house consists of a cellar (basis in Figure 2) and a fundament (grounding in Figure 2), two pillars (development resp. utilisation methods), four driving or governing forces (origins, purpose of function, community of practice, context), and finally the deployment roof (utilisation scenario). The *grounding* is typically implicitly assumed and not disputable. It contains paradigms, the culture in the given application area, the background, foundations and theories in the discipline, postulates, (juristic and other) restrictions, conventions, and the commonsense. The *basis* is the main part of the background and is typically disputable.

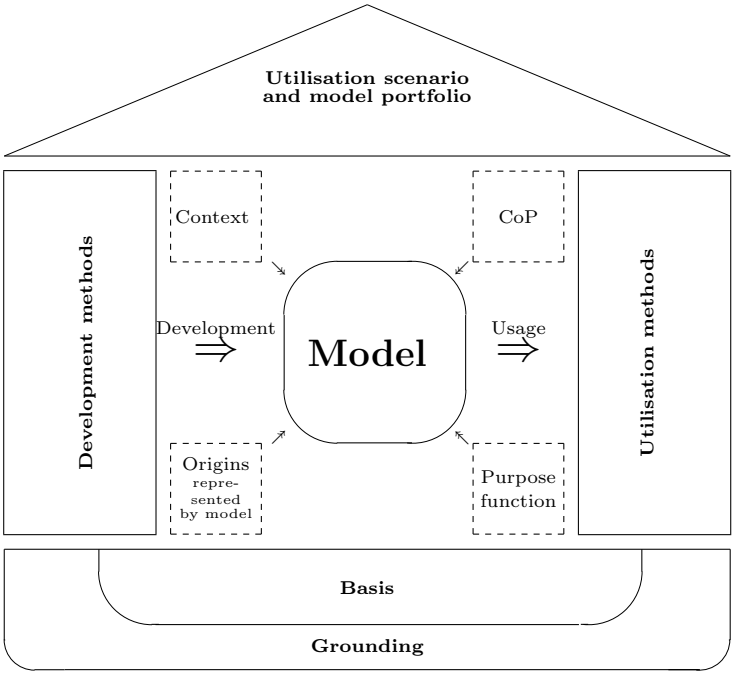


Fig. 2. Facets of the model notion

Definition 9. *A fully-specified model is* function-purpose-goal invariant *if the model can be used instead of the origins in the given scenario and have the same goal, the same purpose, and the same function. A model is* solution-faithful *if the solution of the problem solved with the model is analogous in the world of the origins based on the analogy criterion that is used for stating adequacy.*

2.2 The Conception Frame for the Model Notion

The model notion covers many different aspects. It might thus be of interest whether there is a guideline for development of models. Models are artifacts that can be specified within a W*H-frame [5] that extends the classical rhetorical frame introduced by Hermagoras of Temnos[4]. Models are primarily characterised by W[4]: wherefore (purpose), whereof (origin), wherewith (carrier, e.g. language), and worthiness ((surplus) value). The secondary characterisation dimensions are given by: (1) stakeholder: by whom, to whom, whichever; (2) additional properties of the application domain: wherein, where, for what, wherefrom, whence, what; (3) solution: how, why, whereto, when, for which reason; and (4) context: whereat, whereabout, whither, when.

A practical guideline may just

[4] Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (Who, what, when, where, why, in what way, by what means), The Zachman frame uses a simplification of this frame.

1. start with fixation of two directives: origins to be represented and community of practice that accepts this model;
2. restrict the model utilisation scenario and the usage model to those that are really necessary and thus derive the purpose and function of the instrument;
3. define adequateness and dependability criteria of the instrument within the decision set made so far;
4. explicitly describe the background of the model, i.e. its undisputable grounding and the selected basis; and
5. explicitly specify the context for utilisation of the model.

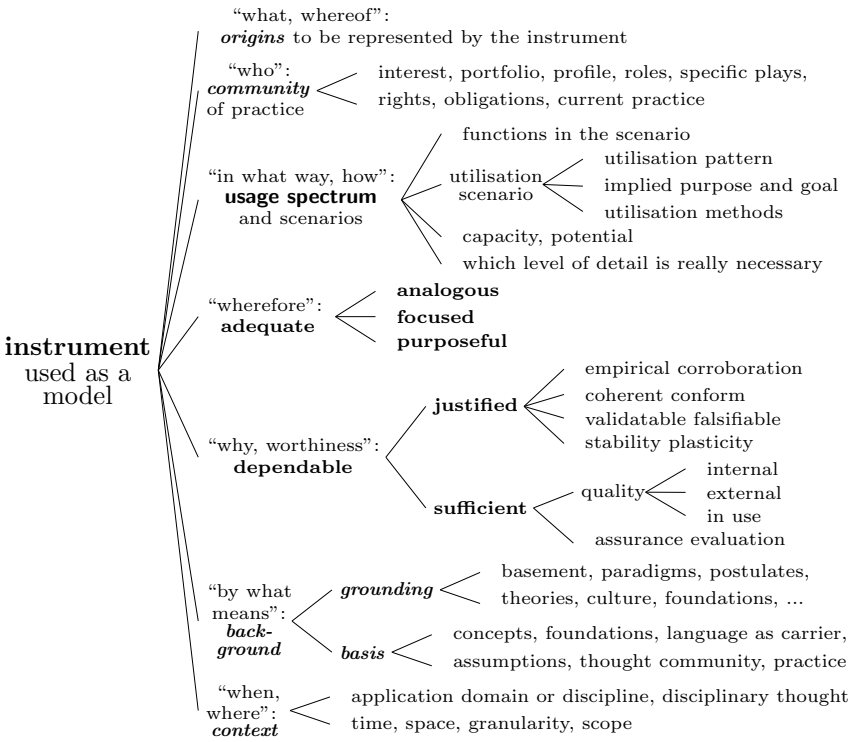


Fig. 3. Conception frame for systematic development of a model

The model development and utilisation depends in this case from:

Judgements of some members of the CoP to deploy the instrument as a model for some origin based on an assessment (deployability, rigidity, modality, confidence) within a CoP, utilisation scenario, and within a context.

Utilisation scenarios and use spectra accepted for the instrument with
functions of the instrument in utilisation scenario,
roles and deployment of the instrument in those scenario, and
resulting purposes and goals for the utilisation.

The instrument as such with some appreciation
as a well-shaped instrument on the basis
– of some criteria in dependence on intended utilisation and criteria for:
  - what is accepted in a CoP, and
  - what is syntactically, semantically, pragmatically well-shaped,

that fits to the intended use, and
is appropriate for the utilisation scenarios and the use spectra.

The orientation also reflect our understanding of a model as an instrument.

