Journal of Universal Computer Science, vol. 16, no. 20 (2010), 3102-3137
submitted: 7/1/10, accepted: 13/9/10, appeared: 1/11/10 © J.UCS

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).

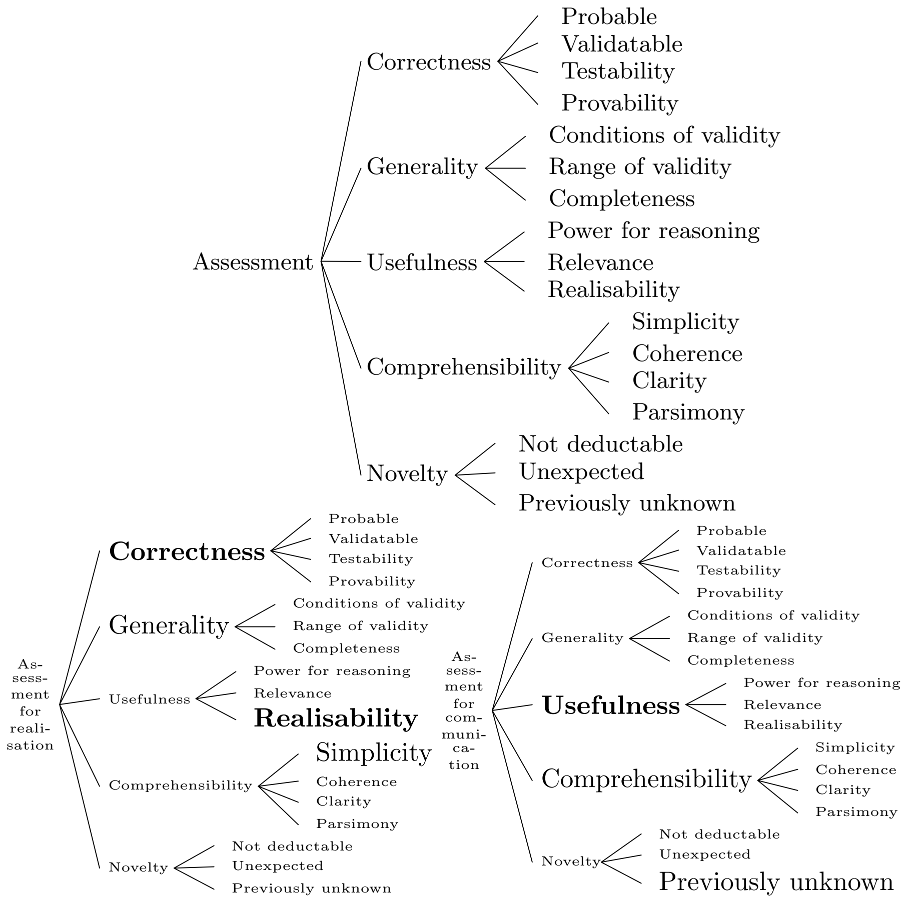

Figure 1: Assessment to quality depending on the purpose of the model: General assessment structure, assessment for realisation, and assessment for communication

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)].

