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 and as well as management of models. 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.

1 Introduction

The main purpose of conceptual modelling is classically understood as the elicitation [3,6,10] 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

[3] and well understood [10] 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".

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. [1], the second dimension has not yet got a sophisticated and well understood support. The third dimension has received much attention by data modelers [8] but did not get through to research literature. It must therefore be our goal to combine the three dimensions into a holistic framework.

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*. 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. [11] 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.

Quality Assessment, Control and Improvement. According to [4] 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 [7]) used as a basis of the conceptualization process; in 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), internal quality (accuracy, suitability, interoperability, robustness, self-contained, independence), and quality of use (understandability, learnability, operability, attractiveness, appropriateness), and
– *dynamic qualities* within a selected development approach (executability, refinement quality, scope restriction, effect preservation, context explicitness, completion tracking).

The information system modelling process is intentionally or explicitly ruled by a number of development strategies, development steps, and development policies. Modelling steps lead to new specifications to which quality criteria can be applied. Typical quality criteria are completeness and correctness in both the syntactical and semantical dimensions. We assume that at least these four quality criteria are taken into consideration.

Outline and Tasks of the Paper. The paper aims in introducing a theory of conceptual modelling. We first describe modelling as a task. Then we show that a theory of conceptual modelling consists of a number of sub-theories that can be developed separately and that can be integrated. This approach leads to a framework for method development for conceptual modelling. We may develop a number of properties that allow to judge on the quality of the models. It is not our intention to present a fully developed theory. Instead we propose a path for the development of such a theory.

It has not been our intention to survey the modelling literature. This would be far too large already for the conceptual modelling research. Good sources to this research are [3,6,10].

Sample Application: *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. Classical approaches 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 well-known Petri net solution is depicted in Figure 1.

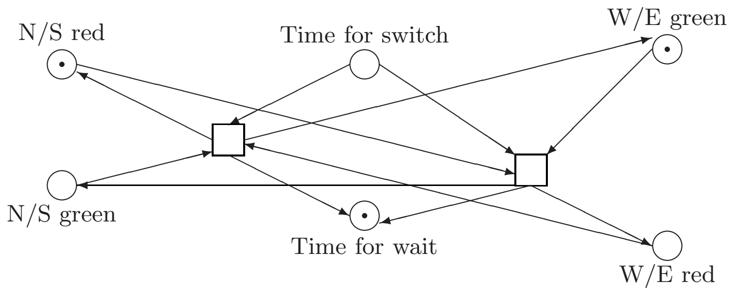

Fig. 1. Traffic control based on Petri nets

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

2 Modelling

The Notion of Model in Science Theory. 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).

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 [9] 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'.

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 Gap. 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" [5]. 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. To overcome the problems of specifications we may either use

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

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 [9] 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.

Sample Application. First we 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 and results in very complex models.

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 *NSgreenEWgreen* is a conflict state and thus not used for the model.

3 Constituents of a Theory of Conceptual Modelling

Next we highlight main constituents of a theory of conceptual modelling. It is surprising that literature mainly covers only the first one.

Theory of Modelling Concepts. Modelling concepts are elements of a certain language 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. We therefore may represent the result of modelling by a tuple (r, t, a, m, c, g, w). The group may use its reference models or modes. The theory of conceptual modelling constructs for object-relational database applications can be based entirely on the extended ER model [10].

Theory of Modelling 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 [10] 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.

Theory of Properties of Modelling Activities. It is often neglected that models have their properties. We therefore need a reasoning, evaluation, control and management facility for providing an insight into the model quality itself.

Model Management. The development of a holistic model that covers all but all aspects of an application overburdens modelling languages and overloads cognitive skills of modelers. Therefore, we may separate different aspects and concerns and model those reduced tasks to separate models. These models must however be harmonised and integrated. Therefore, modelling must allow one to reason on model ensembles and their coherence.

Goals and Portfolio of 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.

Results of Modelling. The conceptual schemata are typically considered to be *the* result of conceptual modelling. We may however have different equivalent representations of the same schema, a documentation of the entire development process and the reasons for development decisions, on the scope and restrictions, and last but not least on the bindings among the schemata. The last result offers an opportunity for evolution.

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

Aggregation/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. *Generalizeation/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.

