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

Bernhard Thalheim

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

17.1 Towards a Theory of Conceptual Models and Conceptual Modelling

Models are different for different purposes. We may develop a model for analysis of an application domain, for construction of a system, for communicating about an application, for assessment, and for governance. These different purposes result in different goals and task portfolios.

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

Models are an essential part of computer science. While preparing a survey on models, we realised that computer science uses more than 50 different models. In analysing these different models, we discover four commonalities:

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

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

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

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

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

Perception support for understanding the application domain.
Explanation and demonstration for understanding an origin.
Preparation to management and handling of the origin.
Optimisation of the origin.
Hypothesis verification through the model.
Construction of an artifact or of a program.
Control of parts of the application.
Simulation of behaviour in certain situations.
Substitution for a part of the application.

Depending on the purpose we shall use different models.

Models are author-driven and addressee-oriented. Figure 17.1 illustrates the association between an origin and the model.

A *model* is typically a schematic description of a system, theory, or phenomenon of an origin that accounts for known or inferred properties of the origin and may be used for further study of characteristics of the origin. *Conceptual modelling* aims to create an abstract representation of the situation under investigation, or more precisely, the way users think about it. Conceptual models enhance models with

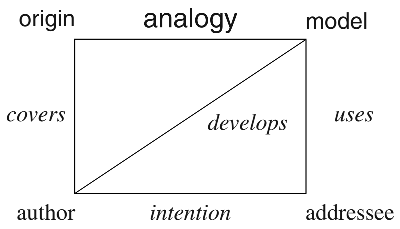


Fig. 17.1 The origin-model-author-addressee relationship for models

concepts that are commonly shared within a community or at least between the stakeholders involved in the modelling process.

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

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

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

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

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

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

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

17.1.1 Artifacts, Concepts and Intentions

17.1.1.1 The Conceptual Model Space

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

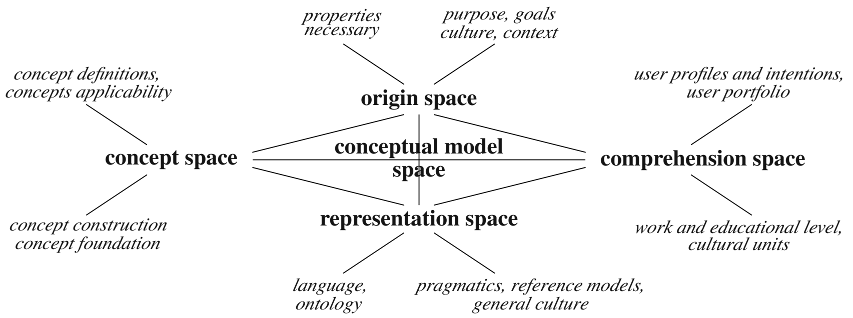


Fig. 17.2 The four aspects of the model space: origin aspect through properties, foundation aspect through concepts, representation aspect through language, user aspect through user comprehension

17.1.1.2 Intentions Driving Modelling

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

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

The second goal of this chapter is the development of a framework for modelling. Modelling is based on an explicit choice of languages, on application of restrictions, on negotiation and on methodologies.

Restrictions depend on logics (deontic, epistemic, modal, belief, preferences) and use shortcuts, ambiguities, and ellipses.

Negotiation supports management or resolution of conflicts and the development of strategies to overcome these strategic, psychological, legal, and structural barriers.

Development methodologies are based on pragmatism and on paradigms. Since modelling is an activity that involves a number of actors, the choice of languages becomes essential.

Modelling is a process and is 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 such acts as construct and evaluate as primary modelling acts.

The third goal of this chapter is to draw attention to explicit consideration of modelling properties both for the models themselves and for the modelling acts. This side of conceptual modelling is often only considered in an implicit form.

The modelling process is governed by goals and purposes. Therefore, we must use different models such as a construction model, a communication model or a discussion model. Modelling is restricted by the application context, the actor context, the system context and the theory and experience context. These contexts restrict the model and the modelling process.

17.1.2 Dimensions of Models and Modelling

17.1.2.1 Main Dimensions of Modelling

Building upon the commonalities observed above for computer science, we introduce four *main dimensions* of models and modelling:

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

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

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

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

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

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

Scope insight, for conscious handling of restriction, capabilities, opportunities.

Guiding rules, for convenience, for completion, refinement, and extension.

Development plans, for partial delivery of models, partial usage and deployment.

Theories supporting development of models.

Quality characteristics for model completion, model evolution, model engineering.

Mapping styles for mapping models among abstraction layers.

17.1.2.2 The Purposes Dimension

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

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

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

I. Analysis: Says what is.
The model does not extend beyond analysis and description. No causal relationships among phenomena are specified and no predictions are made. It thus provides a description of the phenomena of interest, analysis of relationships among those constructs, the degree of generalisability in constructs and relationships and the boundaries within which relationships, and observations hold.
II. Explanation: Says what is, how, why, when, and where.
The model provides explanations but does not aim to predict with any precision. There are no testable propositions. The model provides an explanation of how, why, and when things happened, relying on varying views of causality and methods for argumentation. This explanation will usually be intended to promote greater understanding or insights by others into the phenomena of interest.
III. Prediction: Says what is and what will be.
The model provides predictions and has testable propositions but does not have well-developed justificatory causal explanations. It states what will happen in the future if certain preconditions hold. The degree of certainty in the prediction is expected to be only approximate or probabilistic in IS.
IV. Explanation and prediction: Says what is, how, why, when, where, and what will be.
The model provides predictions and has both testable propositions and causal explanations. A special case of prediction exists where the model provides a description of the method or structure or both for the construction of an artifact (akin to a recipe). The provision of the recipe implies that the recipe, if acted upon, will cause an artifact of a certain type to come into being.