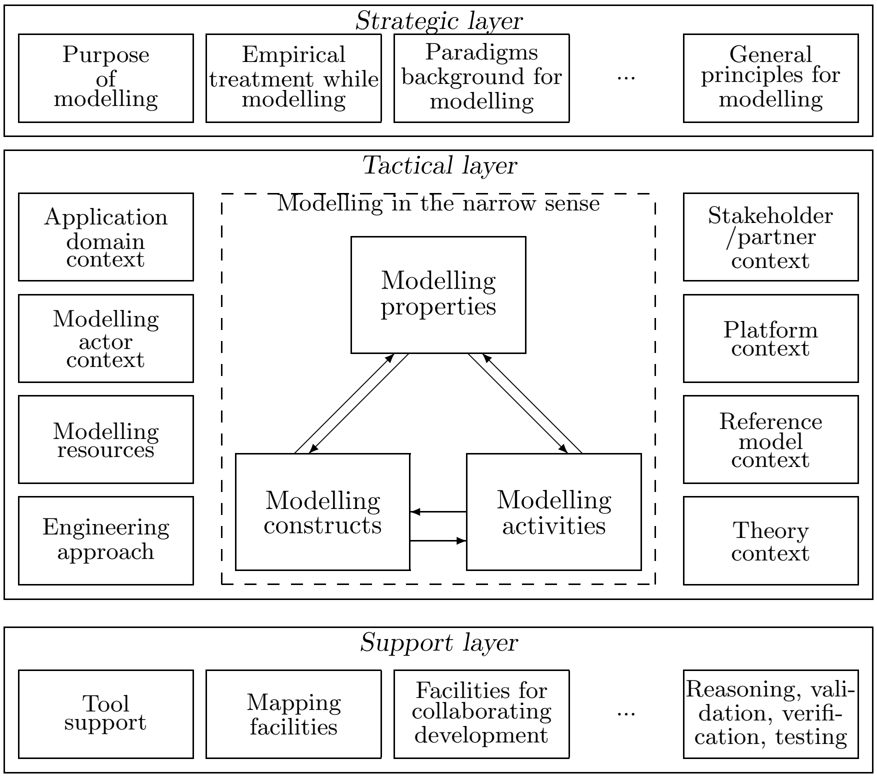

Figure 2: The strategic, tactical and support layers of modelling

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 sub-structures.

– 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 $\mathcal{T}_{\mathcal{L}_{\mathbb{S}}}$) 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.

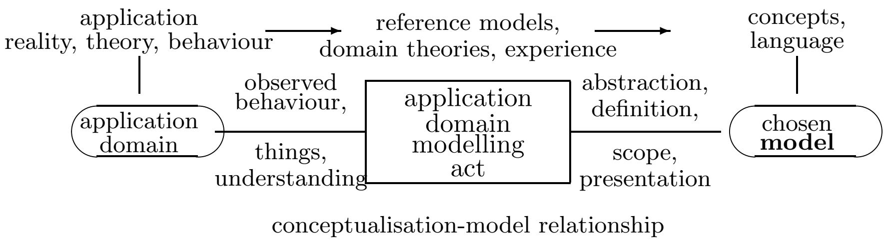

Figure 3: Conceptualisation dimensions of the application domain modelling act

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*,

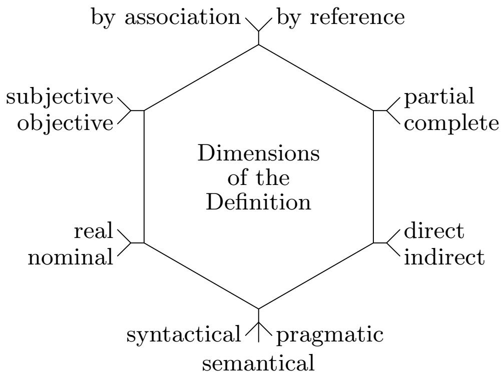

Figure 4: Ingredients for selection of the style and of the formulation used of stating a definition

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 *metamodel* 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,

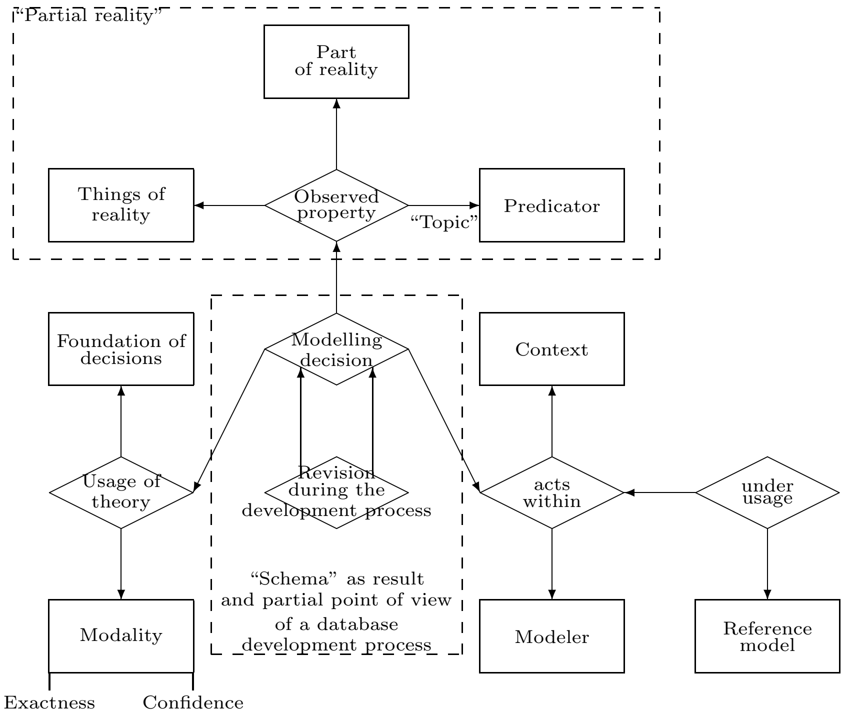

Figure 5: Modelling decisions made in the database development process

– *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.