Concerning issues of theories of conceptual modeling

Roland Kaschek
The University of The Faroe Islands
rolandkaschek@gmail.com

Abstract. In this paper I discuss issues concerning theories of conceptual modeling. First I introduce the required basics and discuss then briefly the acquisition of empirical knowledge by means of conceptual modeling. Following this I argue that, contrary to popular belief, representation is key to modeling rather than abstraction. I furthermore show that the common view of models as being per-se descriptive or prescriptive cannot be sustained. In fact a variety of references can be made from any model to many different things. I argue that the peculiarities of such reference are responsible for resulting normative effects rather than the ontological status of the model.

1 Introduction

As far as I know the theory of conceptual modeling is still in its infancy. This is not only due to the young age of informatics which is a main driver of conceptual modeling. Rather, conceptual modeling exists in a sphere in which conflicting interests overlap. This results in changing priorities and approaches and thus a slow development of the field. Nevertheless theories of conceptual modeling are important for education of modelers and for providing practical guidance in software development projects. Since theories of conceptual modeling to some extent belong to metaphysics they can hardly be proven wrong or right. The individual theories of conceptual modeling, however, may address certain issues in different ways and with different emphasis. They also might be more or less complete, conclusive and coherent. It might be difficult or even impossible to blend parts of different such theories into one coherent theory. It might nevertheless be of some importance for authors on conceptual modeling or practitioners to at least know about parts of such theories. It is the purpose of this paper to bring to the attention of the modeling community solutions concerning a few issues of conceptual modeling that in my view currently widely are not well taken care of.

Conceptual models consist of concepts. With regard to what a concept is I draw from [9]. I focus in this paper on what members of modeling communities do with and by means of models. It is obvious that with a conceptual model one refers to something apart from it. This so-called mapping-property together with the pragmatic and the truncation property of models by Stachowiak [20]
has been considered as key for models. And indeed these characteristics can easily be identified for a model. Consider, for example, a conceptual model for a payroll application. It very likely includes concepts such as “employee”, “bank account”, “work contract” and “employment law”, etc. Some part of the “world” of companies, employees, trade unions, employment laws etc. can be considered as original of that model. The model obviously is short of certain characteristics of that original and so the truncation property holds. The pragmatic property can be observed upon inspection of the model since it was made for a particular purpose and is used by humans for whom the model has a particular meaning.

By means of the concepts in a model one refers to things in the universe of discourse (UoD) the model is about. In such a model one, moreover, is likely to find some sort of association. For example, it is likely that associations exist in the above model between the concepts “employee”, “work contract” and “employment law”. Such an association is an aid in relating to each other instantiations of concepts within the model and thus things within the model’s original. Finally, characterizations will be available of those concepts. For example, it will be part of such model that an instance of “employee” has characteristics such as “name”, “address”, “marital status”, “qualification” and so forth. However, as [17] shows in passing there is currently no agreement by writers on modeling as to what exactly a model is.

Modelers, as can be summarized, using their models do three important things repeatedly: they refer to a number of things in a UoD; they characterize such things and relate them to each other. Of course this is done so models can be created that aid model user to solve their tasks. Obviously, for that to happen the models need to have an appropriate consistency. That consistency can be understood in terms of the logical structure of the model, the way the model is adjusted to the capabilities of its users and the medium within which the model is represented. Whether or not a model has the required consistency hardly can be decided by purely deductive means. Rather the observation will be necessary in general of what model users actually do with the model. It also will be necessary to analyze what they gain by using the model and whether or not that was worth creating and maintaining it.

Modelers use the concepts of modeling languages (such as the Unified Modeling Language (UML), the Entity-Relationship Model (ERM) or similar) to create models. They instantiate these concepts for a given UoD so they can identify, classify and relate to each other things within that UoD. Modeling languages include the concept of “type” to aid modelers in that regard. A simple genealogy of type systems can be outlined at once: Out of an archetype “nucleus” of types, by means of subdivision, two types can be created, namely a “thing” and a “value"-type. This allows thing-type instances to be described in terms of value-type instances. Thus one can attempt to standardize the way things are characterized. The advantage gained by that standardization is the ultimate reason for which this type subdivision is commonly found in modeling languages. In a second step the “thing”-type can be subdivided into “entity”-type and “relationship”-type. This second step already creates the type system
of the ERM. Of course for practical application that type system is too coarse and these types have to be instantiated further. As far as the ERM goes these instantiations aim at focusing on the different utility of the references to be carried out. They thus do not introduce any fundamentally new ways to relate to things within the UoD. With regard to the example above “employee”, “bank account” and “employment law” are likely to be instantiations of the type “entity”. Two instantiations of the type “relationship” that are likely to be found in related UoD’s is “heads department” or “supervises”.