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

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

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

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

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

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

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

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

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

17.1.2.3 The Artifact Dimension

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

The *abstraction layer*, e.g., requirements, specification, realisation or implementation layer.

Chosen *granularity and precision* of the work product itself.

Resources used for development of a model such as the language.

Level of separation of concern such as static/dynamic properties, local/global scope, facets.

Quality properties of the input, e.g., requirements, completeness, conciseness, coherence, understandability.

Decomposition of the work products into ensembles of sub-products.

In addition, modelling of information satisfies quality characteristics such as quality in use, internal quality, and external quality.

17.1.2.4 The User Dimension

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

Author ("by whom"). Results in reflections of the educational level, application of templates, pattern or reference models.
Addressee ("to whom"). Restricts the utilisation of the model or supports the extended application beyond the purpose originally intended.
The broad public ("whichever"). Develops a common understanding of the model depending on the group or the culture of the public.

Users are different and thus modelling has different results because of

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

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

Roles during activities such as stakeholder, developer, consultant, supplier, contractor, documentation developers, or business user.
The practised collaboration partnership based on communication acts, cooperation business processes, and coordination agreements.
Teamwork during all activities with separation of different tasks.
Historical people such as teachers, legacy (heritage) developers, or coders.
Builders of earlier models.

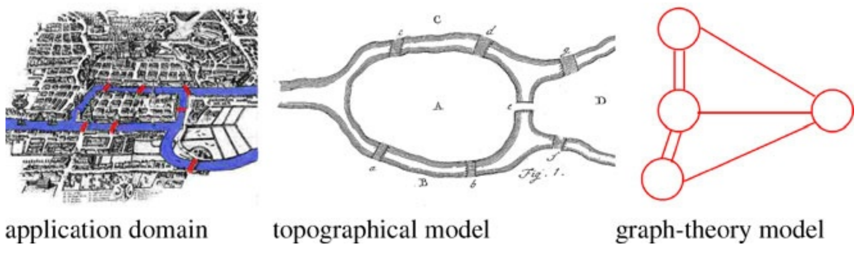
Finally, the user dimension imposes an important restriction to the development, the application, the understanding of models: Models tend to be too large for a single person.

17.1.2.5 The Domain Dimension

The domain dimension clarifies:

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

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



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

17.1.2.6 The Context Dimension

The context dimension is typically used for restricting a model to a specific scope, and thus limits the general utilisation of models. It additionally requires an explicit consideration of these restrictions if the model is used outside its main application area. Context abstraction is a useful vehicle for restricting attention. Typical specific context restrictions to models include:

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

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

Envisioned *evolution paths ("whither")* for the adaptation of the model to future requirements.

17.1.3 Postulates of Modelling

17.1.3.1 General Properties of Models

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

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

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

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

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

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

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

The first three properties are based on Stachowiak's theory of models [12, 16]. The fourth property has been formulated in [17]. The fifth property has been discussed in [9]. The sixth property has been developed within natural sciences (chemistry).

17.1.3.2 Prescription by Models and Description for Models

Figure 17.3 depicts the association between origin and artifacts.

Similarly, we may describe the application of models for construction of other artifacts such as software and hardware. Figure 17.4 gives the reflection of the postulate. We observe that the completion activities change compared with Fig. 17.3.

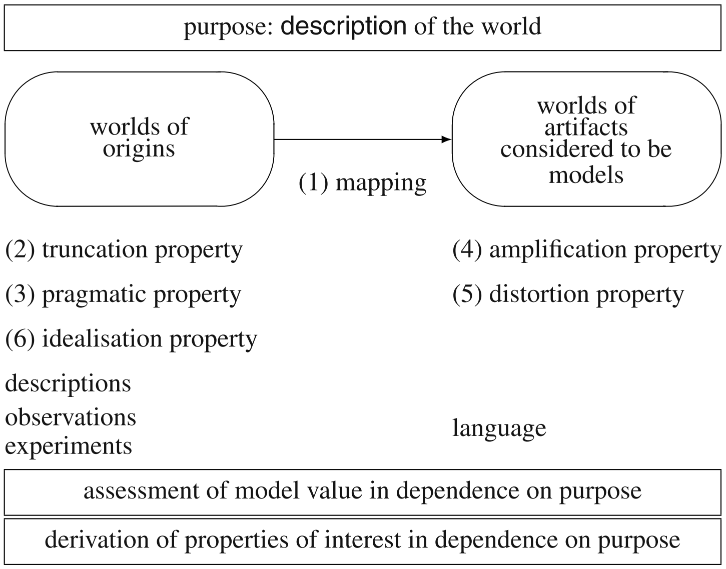

purpose: description of the world
worlds of origins
(1) mapping
worlds of artifacts considered to be models
(2) truncation property
(3) pragmatic property
(6) idealisation property
descriptions
observations
experiments
(4) amplification property
(5) distortion property
language
assessment of model value in dependence on purpose
derivation of properties of interest in dependence on purpose

Fig. 17.3 The association of worlds and resulting postulates of models. The bottom rectangles describe the completion activities that should be completed according to the purpose