3 BPMN Diagrams as Models

The Business Process Modeling and Notation (BPMN) language [8, 14] is a conceptual business process specification language and is standardized by the Object Management Group (OMG). There are many different languages for description of business stories (e.g. SiteLang), of business rules (e.g. business use cases), and of workflows that are essentially specifications of business processes, activities of participants, utilisation with resources, and of communication among the participants. Languages such as S-BPM, BPMN, and EPC concentrate on different aspects of business processes, vary in scope and focus, use different abstraction levels, and are thus restricted in the capacity and potential for modelling. Most of the existing languages evolved over their lifespan and extensionally added features, more features, and other features again. BPMN is not an exception for this kind of overloading.

A business process consists of an ordered set of one or more activities (tasks) which collectively realize a business objective or policy goal. A workflow is the executable specification of a business process. It may describe all or some of the five aspects of business processes [15]:

(1) control flow description for the partial order of the activities, events or steps;
(2) organisation description with participants, theirs roles and plays within the processes, their rights and obligations, their resources, and their assignments;
(3) the data viewpoint description with an association to process elements and access rights for participants;
(4) the functional description that specifies semantics, pragmatics, and behaviour of each element of the workflow, e.g. the operations to be performed, pre- and postconditions, priority, triggers, and time frames for the operations;
(5) the operational assignment of programs that support all elements of the workflow.

The entire modelling process is based on a local-as-design perspective. A holistic or global view on a diagram collection is the task of a designer and becomes problematic in the case of specification evolution.

BPMN 2.0 defined four different kinds of diagrams for workflow specification. We shall briefly review these diagrams in the sequel. The diagram in Figure 4 combines these different aspects. It describes the accomplishment of requirements issued by a customer.

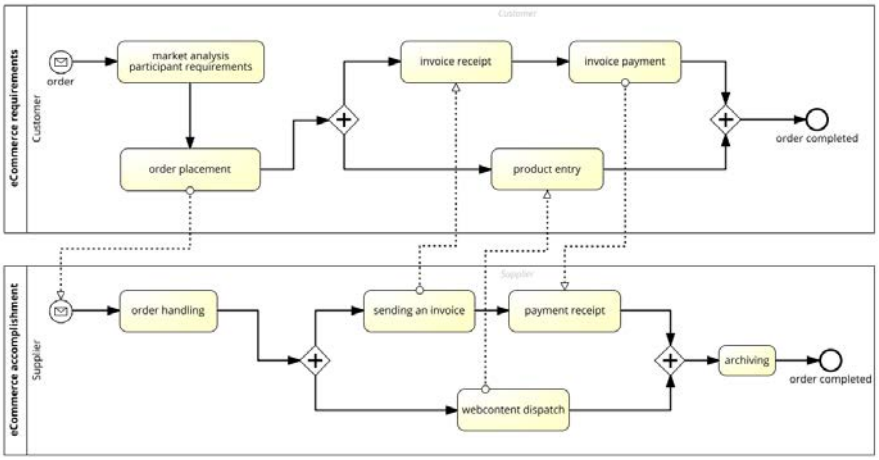


Fig. 4. Fulfillment of customer demands by vendors

3.1 Diagrams in BPMN

Process Diagrams. Process diagrams (also called orchestration diagrams) describe the stepwise task flow for one agent. A task flow might reflect different roles of an agent. These roles are separated by swimlanes. Processes are either public or private. Public processes can also be abstractions of private processes that represent the detailed control and task flow for a singleton agent. Main process elements are (a) atomic or complex activities for direct representation of stepwise actions of an agent, (b) gates for exclusive, non-exclusive, event-based or parallel splitting and joining of the control flow, (c) events for the start of a workflow, for the end of a workflow path, for the complete end of the entire workflow, and an intermediate event for representation of interaction events with agents outside the workflow, and (d) control flow arrows for representation of the order of process elements. Basic activities reflect abstract, service, send, receive, user, manual, business rule, or script tasks. Complex tasks reflect an entire sub-workflow, loops, parallel or sequential multiple executions, ad-hoc workflows, transactions, or specific exception handling workflows such as compensation. Interaction reflects message exchange, timer interaction, escalation enforcement, compensations, conditional interactions, links, and signals. Interaction can be sequential or parallel multiple. Interaction events may also

be boundary events for a complex activity. All elements can be explicitly annotated by comments, by consumed data, or by produced data objects or data stores.

Choreography Diagrams. Choreography diagrams describe the message exchange among agents with reference to sending and receiving events, the message issue, and the graph-based representation of the partial order among these messages.

Collaboration Diagrams. Collaboration use choreography diagrams and process diagrams for explicit binding of senders and receivers of messages to black-box abstractions of agent workflows and abstract from message issues.

Conversation Diagrams. Conversation diagrams survey communication flow among agents as a birds view. They allow to derive dependences among process diagrams of agents.

3.2 Capability of BPMN Diagrams

BPMN modelling becomes nowadays a standard for typical business applications. Therefore, the capability of processes must be specified and well understood. It is thus necessary to know what is the ability to achieve a good model through a set of controllable and measurable features.

BPMN diagrams require a work-around for a number of conceptions such as macro-state, history, and system architecture. There are redundancies in the language itself that lead to flavour- or taste-oriented programming due to the overwhelming number of elements, construct excess and overload, e.g. groups, pool and lane, transformations, off-page connectors. The structuring becomes unclear since activities can be itself a workflow or a collection of workflows. This rather specific kind of abstraction should not be mixed with abstraction in general. Exception handling is completely confusing and only partially defined.

BPMN diagrams can represent only 8 out of 43 workflow resource pattern [10]. The data aspect is provided through properties of tasks, processes, and sub-processes. Their interrelationship is left to the developer community. It is the task of the developers to keep in mind the entire picture of the BPMN diagram collection.

BPMN uses an informal approach to semantics description what has been a matter of confusion. A formal approach to BPMN semantics can however be developed [1–3].

Furthermore, there is no conception of well-formed diagrams. Decomposition and composition is left to the developer. BPMN does not properly support the aspects (2), (4), and (5). The data aspect (3) is partially represented.

3.3 Deficiencies of Diagrams and Diagrammatical Reasoning

Diagrams are not universal for modelling. It is often claimed that diagrams are simple to use, are easy to interpret, have an intuitive semantics, are unique within a user community and have thus a unique pragmatics, and are thus powerful instruments. We observe however a number of obstacles that must be resolved before accepting a diagram as a model, e.g. the following ones:

- *Habituation versus unfamiliarity*: Diagram should be familiar to their users, have a unique semantics and pragmatics without any learning effort. Readers of diagrams must be literal with them.
- *Ambiguity of interpretation versus well-formedness*: Diagrams should not confuse by multiple interpretations (e.g. arrows), by instability and by context-dependence of form-content relations.
- *Incremental graphical construction*: Diagrams should follow the same construction pattern as the origin and should concentrate on typicality.
- *Naturalness of local reasoning*: Local-as-design approaches presuppose locality within the world of origins.
- *Unfamiliarity with non-linear behaviour*: Users are mainly linearly reasoning. Non-linear reasoning should be supported in a specific form.
- *Additional and supplementary elements without meaning*: Diagrams of use elements which do not have a unique or any meaning, e.g. colours, shapes, grid forms for lines etc.
- *Hidden dimensions within the diagram*: Diagrams cannot reflect all aspects although there are essential ones, e.g. time.
- *Representation as fine and visual art*: Finding a good representation is a difficult task and should be supported by a culture of modelling.