Sample Application. The local-as-view model in Figure 2 is based on a *two-layer architecture* that uses a global schema and local view schemata. We explicitly specify properties and binding among the global and local schemata, e.g. master-slave binding. State changes and the pedestrian calls are not recorded after they have been issued. The scheduler is based on this schema and 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 rules 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.

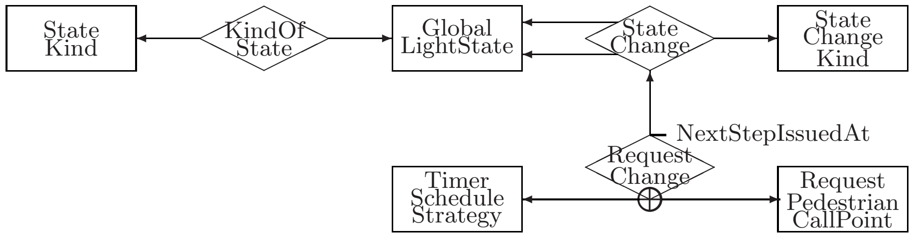

Fig. 2. The traffic light support database schema

4 Application of the Framework to Conceptual Modelling

Methods of Conceptual Modelling. The theory of conceptual modelling is based on a small number of methods. The main methods are *abstraction, modularisation 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.

Properties of Conceptual Modelling. We are interested in a general guidance for the entire modelling process and in a management of all models that supports coherence among the models. The theory of conceptual modelling discussed above should support a selection of a *general modelling strategy*. Layered conceptual modelling is one of the well-known strategies. It is based on modularisation, abstraction and refinement. We decompose a system into components and support a consideration of models in various abstractions and details.

The theory of conceptual modelling may also be used for a selection and development of an assembly of modelling styles. Typical well-known styles [10] are inside-out refinement, top-down refinement, bottom-up refinement, modular refinement, and mixed skeleton-driven refinement. These different kinds of refinement styles allow one to derive *plans* for refinement and and *primitives* for refinement.

The introduction of strategies and styles allow to provide a general support for maintenance of results and qualities achieved so far and for restricting the scope of a change in modelling. We therefore may develop a strategy that has a number of *properties* such as monotonicity of accepted results, incrementality of changes depending only on the most recent solution, finiteness of checks for any criterion requested for the modelling process, application domain consistency without additional reference to later steps, and conservativeness that restricts revisions only to those that are governed by errors or caused by changes in the application.

5 Conclusion

The Theory Framework. The aim of the paper 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.

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.

References

1. Bjørner, D.: Domain engineering. COE Research Monographs, vol. 4. Japan Advanced Institute of Science and Technology Press, Ishikawa (2009)
2. Börger, E., Thalheim, B.: A method for verifiable and validatable business process modeling. In: Börger, E., Cisternino, A. (eds.) Advances in Software Engineering. LNCS, vol. 5316, pp. 59–115. Springer, Heidelberg (2008)
3. Chen, P.P., Akoka, J., Kangassalo, H., Thalheim, B. (eds.): Conceptual modeling: current issues and future directions. LNCS, vol. 1565. Springer, Heidelberg (1999)
4. Kangassalo, H.: Approaches to the active conceptual modelling of learning. In: Chen, P.P., Wong, L.Y. (eds.) ACM-L 2006. LNCS, vol. 4512, pp. 168–193. Springer, Heidelberg (2007)
5. Kaschek, R.: Konzeptionelle Modellierung. PhD thesis, University Klagenfurt, Habilitationsschrift (2003)
6. Olivé, A.: Conceptual modeling of information systems. Springer, Berlin (2007)
7. Schewe, K.-D., Thalheim, B.: Semantics in data and knowledge bases. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2008. LNCS, vol. 4925, pp. 1–25. Springer, Heidelberg (2008)
8. Simsion, G.: Data modeling - Theory and practice. Technics Publications, LLC, New Jersey (2007)
9. Stachowiak, H.: Modell. In: Seiffert, H., Radnitzky, G. (eds.) Handlexikon Zur Wissenschaftstheorie, pp. 219–222. Deutscher Taschenbuch Verlag GmbH & Co. KG, München (1992)
10. Thalheim, B.: Entity-relationship modeling – Foundations of database technology. Springer, Berlin (2000)
11. Thalheim, B.: The conceptual framework to multi-layered database modelling. In: Prc. EJC, Maribor, Slovenia, pp. 118–138 (2009)