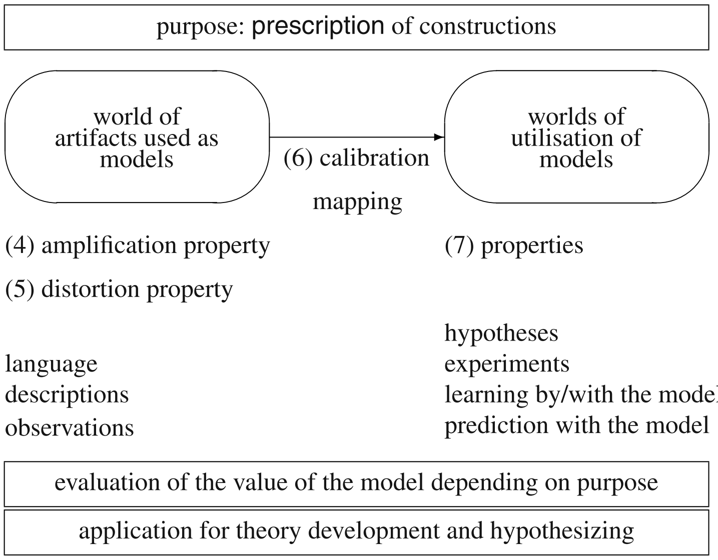

purpose: prescription of constructions
world of artifacts used as models
(6) calibration mapping
worlds of utilisation of models
(4) amplification property
(5) distortion property
language
descriptions
observations
(7) properties
hypotheses
experiments
learning by/with the model
prediction with the model
evaluation of the value of the model depending on purpose
application for theory development and hypothesizing

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

17.1.3.3 The Model Capacity

The model

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

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

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

17.1.3.4 Resulting Restrictions To Be Accepted by Stakeholders

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

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

17.1.4 Artifacts and Models

The four aspects of the conceptual model space in Fig. 17.2 are interwoven. Models use artifacts. Models have their specific representation. Models are supported by conceptualisations.

The interrelationship between models, representations and concepts should be very flexible. We can assume that models may use different representations or artifacts. Artifacts may contain sub-artifacts. Conceptual models are based on concepts. Concepts may be typical for a model within a certain degree of typicality. Concepts may consist of sub-concepts. Therefore, we may associate a model with a concept that has its own sub-concepts.

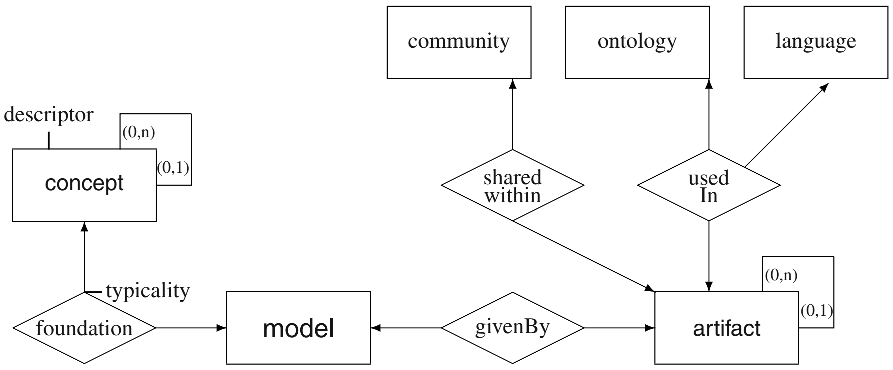


Fig. 17.5 The association between artifacts, representations, and concepts

We assume that concepts are independent of representations. Additionally, we may assume that representations are dependent on the language and some ontology to be used. They are typically commonly accepted or shared within a community or culture. This understanding leads to the structure displayed in Fig. 17.5.

17.2 The Theory of Conceptual Models

17.2.1 Conceptual Models and Languages

17.2.1.1 The Language Dimension

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

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

The model-artifact association is agreed upon within a community. This community is based on a web of knowledge of their members (see Chap. 15).

17.2.1.2 Languages Used for Representation of Models

Languages are the carrier for models. We may accept a logics approach to semiotics and define the language similar to Chap. 12. Each constructive language is based on

$$\begin{tikzcd}
 & \mathscr{L}_{\tilde{G}} \text{ language} \arrow[dl, dash, dashed] \arrow[d, dash, dashed] \arrow[dr, dash, dashed, "\supseteq"] & \\
\Psi(G) \text{ postulates} \arrow[r, "\text{corresponds}"] & G \text{ artifact} \arrow[r, "\text{scope}"] & \Phi(G) \text{ statements}
\end{tikzcd}$$

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

a signature, on a set of base items, and a set of constructors. The language consists of words that are allowed due to well-formedness constraints. Languages $\mathcal{L}$ are used for a number of reasons, e.g. reasoning, representation, illustration, etc. These reasons are driven by the model purposes in our case.

We may now use a subset of words and accept those as *postulates* for a model. To be considered faithful or useful, a model must satisfy these postulates.

We may develop different artifacts as a potential models. We are, however, interested in some properties that these models must satisfy. We therefore develop a general understanding of artifacts that are used for models within a language.

Postulates must be explicitly given. They might be changed whenever the purpose of modelling is changing. They do not restrict models to one artifact. Instead, we might also use a number of artifacts in parallel. Figure 17.6 displays the relationship between an artifact and its postulates and properties.