All these obstacles are observed in the case of BPMN diagrams [10].

Diagrams must be developed on the principles of visual communication, of visual cognition and of visual design [12]. The culture of diagramming is based on a clear and well-defined design, on visual features, on ordering, effect, and delivery, and on familiarity within a user community.

One of the main obstacles of diagrams is the missing abstraction. The simplest way to overcome it is the development of a model suite [19] consisting of a generic model and its refinement models where each of them is adequate and dependable. Generic models [31] reflect the best abstraction of all models within a model suite.

4 Evaluation of the BPMN Approach

BPMN is a powerful diagrammatic languages that uses more than 100 modelling elements. The same situation in the reality or the implemented system can be specified by a variety of diagrams. Since a theory of diagram equivalence is missing, [27] introduced seven evaluation methods for models:

- PURE – SMART – CLEAR evaluation for the goal-purpose-function evaluation of an instrument in a given application context, for given artifacts to be represented, for a given community of practice, and for a given profile (goal, purpose, and function) under consideration of the utilisation scenarios;
- PEST evaluation for assessment of internal, external, and quality of use;
- QUARZSAND evaluation for assessment of the model development, and
- SWOT – SCOPE evaluation for description of the potential of the model, i.e. the general properties of a given instrument or the modelling method.

Since we did not explore the directives in detail nor the adequateness and dependability of an instrument that is a candidate model, we concentrate on the last two methods in this section. The evaluation of adequacy and dependability has been developed in [28]. We concentrate here on the capacity and potential of the BPMN approach.

4.1 The Capacity of the BPMN Approach

Capacity is a strategic measure whereas the potential is a tactical one. The potential can be used to derive the added value of a utilisation of a model within a given scenario. The potential allows to reason on the significance of a model within a given context, within a given community of practice, for a given set of origins, and within the intended profile.

The capacity relates an instrument to utilisation scenarios or the usage spectra. We answer the questions whether the instrument functions well and beneficial in those scenarios, whether it is well-developed for the given goals and purposes, whether it can be properly, more focused, comfortably, simpler and intelligible applied in those scenarios instead of the origins, and whether the instrument can be adapted to changes in the utilisation. The answers to these questions determine the main content or cargo, the comprehensiveness, and the authority or general value of a model. Another important aspect is the solution-faithfulness of the instrument. The capacity is an essential element of the model cargo, especially of the main content of the model.

BPMN diagrams can be used in description, prescription, explanation, documentation, communication, negotiation, inspiration, exploration, definition, prognosis, reporting and other scenarios. We discover that communication, negotiation, and inspiration are supportable. Description, prescription, and definition can be supported if the BPMN diagrams are enhanced and a precise semantics of all BPMN elements is commonly used in all four kinds of diagrams. The adequacy and especially the analogy to the origins (i.e. storyboards or business processes) is assumed to be based on homomorphy what is rarely achieved. This homomorphy is suitable if all processes are completely and in detail specified and all variations and exceptions are consistent.

The general utility of BPMN diagrams becomes rather low if the specific background of the modelling approach is not taken into consideration. BPMN diagrams are process-oriented, based on an orthogonal separation of flow element into activities, gates, and events, differentiate actors within their roles, and support communication among actors based on message exchange. The execution semantics is based on a token interpretation of control flow. Actors are isolated in their execution if binding is not done through message exchange or implicit hidden resource conditions. Data and resource are however local. All processes are potentially executed in parallel. The local-as-design approach might be appropriate if business processes are not intertwined. The concentration on the same abstraction level restricts the applicability of BPMN modelling. Generic workflows [31] provide a solution to this limitation.

4.2 The Potential of the BPMN Approach

The potential describes the (in-)appropriateness of a modelling approach within the directives. The suitability of BPMN diagrams depends on whether the application and the context support the local-as-design approach, on whether the demands of the community of practice can be satisfied, on whether the instrument is adequate (analogous, focused, purposeful), on whether the goals can be achieved with the given instrument, on the fruitfulness of the instrument compared with other instruments, and on the threats and obstacles of utilisation of BPMN diagrams.

4.3 The SWOT Evaluation of the Potential

The SWOT analysis is a high-level method that allows to evaluate the general quality of an instrument and its general assumptions of deployment.

Strengths. The BPMN approach is standardised and uses a large variety of constructs. It thus allows development of detailed models. It has a high expressibility. Both intra- and inter-organisational aspects can be represented. The approach is well supported by tools.

Weaknesses. The large variety of competing elements is also a weakness. The complexity and integration of diagrams may cause solution-unfaithfulness. The language requires high learning efforts. Processes that are dynamic at runtime cannot be modelled. Exchange among tools is an open problem.

Opportunities. Most business processes can be adequately described due to the variety of elements. The standardisation provides at least a base semantics.

Threats and Risks. None-technical users might be unable to cope with diagrams. Work-arounds hinder comprehensibility. Vendors define their own extensions. The BPMN standard does not completely define the execution.

4.4 The SCOPE Evaluation of the Potential

The SCOPE analysis of a model embeds the model into the application context, refines the capacity evaluation of an instrument, and considers the community of practice and their specific needs.

Situation. BPMN diagram suites provide some kind of formalisation of business processes. Communication is specified to a certain degree. Control flow is well-represented.

Competence. BPMN diagrams must be combined with other models since the other four aspects (organisation, data flow, functions, operational assignment) are only partially reflected.

Obstacles. Typical challenges of BPMN modelling are the specification complexity, diagram coherence, exception handling, and the development of an execution semantics. There is no common agreement on well-formedness of diagrams.

Prospects. A separate BPMN diagram is easy to read and to interpret.

Expectations. The BPMN approach can be combined with local-as-design-oriented conceptual data models, storyboards, business rule specification and other modelling approaches as one kind of models within a model suite.

4.5 The Resulting Potential of the BPMN Approach

The BPMN diagram has a high potential for communication and negotiation utilisation scenario. The potential for system construction within a description-prescription scenario is however rather limited due to missing co-design support. A similar inappropriateness can be stated for explanation, prognosis, exploration, definition, and reporting scenario. The potential within a documentation scenario is rather small. The highest potential of the BPMN approach can be however observed for inspiration scenario. The process, choreography, conversation, and collaboration diagrams are an appropriate means for an implementation plan based on inspiring diagrams.