By subdividing further the “entity”-type an “object”-type with or without behavior is created and by doing so a new view on the UoD is facilitated. It is remarkable that a number of major authors on these subjects were not really clear about what an “entity” or “object” actually is. Chen, for example [4, p. 10], wrote: “An entity is a ‘thing’ which can be distinctly identified. A specific person, company, or event is an example of an entity.” This explanation actually does not answer the question what an “entity” is and takes resort to examples. For, the only thing Chen says about entities is that they can be identified. Certainly that is the minimum requirement for a related meaningful discourse. It, nevertheless, does not explicate what a model is. Similarly the amigos [2, ch. 1] wrote: “Simply put, an object is a thing, generally drawn from the vocabulary of the problem space or the solution space . . . Every object has identity (you can name it or otherwise distinguish it from other objects), state (there’s generally some data associated with it), and behavior (you can do things to the object, and it can do things to other objects, as well).” Again it remains unclear what a “thing” really is.

The problem to define terms like “entity”, “object” and comparable ones seems to result from the natural attitude to use an ontological approach. The problem in fact disappears if an activity oriented approach is used. For example, I suggest to say that an “entity” is a successful deictic reference schema (DRS). Google defines deictic by “Of, relating to, or denoting a word or expression whose meaning is dependent on the context in which it is used, e.g., here, you, me, that one there, or next Tuesday”. For this paper that context is the UoD the model is about and that a computerized problem solution is going to be worked out. I call a DRS successful if it is repeatable, reliable and inter-subjective. With these terms I mean that the related DRS can be instantiated at will, the outcome of its instantiations by a single individual do not vary considerably and the individual stakeholders, who have to instantiate it, can do so with acceptable variety in the outcome, respectively. Of course DRS have further characteristics that play a role in conceptual modeling such as their deployment cost and their implied education cost. These characteristics are ignored in the rest of the paper. The advantage of considering concepts as DRS is that ontology related issues need not be discussed, that the minimal requirement is put forward and that characteristics become apparent that not necessarily would be discussed at all in an ontology-driven approach. I adopt the concept DRS from [9].

Carnap [3, p. 11] has emphasized the importance of referencing, characterizing and relating to each other things that are included of an original of a
model. His book “Der logische Aufbau der Welt” includes an implicit precursor of Chen’s ERM. In it Carnap (pp. 17) discusses, as an example, how the European railway system of his time could be modeled as a graph and be analyzed in graph-theoretic terms. While discussing his example Carnap points out that the only resemblance between model and original that he is going to acknowledge rests on his modeling system. According to it: (1) each railway station is represented by exactly one vertex; (2) each vertex in the graph would represent a railway station; (3) each arc between any pairs of vertices represents a direct train connection between the related stations; and (4) each direct railway connection between any two stations is represented by exactly one arc pointing in the required direction. Carnap thus introduces the binary ERM with 1 : 1 relationships only. He used it for modeling the European railway system. The nature of his example did not lead him to consider associations between non-singleton sets of things. Each association he considers is the cohesion between exactly two railway stations. Chen has extended this limited view of association when he defined the ERM. He used higher arity relationship types as well as $m : n$ and $1 : n$ relationship types. Chen’s version of the ERM is still in use in recent textbooks, see for example [18].

2 Models and empirical knowledge

In 1845 Marx wrote [15] in the second of his thesis on Feuerbach: “The question whether objective truth can be attributed to human thinking is not a question of theory but is a practical question. Man must prove the truth, i.e. the reality and power, the this-sidedness of his thinking in practice...” I take this as the identification of two dimensions of human thought. I refer to them by the term constitution and deduction, respectively.

Modeling communities in their practice manage to capture and exploit the required knowledge about our world. That practice includes the activity kinds of “referring”, “relating” and “characterizing”. That practice, moreover, includes to change or abandon models if successful use of them can not be established (anymore). Modeling, in a way, thus is an experimental pursuit. It involves conducting experiments to figure out the ways in which a modeling community can refer, relate and characterize beneficially for a given task. Obviously, to that end, occasionally even the UoD has to be changed. Models in consequence are not objective. They are always tools of a community and successful models therefore are tailored towards that community and depend on it. With regard to that view I draw a lot from [7]. The empirical knowledge captured via a model is the knowledge that proceeding such and such way with such and such conventions

1 To my knowledge this book has been quite influential. Its first English translation according to the Wikipedia (https://en.wikipedia.org/wiki/Rudolf_Carnap) nevertheless has only been published in 1967.

2 In [12, p. 122] I found the interesting note that the first “illustration” of a directed graph allegedly was prepared by Bertrand Russel and dates only back to 1919.

3 I do not know whether or not he did that elsewhere.
under such and such circumstances one can, at acceptable cost and given peculiar circumstances, solve a given task to the required degree and achieve the required solution quality.

The term “entity” can be defined as “a successful DRS that capitalizes on that reference’s utility and characteristics”. Similarly the term “relationship” can be defined as “a successful DRS that capitalizes on that reference’s capability to associate to each other specified entities”. A weak entity, following this line of thinking, is an entity with regard to which a successful reference schema only can be established in terms of its utility plus another entity. The term “process” can be defined as “a successful DRS that capitalizes on changes within the limits of persistence”. Terms like “entity type”, “relationship type”, “process type” and similar ones can be defined as the related plurality concepts whose extension are exactly those entities, relationships or processes that have been obtained as successful DRS.