Constructive languages thus provide support for:

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

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

$\mathcal{L}_{\tilde{G}}$	$\Psi(G)$	corresponds	G	scope	$\Phi(G)$
Logics	Axioms	Satisfy	Structure	Satisfy	Essential properties
$\mathbb{N}$	Peano axioms	Satisfy	Standard model	Derivable	Peano arithmetics
Empirism	Postulates	Accepted	Artifact	Supports	Observation
Technics	Construction requirements	Enforce	Product	Has	Properties

This approach also carries classical approaches used in mathematical logic:

$\mathcal{L}_{\tilde{G}}$	$\Psi(G)$	corresponds	G	scope	$\Phi(G)$
Logics	Axioms	Satisfy	Structure	Consider	Essential properties of G
Logics	Axioms	Satisfy	Structure	Satisfy	Relevant theorems of $(\mathcal{L}_{\tilde{G}}, \Psi(G))$

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

The same approach can also be used for conceptual modelling:

$\mathcal{L}_{\tilde{G}}$	$\Psi(G)$	corresponds	G	scope	$\Phi(G)$
Database	Requirements	Realise	DB schema	Satisfy	Integrity constraints
Workflow	Requirements	Realise	WF schema	Satisfy	Integrity constraints

Thus, ingredients used for modelling of databases, information systems and workflow systems are languages, restrictions, negotiations for the property to be a model, and methodologies for artifact development. Languages are given with syntactics, semantics, and pragmatics. We typically use inductive expression formation based on alphabets. Inductive construction also supports the description of behaviour defined on expressions.

Restrictions depend on the logics to be used, e.g., first-order hierarchical predicate logics [18], deontic logics, epistemic logics, modal logics, logics for belief reasoning of for preference derivation. Negotiations provide a means to identify, define analyse barriers and manage or resolve conflicts. Methodologies of development are based on engineering approaches and are guided by certain pragmatism and a number of paradigms.

17.2.1.3 Principles of Language Use

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

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

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

17.2.1.4 The Matter of Language Choice

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

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

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

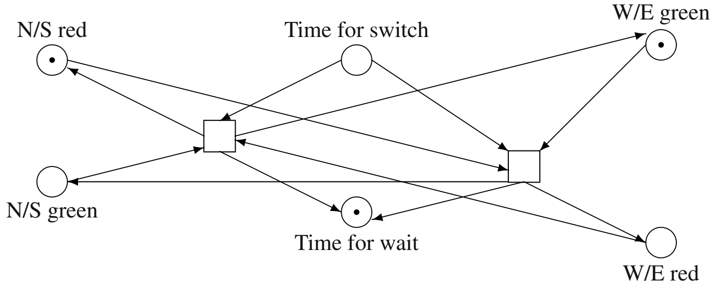


Fig. 17.7 Traffic control based on Petri nets

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

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

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

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

We might prefer the local-as-view approach. States reflect the entire state of the crossroad, i.e. *NSredEWgreen, NSredEWred, NSgreenEWred*. The last state reflects that the north-south direction is open and the east-west 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.

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

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

The local-as-view model is based on a *two-layer architecture* that uses a global schema and local view schemata. The extended ER model [25] provides a number of opportunities for the representation of hierarchies. A typical hierarchy in our traffic light application is the specialisation hierarchy for states. Since states can be multiply classified depending on the time of day and the day of the week, we might choose the bulk representation for the classification of types through a *StateKind* instead of

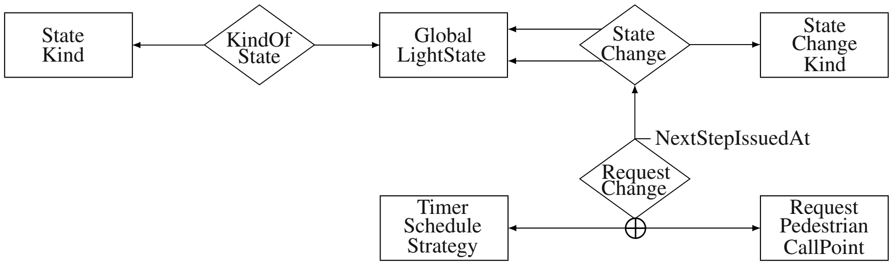


Fig. 17.8 The traffic light support database schema

explicit specialisation types. State changes may also be classified in a similar way. We might, however, prefer to separate calls for state change made by pedestrians from those state changes that are triggered by a clock.

Based on these choices, we derive modelling activities for the database schemata and workflow rules. We explicitly specify properties and binding among the global and local schemata, e.g. master-slave binding.

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

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

```
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).
```

Similarly, we can specify views for local display.

17.2.1.5 Pragmatics that Cannot Be Neglected

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

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

Semantics of generative languages can be either defined by meta-linguistic semantics, e.g. used for defining the semantics of predicate logics, by procedural or referential semantics, e.g. operational semantics used for defining the semantics of programming languages, or by convention-based semantics used in linguistics. Semantics is often defined on the basis of a set of relational structures that correspond to the signature of the language.

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

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

Given a type R and substructures X, Y of R.
The functional dependency $R : X \longrightarrow Y$ is valid in R^{C} if $o|_Y = o'|_Y$ whenever $o|_X = o'|_X$ for any two objects o, o' from R^C.

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

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

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

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

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

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

17.2.2 Concepts and Models

Concepts are the basis for conceptual models. They specify our knowledge what things are there and what properties things have. Concepts are used in everyday life as a communication vehicle and as a reasoning chunk. Concepts can be based on definitions of different kinds.

Thus, our goal for the development of a theory of conceptual modelling and of conceptual models can only be achieved if the conceptual model definition covers any kind of conceptual model description and goes beyond the simple textual or narrative form.

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

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

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

- Concepts typically also depend on the application context, i.e. the application area and the application schema. The association itself must be characterised by the kind of association.