Similar to SPICE assessments [7], we may rate maturation of a model and a modelling approach to: (0) ad-hoc , (1) informal, (2) systematic and managed, (3) standardised and well-understood, and (4) optimising and adaptable, and (5) continuously improvable styles. The evaluation shows that the BPMN approach has not yet reached level (2). This observation leads us to the conclusion that PURE-SMART-CLEAR and PEST evaluations are heavily dependent from the directives for BPMN diagram modelling.

A model must be of high utility, must have a high added value, and should have a high potential. These parameters also depend on the well-formedness of the instrument. The BPMN approach can be enhanced by criteria for well-formedness for syntactical, semantical and pragmatical well-shaped diagrams [28].

5 Towards a Theory of Modelling

5.1 Models Burdened with Directives and Background

The directives and the background (see Figure 2) heavily influence the way how a model is constructed, what is taken into consideration and what not, which rigidity is applied, which basis and grounding is taken for granted, and which community of practice accepts this kind of model.

The model incorporates these influences without marking them in an explicit form. The model is laden or *burdened* by these decisions. Additionally, models are composed of elements that are selected, changed and adapted within a development process. Figure 5 depicts elements of this burden and this development history.

5.2 The Anti-Profile of an Instrument as a Model

We may now directly conclude that an instrument might or might not be adequate and dependable for any utilisation scenario due to its insufficiency to function in this scenario.

Definition 10. *A* utilisation pattern *of an instrument describes the form of usage of an instrument, the discipline of usage, the applications in which the instrument might be used, and the conditions for its utilisation.*

Definition 11. *A* utilisation scenario *consists of a utilisation pattern and a number of functions a specific instrument might play in this utilisation pattern.*

Definition 12. *A* usage spectrum *consists of collection of utilisation scenarios. A* portfolio *of an instrument combines the usage spectra.*

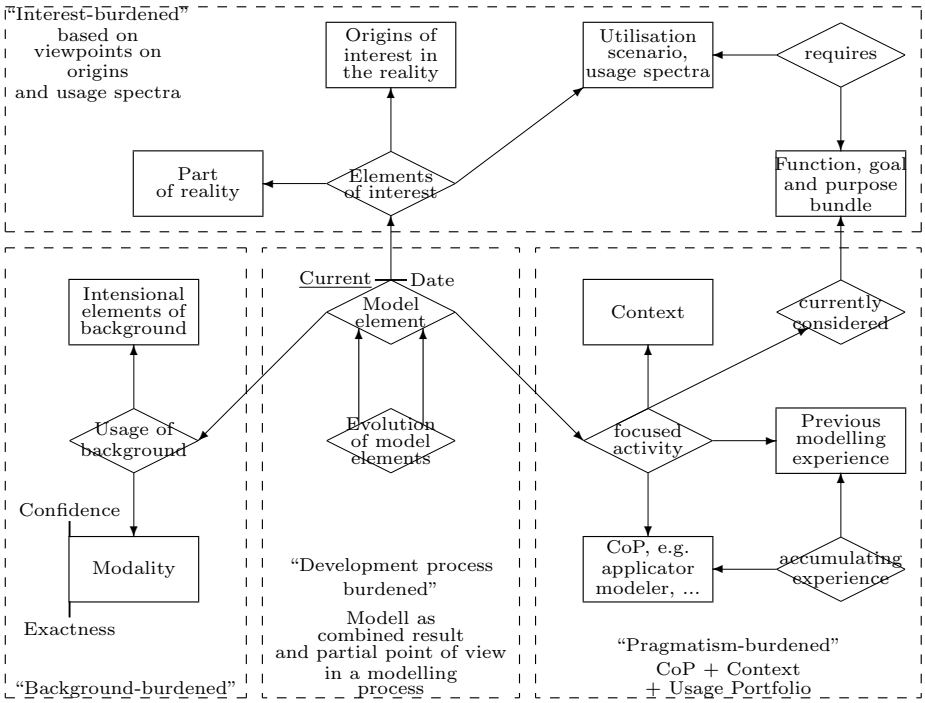


Fig. 5. Models are burdened by their development history, the background and the directives

Definition 13. *A* profile *of an instrument as a model consists of the goals, the purposes, and the functions of the instrument within a portfolio.*

We can now roughly describe an anti-profile of a model and resulting *utilisation proscriptions* of a an instrument as a model by answering the following questions:

– For which scenarios is the instrument useless?

– In which of the following scenarios efficiency and effectiveness is not given for the instrument: description & prescription, realisation & coding, theory development, theory refinement, causality consideration, inexplicability, demonstration, prediction, explanation, mastering of complexity, understanding, or ... ?
– Are essential parameters of the origins missing? Are some of the essential parameters only represented via mediating or dependable parameters? Are there dummy or pseudo dependences among the parameters?
– What cannot be adequately represented? Is the dependability really sufficient? In which case users need a special understanding and education? Which tacit knowledge is hidden in the instrument?
– In which cases the instrument cannot be effectively used? What must a user respect and obviate before using the instrument?
– Which biases and which background are palmed off? Which assumptions, postulates, paradigms, and schools of thought are hidden and not made explicit? Models might condition conclusions.

Since models are instruments their utilisation conditions conclusions and results. Therefore, it is appropriate to describe the anti-profile of a model as well.

5.3 Questions to Answer Before Using an Instrument as a Model

The rhetoric frame and its extension to the W*H frame [5] can now be used for derivation of questions one must answer before using the model:

– What is the function of the model in which scenario? What are consequential purposes and goals? What are anti-goals and anti-purposes?
– Which origins are going to be represented? Which are not considered? Does the model contain all typical, relevant and important features of the origins under consideration and only those?
– Is the instrument adequate and dependable within the utilisation scenario? What are the parameters for adequacy and dependability? How purpose-invariance and solution-faithfulness is going to be defined?
– What kind of reasoning is supported? What not? What are the limitations? Which pitfalls should be avoided?
– Do you want to have a universal model that contains all and anything? Would it be better to use a model suite where each of the models represent some specific aspects? What about the nonessential aspects?

6 Conclusion

A general understanding of the notion of a model has been already started with development of Computer Science. Milestones are the papers and books by H. Stachowiak (1980ies and 1990ies), B. Mahr (2000ies until 2015), W. Steinmüller (1993), and R. Kaschek (since 1990ies) [9, 11, 17, 18, 21]. These notions treat models from a phenomenological point of view through properties that a model should have (e.g. as main properties: mapping or analogy, truncation, pragmatic

properties). We need however also an explicit definition of the notion of model. Such general notion has been developed in a series of papers, e.g. [22, 24, 25, 27, 29].

The model notion is universal one and based on two parameter sets for adequateness and dependability. The parameter sets seem to be complex and need a methodological support. This paper develops such a support facility based on the notion of a conception frame. The practicality of the approach is demonstrated for the workflow specification language BPMN. BPMN shares the positive treatment with most other formal or informal languages in Computer Science. The capacity and thus the restrictions or obstacles are not explicitly communicated. We see however that the evaluation, capacity, potential, and capability can be explicitly provided based on our approach.