Concepts are typically ordered hierarchically and can thus be layered. We assume that this ordering is strictly hierarchical and the concept space can be depicted by

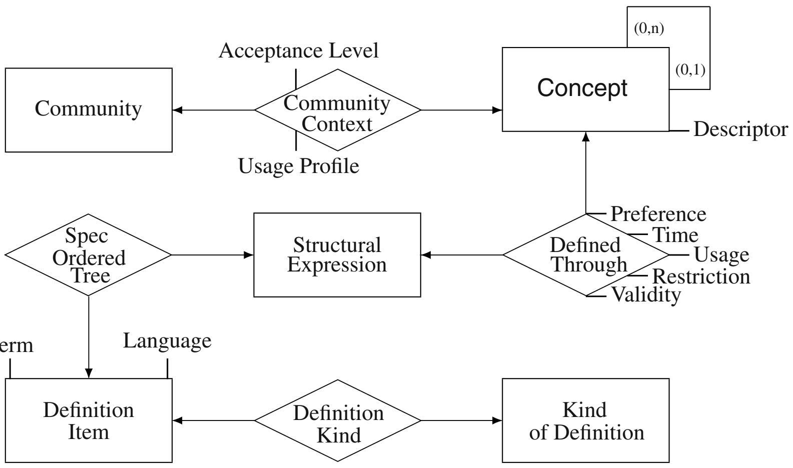


Fig. 17.9 The main schema for concept definition and formation

a set of concept trees. A concept is also dependent on the community that prefers this concept. A concept is typically given through an embedding into the knowledge space of users involved.

The schema in Fig. 17.9 displays the general structure for content definition. This schema also covers all aspects discussed in [13]. This schema extends the relationship between artifacts, representations and concepts introduced in Fig. 17.5.

Concept gathering can be understood as a technique that combines concept representation [5, 13, 19] and (algorithmic) learning approaches.
A concept gathering system is based on:

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

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

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

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

A run of the concept gathering system *results in*

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

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

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

17.2.3 Information Exchange of Stakeholders Based on Models

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

- The first category of definitions is based on the mathematical notion of entropy. This notion is independent of the user and thus inappropriate in our project context.
- The second category of definitions bases information on the data a user has currently in his data space and on the computational and reasoning abilities of the user. Information is any data that cannot be derived by the user. This definition is handy but has a serious drawback. Reasoning and computation cannot be properly characterised. Therefore, the definition becomes fuzzy.
- The third category is based on the general language understanding of information: the communication or reception of knowledge or intelligence. Information can also defined as

  – knowledge obtained from investigation, study, or instruction;
  – intelligence or news;
  – facts and data.

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

All these definitions are too broad. We are instead interested in a definition that is more appropriate for the internet age:
Information as processed by *humans*,

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

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

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

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

$$I_{\mathrm{B}} = i_{\mathrm{B}}(o_{\mathrm{B}}(r_{\mathrm{B}}(s_A(D_A))), K_{\mathrm{B}}, t_{i_X}) .$$

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

$$I_{\mathrm{B}} = i_{\mathrm{B}}(o_{\mathrm{B}}(r_{\mathrm{B}}(s_A(D_A, t_{s_A}), t_{r_{\mathrm{B}}}), t_{o_{\mathrm{B}}}), K_{\mathrm{B}}, t_{i_{\mathrm{B}}}) .$$

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

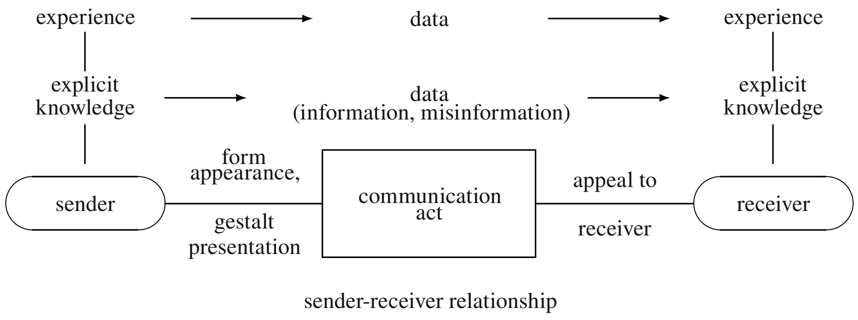


Fig. 17.10 Dimensions of the communication act

The act of communication is specified by

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

17.2.4 Mappings Among Models and Originals

17.2.4.1 Modelling Supported by Mapping

Thus far, two of the four main dimensions have been established. Let us now consider the mapping between two worlds: source world and target world. Examples of source-target pairs include:

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

The first mapping (e.g., in Fig. 17.3) is typically based on a *description* of the origins that is represented by a model about these origins. The second mapping (e.g., in Fig. 17.4) is typically based on a *prescription* made by the model for a realisation of the model by a technical artifact. The third mapping has been used above for the association between the topographical model and the graph model for the Königsberg bridge problem on page 551.

We can observe other pairs of such mappings, depending on the purpose. For instance, documentation uses an artifact to be documented and another artifact that documents essential elements of the first artifact. It typically extends the first artifact for pragmatic rules for exploitation of the first artifact and by behavioural scenario as examples of deployment. It bases the documentation also on an idealisation of the first artifact.

A similar association may be developed for the other purposes:

- perception support for understanding the application domain;
- explanation and demonstration for understanding;
- preparation to management and handling of the original;
- optimisation of the application domain operating;
- hypothesis verification through the model;
- control of parts of the application;
- simulation of behaviour in certain situations; and
- substitution for a part of the application.

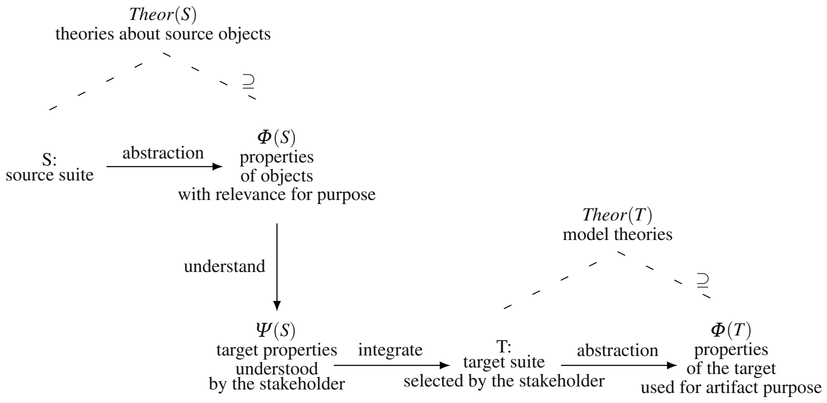


Fig. 17.11 Mappings between source artifacts and target artifacts

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

Furthermore, we observe another important property of the mapping:

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

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

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

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

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

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

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

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

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

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

The main postulate for the mapping is however the

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

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

- The mapping is a realisation of an analogy property.
- It is possible to re-map properties observed for the target artifact to the source artifact if those are not caused by idealisation, distortion or amplification.
- The target artifact can also be used for other mapping with different intentions and goals.

We shall discuss specific forms of analogies below.

As an example, we may refine Fig. 17.11 to classical ER modelling, as displayed in Fig. 17.12. This figure allows also to reason on the advantages and on the disadvantages of the ER modelling approach.

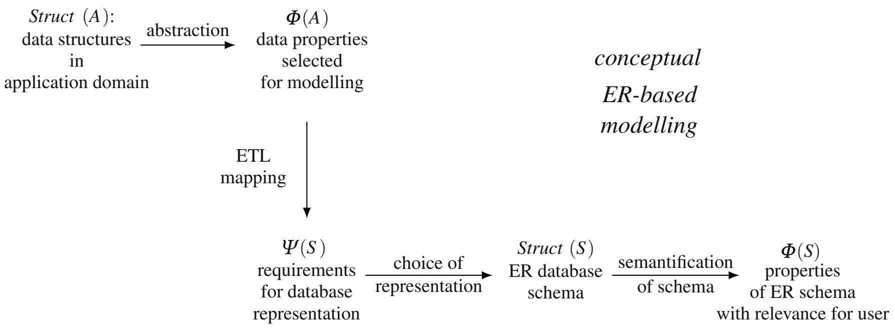

Struct (*A*): data structures in application domain
abstraction
Φ(*A*) data properties selected for modelling
conceptual ER-based modelling
ETL mapping
Ψ(*S*) requirements for database representation
choice of representation
Struct (*S*) ER database schema
semantification of schema
Φ(*S*) properties of ER schema with relevance for user

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

17.2.4.2 Modelling with a Manifold of Models

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

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

Figure 17.13 displays one pitfall of multi-language modelling. The models may consider different aspects of the origin, they may contradict and they may not be integratable. For instance, if we use class diagrams, statecharts, activity diagrams, time diagrams, component diagrams, interaction diagrams and others within a software development team, then integration of different aspects might become infeasible. Thus, we may apply two rather rigid modelling restrictions, which serve as the main principles for multi-language modelling:

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

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

Chapter 12 considered co-evolution of models and introduced a formalism to handle coherence. *Coherence* describes a fixed relationship between the models in a model suite. Two models are coherent when each change in one of the models is propagated to the other model. This change transfer implicitly assumes that the integrity constraints of the corresponding model types remain to be valid. Models are non-coherent if there is a random or changing relationship. We aim for an explicit specification of the association schema and use an explicit specification of the *collaboration* among models. For instance, the master-slave association or collaboration

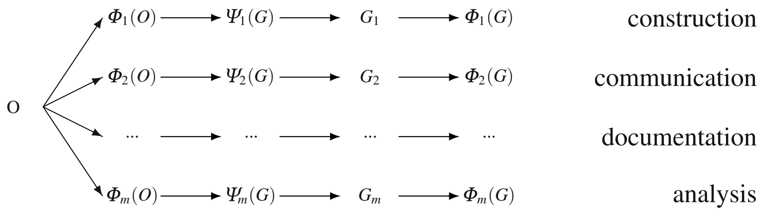


Fig. 17.13 Models reflecting different purposes

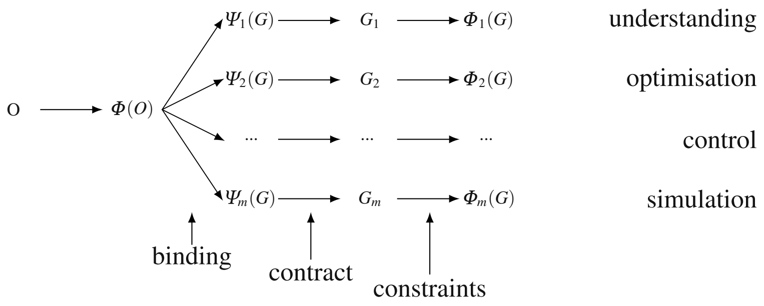

$O \to \Phi(O)$
$\Psi_1(G) \to G_1 \to \Phi_1(G)$
$\Psi_2(G) \to G_2 \to \Phi_2(G)$
$\dots \to \dots \to \dots$
$\Psi_m(G) \to G_m \to \Phi_m(G)$
understanding
optimisation
control
simulation
binding
contract
constraints

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

propagates any change of the master to its slaves. Slaves do not have any right the change the master without consensus with the master.