Since the model notion is a mathematical definition, it seems to be achievable to develop a theory of modelling in the sense of a theory. In this paper, we only discuss two components of such a theory: the explicit description of the background of models and the anti-profile. The conception frame for the model definition may also be used for derivation of question forms that a modeller can use before delivering an instrument as a model to a community of practice. The development of a full theory is however a research issue for the next decades.

References

1. E. Börger, O. Sörensen, and B. Thalheim. On defining the behavior of or-joins in business process models. *Journal of Universal Computer Science*, 15(1):3–32, 2009.
2. E. Börger and B. Thalheim. A method for verifiable and validatable business process modeling. In *Software Engineering*, LNCS 5316, pages 59 – 115. Springer, 2008.
3. E. Börger and B. Thalheim. Modeling workflows, interaction patterns, web services and business processes: The ASM-based approach. In *ABZ*, volume 5238 of *Lecture Notes in Computer Science*, pages 24–38. Springer, 2008.
4. S. Chadarevian and N. Hopwood, editors. *Models - The third dimension of science*. Stanford University Press, Stanford, California, 2004.
5. A. Dahanayake and B. Thalheim. *Correct Software in Web Applications and Web Services*, chapter W*H: The conceptual Model for services, pages 145–176. Texts & Monographs in Symbolic Computation. Springer, Wien, 2015.
6. L. Fleck. *Denkstile und Tatsachen, edited by S. Werner and C. Zittel*. Surkamp, 2011.
7. ISO/IEC. Information technology - process assesment. parts 1-5. IS 15504, 2003-2006.
8. ISO/IEC. International organization for standardization: Information technology object management group: Business process model and notation. http://www.omg.org/spec/BPMN/ISO/19510/PDF/, 2013.
9. R. Kaschek. *Konzeptionelle Modellierung*. PhD thesis, University Klagenfurt, 2003. Habilitationsschrift.
10. F. Kossak, C. Illibauer, V. Geist, J. Kubovy, C. Natschläger, T. Ziebermayr, T. Kopetzky, B. Freudenthaler, and K.-D. Schewe. *A Rigorous Semantics for BPMN 2.0 Process Diagrams*. Springer, 2014.
11. B. Mahr. Information science and the logic of models. *Software and System Modeling*, 8(3):365–383, 2009.

12. T. Moritz. *Visuelle Gestaltungsraster interaktiver Informationssysteme als integrativer Bestandteil des immersiven Bildraumes*. PhD thesis, HFF Berlin-Babelsberg, 2006.
13. I. Nissen and B. Thalheim. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*, chapter Bedeutung, Entwicklung und Einsatz, pages 3–28. De Gryuter, Boston, 2015.
14. OMG. Object management group: Business process model and notation (BPMN) 2.0. http://www.omg.org/spec/BPMN/2.0, 2011.
15. H. Pichler and J. Eder. Business process modeling and workflow design. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 8, pages 259–286. Springer, Berlin, 2011.
16. J.E. Safra, I. Yeshua, and et. al. *Encyclopædia Britannica*. Merriam-Webster, 2003.
17. H. Stachowiak. Modell. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon zur Wissenschaftstheorie*, pages 219–222. Deutscher Taschenbuch Verlag GmbH & Co. KG, München, 1992.
18. W. Steinmüller. *Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik*. Wissenschaftliche Buchgesellschaft, Darmstadt, 1993.
19. B. Thalheim. *The Conceptual Framework to Multi-Layered Database Modelling based on Model Suites*, volume 206 of *Frontiers in Artificial Intelligence and Applications*, pages 116–134. IOS Press, 2010.
20. B. Thalheim. Towards a theory of conceptual modelling. *Journal of Universal Computer Science*, 16(20):3102–3137, 2010. http://www.jucs.org/jucs_16_20/towards_a_theory_of.
21. B. Thalheim. The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In *The Handbook of Conceptual Modeling: Its Usage and Its Challenges*, chapter 17, pages 547–580. Springer, Berlin, 2011.
22. B. Thalheim. The science and art of conceptual modelling. In A. Hameurlain et al., editor, *TLDKS VI*, LNCS 7600, pages 76–105. Springer, Heidelberg, 2012.
23. B. Thalheim. The conception of the model. In *BIS*, volume 157 of *Lecture Notes in Business Information Processing*, pages 113–124. Springer, 2013.
24. B. Thalheim. The conceptual model ≡ an adequate and dependable artifact enhanced by concepts. In *Information Modelling and Knowledge Bases*, volume XXV of *Frontiers in Artificial Intelligence and Applications, 260*, pages 241–254. IOS Press, 2014.
25. B. Thalheim and A. Dahanayake. A conceptual model for services. In *Invited Keynote, CMS 2015, ER 2015 workshop*, LNCS 9382, pages 51–61, Berlin, 2015. Springer.
26. B. Thalheim and I. Nissen, editors. *Wissenschaft und Kunst der Modellierung*. De Gruyter, Ontos Verlag, Berlin, 2015.
27. B. Thalheim and I. Nissen. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*, chapter Ein neuer Modellbegriff, pages 491–548. De Gryuter, Boston, 2015.
28. B. Thalheim and I. Nissen. *Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung*, chapter Fallstudien zum Modellbegriff, pages 549–602. De Gryuter, Boston, 2015.
29. B. Thalheim and M. Tropmann-Frick. The conception of the conceptual database model. In *ER 2015*, LNCS 9381, pages 603–611, Berlin, 2015. Springer.
30. M. Thomas. Modelle in der Fachsprache der Informatik. Untersuchung von Vorlesungsskripten aus der Kerninformatik. In *DDI*, volume 22 of *LNI*, pages 99–108. GI, 2002.

31. M. Tropmann-Frick, B. Thalheim, D. Leber, G. Czech, and C. Liehr. Generic workflows - a utility to govern disastrous situations. In *Information Modelling and Knowledge Bases*, volume XXVI of *Frontiers in Artificial Intelligence and Applications, 272*, pages 417–428. IOS Press, 2014.

Open Problems of Information Systems Research and Technology

Bernhard Thalheim

Christian Albrechts University Kiel, Department of Computer Science, D-24098 Kiel, Germany;
thalheim@is.informatik.uni-kiel.de

Abstract. Computer science and technology is an area of very intensive research and development. It is estimated that the half-life period of knowledge in this area is about 12 to 18 months. No other branch of science has such short half-life period. Therefore new problems have to solved. At the same time some of the old open problems must be solved. This invited talk aims at a systematic survey of open problems and proposes a number of solutions to their solution. We structure these open problems in structuring problems, into size or more generally complexity problems, functionality problems, interactivity problems, distribution problems, and general problems.

1 Computer Science and Technology

Let us first consider the primary sources for open problems of information systems research and technology. They are mainly caused by the technology itself and stream of research in this area. They are also caused by the evolution of information systems. Another obstacle are the notions we use differently in different settings. Computer Science and Technology (CS&T) is a young branch of science and technology. It took hundred if not thousands of years for other sciences to become matured. Mathematics, for instance, can be based on three main principles: topology, order, and algebra. Social sciences use the principles individual, development, and society. CS&T may however also be based on four main principles in Figure 1: structuring, evolution, collaboration and modelling. Each of these principles has a number of aspects. Structuring is either structuring in the

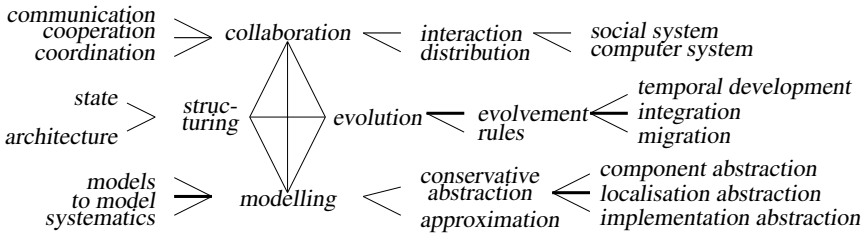


Fig. 1. The four principles of Computer Science and Technology

large, i.e., architecture, or in the small that is based on the notion of the state. Evolution

can also be either evolution in the small which is based on rules for state transformation or evolution in the large with its temporal, integration or migration aspects aspects. Collaboration is far less well understood. Collaboration services can be build based on the 3C framework [5] that separates supporting services into communication services, coordination services, and cooperation services. It is either on interaction between social systems and computer systems or on distribution among systems. Modelling is far less well understood although more than 50 different kinds of models are known for CS&T [12]. It is however possible to develop a general notion of the model [10], of the activity of model development [8] and systematic methodology backed modelling [9].

Information, data and knowledge are often used as synonyms. The separation into data, information and knowledge is displayed in Figure 2. For the definition of notion of knowledge we refer to [11]. There are several definitions for information. Information

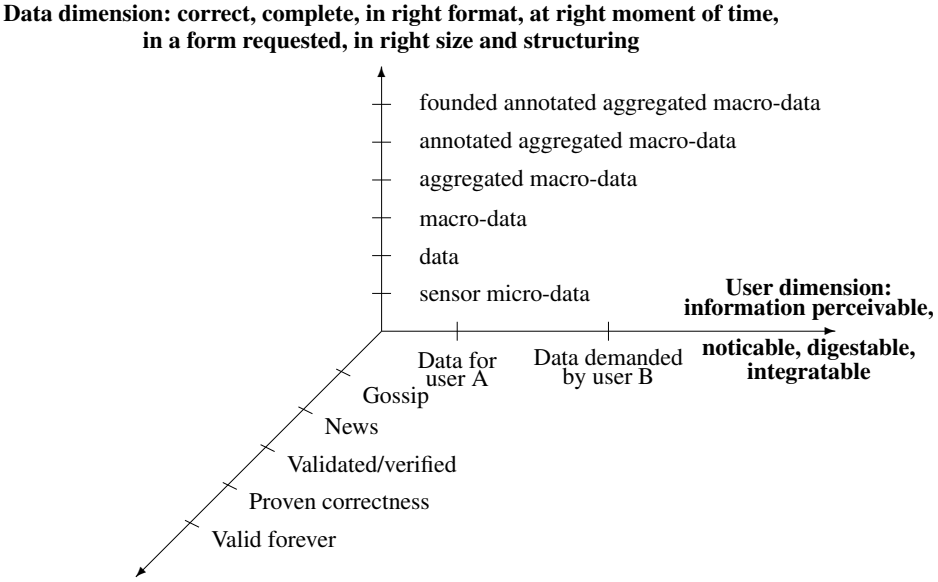


Fig. 2. Three dimensions of semiotics: Syntax by data, semantics by knowledge, and pragmatics by information

is defined in [2]. as raw data that is well-formed and meaningful data and that has been verified to be accurate and timely relative to its context, is specific and organised for a purpose, is presented within a context that gives it meaning and relevance, and which leads to increase in understanding and decrease in uncertainty. The second definitions is based on the mathematical notion of entropy. The third definition bases information on the data a user has currently in his data space and on the computational and reasoning abilities of the user. Business information systems understand information as data that have been shaped into a form that is meaningful, significant and useful for

human beings. If we consider information systems as an element of social systems then we can use the anthroposophic form: Information as processed by humans, is carried by data that is perceived or noticed, selected and organised by its receiver, because of his subjective human interests, originating from his instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom, simultaneously processed by his cognitive and mental processes, and seamlessly integrated in his recallable knowledge. The separation into data, information and knowledge is displayed in Figure 2.

Computer science and technology has four sources: mathematics, electronics, engineering and applications. The first two sources are well acknowledged. The scientist adds to the store of verified, systematised knowledge of the physical or virtual world. The engineer brings this knowledge to bear on practical problems[1]. The software engineering triptych [3, 1] consists of the application domain description, the requirements prescriptions, and finally the systems specifications. Requirements must be reasonably well understood before software can be designed, programmed, and coded. The application domain must be reasonable understood in all its dependencies, weak and sufficient properties before requirements can be expressed properly. Software engineering classically considers only the second and the third dimension. Application domain engineering has not yet got its foundations.

2 Open Problems

In the sequel we list main open problems. The suggestions, hypotheses and proposals for their solution are given in the lecture and in an extended variant of this paper. We first start with a list of problems that is now over 25 years old [6, 7].

2.1 The MFDBS List

(MFDBS1) Satisfiability of specification: Given a structure specification of a database, e.g., a schema with integrity constraints. Is this specification by a finite, non-empty database.

(MFDBS2) Integration of static and operational constraints: Information systems use both static and operational constraints. Are there systems that allow a coherent treatment of both kinds of constraints?

(MFDBS3) Strong and soft interpretation of constraints: Integrity constraints are often only considered in their strong logical semantics. We might however use also natural semantics and allow systematic exceptions of validity of these constraints in a database. Is there any theory that can handle this?

(MFDBS4) Global normalisation: Classical normalisation theory treats each type in a separate form despite the association of these types within a schema. Is there a mechanism to integrate normalisation for global handling?

[1] "Scientists look at things that are and ask 'why'; engineers dream of things that never were and ask 'why not'." (Theodore von Karman) [4]

(MFDBS5) Continuous engineering and consistent extensions: Database systems are constantly evolving, are extended, are integrated with other systems. How we may support this kind of evolution?

(MFDBS6) Integration of quality requirements into specification: Software engineering knows more than 40 different characteristics of quality of a software product. The same applies to information systems. How these quality characteristics can be given in a coherent and manageable form?