If we enforce these principles, then the model variety can be handled in a simpler and feasible way. Figure 17.14 displays the advantages of a coherent set of models, based on a complete set of properties of the origin. It allows us to introduce a binding between these models. This binding can be mapped to a contract in the sense of Chap. 12. The contract may be used for the derivation of constraints the different models must obey in order to be coherent.

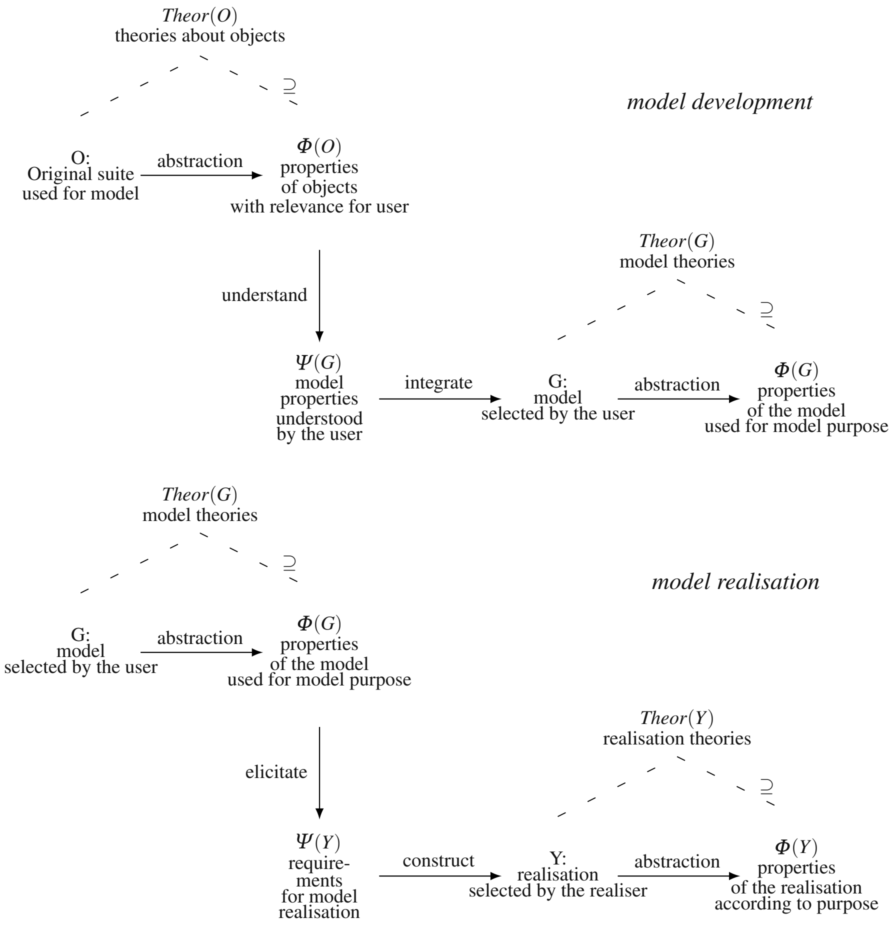
This approach directly results in a *coordination of models* on the basis of *separation of aspects*.

17.2.5 Development Phases That Use Models

17.2.5.1 Description Through Models and Prescription by Models

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

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



Theor(O)
theories about objects
⊇
model development
O: Original suite used for model
abstraction
Φ(O) properties of objects with relevance for user
understand
Theor(G)
model theories
⊇
Ψ(G) model properties understood by the user
integrate
G: model selected by the user
abstraction
Φ(G) properties of the model used for model purpose
Theor(G)
model theories
⊇
model realisation
G: model selected by the user
abstraction
Φ(G) properties of the model used for model purpose
elicitate
Theor(Y)
realisation theories
⊇
Ψ(Y) require-ments for model realisation
construct
Y: realisation selected by the realiser
abstraction
Φ(Y) properties of the realisation according to purpose

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

17.2.5.2 Reasoning Support for Modelling

Design science [7] has been aiming at an explicit support for the modelling process. This support includes an explicit consideration of the quality of the model, the modelling process, and supporting theories. We may combine the informal discussions with our approach and separate the modelling acts by the things that are under consideration.

Figure 17.16 displays the different ways of working during a database systems development. We use here the two-phase model: description followed by prescription.