(MFDBS7) Complexity of constraint sets: Constraint set complexity has so far mainly considered only for simple classes of constraints. Develop an approach for complexity characterisation of constraint sets!

(MFDBS8) Enforcement of integrity constraint sets: Constraints are declared within a formal language. They must be enforced within a database system in both declarative and procedural way. Develop a formal theory of enforcement of integrity constraints that allows to enforce constraints either procedurally or at the interface level!

(MFDBS9) Implication problems on the level of kinds of constraints: The logical handling of integrity constraints is still based on a strict separation of these constraints according to their kind. Practical applications are however using functional, inclusion, join, exclusion, cardinality etc. constraints all together. Develop an approach that allows to manage constraints without a separation into kinds of constraints!

(MFDBS10) Treatment of semantics by views: Database technology uses still a local-as-view approach and specifies views on top of the logical or conceptual schemata. Applications use however views and onyl in very rare cases the global schema. Develop an approach for management of semantics at the view level!

(MFDBS11) Distributed integrity management: Is there any mechanism to manage integrity constraints in distributed environments?

(MFDBS12) Integration of vertical, horizontal and deductive normalisation: Classics considers only vertical normalisation. how to incorporate horizontal and deductive normalisation in this management?

(MFDBS13) Treatment of incomplete specifications: Specifications are normally incomplete. How to handle this incompleteness?

2.2 Open Problems: Technology

(TT1) Partial identifiability: Objects can be partially depending on the profile and portfolio of their utilisation. Develop a treatment of this identification!

(TT2) Query optimisation considers so far relational algebra. Develop an extension that allows aggregation, grouping, ordering!

(TT3) Inheritance is defined at design, code, or decision layers. Develop a layer-independent approach to inheritance.

(TT4) Maintenance optimisation: Systems provide their own mechanism for maintenance, e.g. time/mode/strictness of integrity maintenance. Develop a general approach to maintenance optimisation!

(TT5) View towers: Typical information system applications based on the local-as-view approach use towers of views. Develop a technology for view tower handling, e.g., with partial local enforcement!

(TT6) Component systems support industrial production-line development of information systems. Their technology needs sophisticated collaboration support.

(TT7) Quality management for distributed information systems: Information systems are often not entirely correct or complete or up-to-date. We need however a supporting quality management for these systems.

2.3 Open Problems: Co-Design

Modern information systems development is based on development of structuring, on functionality development, and on development of interaction features. Additionally, systems are distributed. Therefore, co-design and co-evolution of systems becomes a major feature.
Co-design is still under development. The current state of the art is challenging. Without conceptual models we face late specification, inflexibility, and un-maintainability of systems. Extension, change management and integration become a nightmare.

(C1) Coherence of models: Co-design can be based a separation of concern and on representation by various models. A model suite must however be based on coherence conditions and associations.

(C2) Compiler-based transformation of conceptual schemata: Conceptual schemata are often directly mapped by an interpreter. The logical schema is later optimised. We may however develop a compiler approach beyond the interpretation or rule-based approach.

(C3) Semantics treatment: Classical semantics of databases and information systems uses the strict approach of mathematical logics. It is often inappropriate. Therefore, we need a flexible treatment of semantics, e.g., depending on the kind of constraints or sets of constraints.

(C4) Global-as-view schemata: The classical information system architecture is based on a local-as-view approach. Develop new approaches opposite to classical local-as-view 2/3-layer architectures!

(C5) Object-relational design starts with an OR-schema (e.g., HERM-schema) and ending with a object-relational schema of modern DBMS. Develop novel design methods for OR logical schemata.

(C6) Content services: Information services are becoming one of the major technologies in the web. They are based on content delivery to different users and content extraction from various resources. Develop a faithful content service on top of database systems.

(C7) Faithful mapping from and to UML: UML is becoming a communication language for systems development. It does however not provide a holistic semantics for its diagrams. Brute-force interpretation of diagrams is the current state of art. Develop a faithful mapping that allows to use collections of diagrams of different kinds.

2.4 Open Problems for Structuring

(S1) Conceptual modelling in the large is partially as extension of conceptual modelling in the small. Develop additional solutions for modelling in the large.

(S2) List semantics for (extended) entity-relationship modelling: ER models are typically interpreted based on set semantics. Sometimes ER schemata are also mapped to XML schemata. We need however an approach that supports mapping to list-based languages.
(S3) Centre-periphery integration into schemata: Classical information system modelling treat the schema as a holistic solution. Schemata have however an inner structure. Especially they have central or kernel types and peripheric types. This separation is based on overlay techniques. Develop a technique for centre-periphery separation within a schema.
(S4) Level of locality and globality: Beside global as view and local as view models we might also look for compromises between local-centric or global-centric schemata. Develop structuring with the best globality.
(S5) Null marker logics: Null markers (inappropriately called values) carry a variety of different application semantics. Develop techniques for management of these markers beyond the existence, non-applicability and unknown.
(S6) Pattern of structures: Information system development can be based on experience and skills of developers. They use general solutions and pattern for their work. Develop pattern beyond the small patterns of Blaha.
(S7) Redundant schemata: Schemata need redundant types. Develop an approach for explicit maintenance of redundancy of types and data in information systems.

2.5 Open Problems: Constraints

(TIC1) Complexity: The complexity of constraint sets is only known for some basis kinds of constraints. Develop a complexity theory for real life constraint sets! Develop an approach to average complexity of such constraint sets!
(TIC2) Incomplete constraint sets: Normalisation requires complete constraint knowledge. Develop an approach for incomplete knowledge.
(TIC3) Denormalisation: Normalisation is lucky break. Typical applications need however well-settled denormalisation as well. Develop a definition, treatment, algorithmics for denormalisation.
(TIC4) Global normalisation: Develop an approach to global schema normalisation beyond classical local vertical normalisation.
(TIC5) Partial axiomatisation: Incomplete constraint sets are the normal case for specification. Therefore we need an approach for deductive systems that provide a partial axiomatisation.
(TIC6) Graphical reasoning has shown to be useful for handling, reasoning and management of functional and multivalued dependencies. Develop graphical reasoning systems for other constraint sets.
(TIC7) Real-life constraint sets: Constraints cannot be handled only within its specific kind. We need techniques for constraint handling outside the orientation to kinds.

2.6 Open Problems: Functionality

(F1) (e)ER-SQL: SQL is becoming the main database language and is currently going to be combined with NoSQL features. (e)ER-SQL can be graphically developed based on VisualSQL. Develop an integration of VisualSQL with NoSQL features!