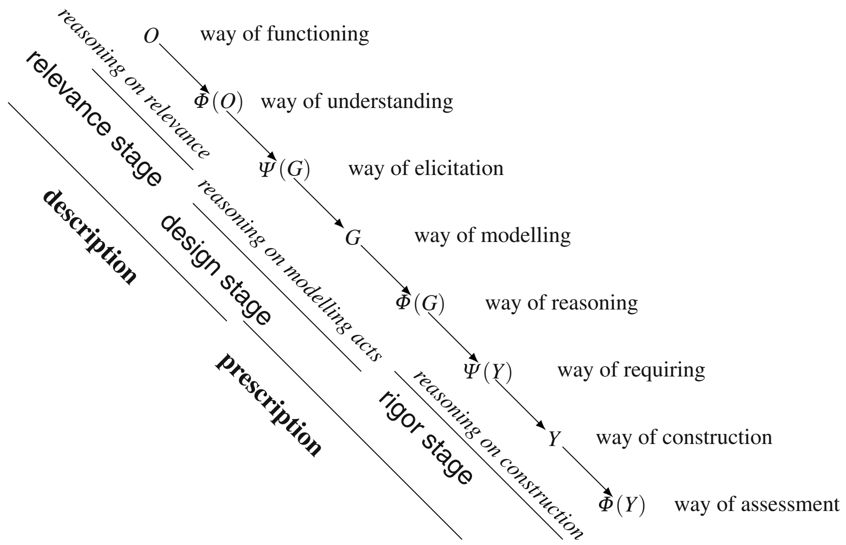

O way of functioning
$\Phi(O)$ way of understanding
$\Psi(G)$ way of elicitation
G way of modelling
$\Phi(G)$ way of reasoning
$\Psi(Y)$ way of requiring
Y way of construction
$\Phi(Y)$ way of assessment
reasoning on relevance
relevance stage
reasoning on modelling acts
design stage
reasoning on construction
rigor stage
description
prescription

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

These different "ways of working" characterise

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

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

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

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

The rigor stage may be replaced by other stages that support different purposes.

We have concentrated on prescription and construction of new systems. Another application is *model refinement* similar to two-model representation of the Königsberg bridge problem on page 551.

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

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

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

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

Degree of *structural analogy.* The degree of similarity of either the original with the model or of the model with its reflection.

Degree of *qualitative analogy.* The degree to which the character and constitution is reflected.

Degree of *structural adjustment.* The extent to which the structure is considered independent on the later use.

Degree of *qualitative adjustment.* Characterizes what is going to be used for the later exploitation and what part is not going to be used.

Degree of *functional adjustment.* Characterises the functions that are considered and the functions that are not considered.

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

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

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

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

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

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

Another such rule is:

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

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

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

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

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

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

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

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

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

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

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

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

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

17.3 Conclusion

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

The programme requires far more work. The theory needs a variable taxonomy that allows a specialisation to languages chosen for a given application domain, must be based on a mathematical framework that allows one 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 following are the findings of this chapter:

- A model is a representation of something for someone's purpose somebody and developed by someone else.
- Each model is author-driven and addressee-oriented, is aspect-related, is purpose-specific, is limited in space, context and time, and is perspective.
- The model quality is also given by those elements that are not observed for the origin or not realised in the reflection or realisation.
- Due to amplification, distortion and idealisation, models cannot be used outside their purpose. If the purpose changes then the model should change as well.

- Models are similar to concepts; they are abstract and concrete; they associate worlds, e.g., the world of origins and models.
- Conceptual models are similar to other systems that are context- and utilisation-dependent. They have their value within the purpose range.

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

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

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

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

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

References

1. Beeri C and Thalheim B (1999) Identification as a primitive of database models. In: Proceedings FoMLaDO'98. Kluwer, London, pp 19–36
2. Börger E, Thalheim B (2008) A method for verifiable and validatable business process modeling. In: Software Engineering, Lecture notes in computer science, vol 5316. Springer, Heidelberg, pp 59–115
3. Chen PP, Akoka J, Kangassalo H, Thalheim B (eds) (1998) Conceptual modeling: current issues and future directions. Lecture notes in computer science, vol 1565. Springer, Heidelberg
4. Deppert W (2009) Theorie der Wissenschaft. Lecture notes for academic year 2008/09, Christian-Albrechts-University at Kiel, http://wolfgang.deppert.de
5. Fiedler G, Thalheim B (2009) Towards semantic wikis: modelling intensions, topics, and origin in content management systems. Inform Model Knowl Bases XX:1–21
6. Gregor S (2009) Building theory in the sciences of the artificial. In: DESRIST. ACM, New York
7. Hevner A, March S, Park J, Ram S (2004) Design science in information systems research. MIS Quaterly 28(1):75–105
8. Jackson M (2006) Problem frames. Pearson, Harlow, Essex
9. Kaschek R (2003) Konzeptionelle Modellierung. PhD thesis, University Klagenfurt
10. Mahr B (2009) Information science and the logic of models. Softw Syst Model 8:365–383
11. Mäkinen T (2010) Towards assessment driven software process mpdeling. PhD thesis, TUT Pori
12. Mittelstraß J (ed) (2004) Enzyklopädie Philosophie und Wissenschaftstheorie. J.B. Metzler, Stuttgart
13. Murphy GL (2001) The big book of concepts. MIT Press, Cambridge
14. Olivé A (2007) Conceptual modeling of information systems. Springer, Berlin
15. Simsion G (2007) Data modeling – Theory and practice. Technics Publications, Denville

16. Stachowiak H (1992) Modell. In: Seiffert H and Radnitzky G (eds) Handlexikon zur Wissenschaftstheorie. Deutscher Taschenbuch Verlag, Munich, pp 219–222
17. Steinmüller W (1993) Informationstechnologie und Gesellschaft: Einführung in die Angewandte Informatik. Wissenschaftliche Buchgesellschaft, Darmstadt
18. Thalheim B (2000) Entity-relationship modeling – foundations of database technology. Springer, Berlin
19. Thalheim B (2007) The conceptual framework to user-oriented content management. Information Modelling and Knowledge Bases, XVII:30–49
20. Thalheim B (2009) Towards a theory of conceptual modelling. In: ER Workshops. Lecture Notes in Computer Science, vol 5833. Springer, Heidelberg, pp 45–54
21. Thalheim B (2009) The conceptual framework to multi-layered database modelling. In: Proceedings EJC, Maribor, pp 118–138
22. Whorf BL (1980) Lost generation theories of mind, language, and religion. Popular Culture Association, University Microfilms International, Ann Arbor