(F2) Holistic functionality description: Functionality specification is often using different languages for the application domain, for business processes and for conceptual functionality specification. We need however a holistic specification technique.
(F3) Dynamic semantics is still a step-child of integrity constraint handling. Develop an appraoch to dynamic semantics beyond transition rules that reflect static semantics.
(F4) Robust workflows: Workflows often use exception handling and try to reflect any potential case. This technique is infeasible for most processes. Instead develop an engineering approach that handles errors and failures without requesting a complete specification.
(F5) Flexible workflows: Workflows typically specify a hard-coded control flow. Life is however more flexible. Develop techniques for controllable deviations from flow and coherent finalisation.
(F6) Information systems functions: Information systems are often entirely based on database techniques. Users need however also functions on their own beyond retrieval and state change operations.
(F7) Generic views: View tower development may result in hundreds of almost unmanageable views. Develop a view derivation technique similar to generic functions.

2.7 Open Problems: Algebra

(TA1) Transaction semantics for higher SQL: SQL:1999, SQL:2003, and SQL: 2007 provide extended and sophisticated techniques for transaction handling. We need a holistic management for such transactions.
(TA2) Spatial operations have been for more advanced spatial data structures. We need however an algebra that allows to compute also spatial data.
(TA3) Program transformation: Database programs can be specified at the conceptual level. We lack however program transformation techniques.
(TA4) Greatest consistent specialisation: The GCS approach allows to derive specialisations of operations that internally maintain integrity constraints. Develop GCS techniques beyond functional and inclusion constraints!
(TA5) Trigger assembly creation: Triggers are currently linear static programs. They need very sophisticated development techniques and a deep understanding of their impact. Develop techniques for trigger assemblies.
(TA6) Transformation of expressions: Queries can be simplified if we know integrity constraints. We need a systematic transformation of expressions in the presence of IC.
(TA7) Extension of structural recursion: Structural recursion is the main definition technique for algebraic operations. We need however also structural recursion for holistic expressions.

2.8 Open Problems: Distribution

(D1) Partial consistency of databases: Distributed databases and information systems follow a variety of consistency paradigms. We need however also techniques that support collaboration of systems based on specific contracts and protocols.

(D2) Recharging of partner databases: Databases may collaborate based on pattern such as publish-subscribe. Develop a data integration technique depending on subscription mode of partner databases.
(D3) Coordination: The 3C approach to collaboration uses coordination as a central conception. Develop techniques for coordination beyond contracts.
(D4) General model of services: One of the most overused notions in CS&T is the service as specific software with support and with compliance. Develop a general model for services.
(D5) Exchange frames: The 3C framework uses techniques of communication similar to protocol engineering. It depends on the chosen architecture. Exchange frames are typical supporting means. They are given in a very technical way and thus depend on the underlying firmware. Develop a general technique for exchange frames.
(D6) Component database systems: with specific coupling facilities, collaboration contracts
(D7) Pattern of distributed systems: Pattern have been developed for specific integration in software engineering research. Information systems are more specific. Therefore, we need a specialisation of these pattern to information and database systems and specific pattern for these systems.

2.9 Open Problems: Interactivity

Interactivity is nowadays the main concern for web information systems. Their technology, their specification and their theory are not yet systematised. They will however have the same fate as classical information systems: they will stay with us longer than it has been expected when they have been developed. At the same time they face the same problems as we have already observed for human-computer interfaces.

(I1) Edutainment stories: One main kind of web information systems are edutainment systems (often called e-learning systems). Develop techniques and solutions for edutainment beyond classical blended learning and towards collaboration and true interaction.
(I2) New stories: In the app age we face a large variety of solutions based on small components. Their integration and coherent deployment is still a challenge to the user. It seems that they are partially arbitrarily combinable and thus only integrateable by a human user. There are however main deployment pathes based on mini-stories with data coherence. Develop a technology for such systematic treatment of new stories.
(I3) Screenography is a generalisation of scenography and dramaturgy. It supports systematic development of screens and allows an adaptation to the user. Develop an approach integration of screenography into conceptual modelling.
(I4) Life case bakery: System development is still based on packages that provide a fully fledged solution to a collection of application problems. Real life is however based on life event and life situations. Life cases may reflect both. They can be mapped to business use cases and business stories which are the basis for requirements prescription. We need techniques for continuous adaptation to the specific life event or life situation.

(I5) I^***-generalisation:** The I^* model allows to model the intentions, desires, beliefs and goals of users. Goals can also be soft goals. These specification techniques for (Soft)Goals ∪ Tasks ∪ Actors ∪ Resources can be generalised to the story spaces, obligations and life cases.

(I6) Privacy of users: Privacy of users becomes a major bottleneck of the 21^{st} century. We currently lack techniques privacy profile, controlled opening of shared data, flexible protection etc.

(I7) Workspace integration of users: Users have their own systems with their own workrooms, workspace and libraries. These systems vary a lot and cannot be generalised to some holistic system environment. Instead we need techniques for flexible integration of user workspaces into current information systems.

References

1. D. Bjørner. *Software Engineering 3: Domains, requirements, and software design*. Springer, Berlin, 2006.
2. G.M. Greco, G. Paronitti, M. Turilli, and L. Floridi. The philosophy of information: A methodological point of view. In *Wissensmanagmeent*, pages 563–570. DFKI, Kaiserslautern, 2005.
3. L. J. Heinrich. *Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur*. Oldenbourg Verlag, München, 1996.
4. A. Samuel and J. Weir. *Introduction to Engineering: Modelling, Synthesis and Problem Solving Strategies*. Elsevier, Amsterdam, 2000.
5. K.-D. Schewe and B. Thalheim. Development of collaboration frameworks for web information systems. In *IJCAI'07 (20th Int. Joint Conf on Artificial Intelligence, Section EMC'07 (Evolutionary models of collaboration)*, pages 27–32, Hyderabad, 2007.
6. B. Thalheim. Open problems in relational database theory. *Bull. EATCS*, 32:336 – 337, 1987.
7. B. Thalheim. *Entity-relationship modeling – Foundations of database technology*. Springer, Berlin, 2000.
8. B. Thalheim. The art of conceptual modelling. In *Information Modelling and Knowledge Bases XXII*, volume 237 of *Frontiers in Artificial Intelligence and Applications*, pages 149–168. IOS Press, 2012.
9. B. Thalheim. The science and art of conceptual modelling. In A. Hameurlain et al., editor, *TLDKS VI*, number 7600 in LNCS, pages 76–105. Springer, Heidelberg, 2012.
10. B. Thalheim. The conception of the model. In *BIS*, volume 157 of *Lecture Notes in Business Information Processing*, pages 113–124. Springer, 2013.
11. B. Thalheim, Y. Kitawara, E. Karttunen, and H. Jaakkola. Future directions of knowledge systems environments for web 3.0. In *Information Modelling and Knowledge Bases*, volume 225 of *Frontiers in Artificial Intelligence and Applications*, pages 413–446. IOS Press, 2011.
12. M. Thomas. Modelle in der Fachsprache der Informatik. Untersuchung von Vorlesungsskripten aus der Kerninformatik. In *DDI*, volume 22 of *LNI*, pages 99–108. GI, 2002.