When modelers employ a modeling language, such as the ERM, to create a model then they work out two things, namely the model’s thesaurus and its graph. The thesaurus (aka data dictionary or data ontology) contains the definitions of the terms used to give meaning to the model. Usually the graph (aka diagram) is labeled by the thesaurus terms since they are not part of the modeling language. It is thus chiefly by means of the thesaurus that modelers and model users achieve to refer to and to characterize UoD-things. The graph mainly helps to represent each concept’s context within the UoD. It seems that the thesaurus’ passive role as information repository in part is reflected in current work such as in [6] or [21]. Others, however, such as [17], ignore it entirely. To assure the success of a modeling project one obviously has to guaranty that the intended references, as defined in the thesaurus, will be carried out successfully in the related modeling community’s practice. Thus the quality of a model’s thesaurus is critical for the successful use of the model. Therefore that quality needs to be managed. Based on examples and counter examples an initial validation of the thesaurus can be attempted. With regard to functional dependencies as used in database design, for example, one could try to use Armstrong relations for that purpose since these relations are known to only satisfy the stated dependencies [1, p. 168].

Additionally to providing definitions, aliases and explanations the thesaurus is very important with regard to referring and characterizing since it can be ordered alphabetically. Therefore any defined terms can be found efficiently in the thesaurus and failing to find a term is a very strong indicator for that term not to be in the thesaurus. The latter for large models is not necessarily true with regard to the model’s graph. The actual reference works such that the thesaurus for each term, by means of its definition and further elaboration, gives a link into the UoD, i.e. provides a DRS. Path expressions may then be used to specify a collection of UoD-things. Each reference then is the intersection of the thing sets specified by initial references and path expressions on top of them.
3 Modeling vs. abstracting

It has been observed that a close relationship exists between abstraction and modeling. The amigos, for example, say [2]: “What, then, is a model? Simply put, A model is a simplification of reality.” Similar utterances are quoted in [17]. I am going to argue that the point has been made wrongly by these authors. I am going to explain what I mean here based on an example that I find typical for model and modeling. I reuse for that an example from the literature, [11]. Consider a conventional city map of Torshavn. Then most likely no one objects the possibility to consider that map as a model of Torshavn. And of course it also represents an abstraction of Torshavn. This is a consequence of ignoring most and only retaining a few characteristics of that city. Those latter characteristics are then apparent upon inspection of that map. The map, however, does not tell how the wind feels when you walk the harbor in winter. It doesn’t tell what folks look like who walk the streets; how the music sounds in the pub during a jazz-concert and many things more. Obviously for the purpose of navigating through the city, in most cases, the ignored information is irrelevant. It would, nevertheless be false to simply consider that map as a mere abstraction of that city. In fact important usage characteristics of that map result from characteristics it does not inherit from its original: That map can be folded, it can be put into one’s pocket and you can have it on you when you walk the streets in order to get to some particular location. None of these characteristics is a characteristic of the city Torshavn. While it is a small city you cannot fold it, put it into your pocket or carry it around when you walk the streets and are looking for a pub. The example suggests that models in general have characteristics that are not derived from their original. This point has already been made by Stachowiak. He, however, to my knowledge, fails to attribute a model’s usability chiefly to its excess characteristics. These characteristics are incorporated into the physical or logical design of the model and allow for a specific intended use. It is therefore wrong to consider models merely as abstractions. This analysis also shows that abstraction and modeling are in fact very different operations. The above example of course is no proof that models are not just mere abstractions. However, a related argument is this: if a model would be just an abstraction then it would have characteristics of its original only. Therefore one could do with the model exactly what one could do with that abstraction. That would render the model superfluous since all that could be done with it could already be done with that abstraction. Models are thus not just abstractions. They are abstractions injected into some “material” that enables us to make the intended use of the information as represented by that abstraction. Models are chiefly representations and successful models are handy representations. The purpose of these representations usually is either to get hold of the required abstraction or to make use of it. It is, by the way, not at the liberty of the modeler to not represent the model at all. The model is supposed to be shared by the modeling

4 The capital of the Faroe Islands.
5 That point in a less convincing way has been made earlier, [10].
community. It thus has to be represented. That gives the modeling community, however, the chance to represent it appropriately.

It is not difficult to see more clearly some of the differences between modeling and abstraction. Let us understand an abstraction $A$ of a thing $B$ in an educated but perhaps pre-scientific sense as something that results from $B$ by ignoring certain features of it. Then we find that a claim that $A$ is an abstraction of $B$ can actually be checked and either be confirmed or refuted. However, the utterance that $A$ is a model of $B$ cannot be refuted. One might, at best, be able to show that $A$ is not a good-enough model of $B$. Since no structural or other kind of similarity needs to exist between model and original one cannot falsify the mentioned utterance. Moreover, abstraction frequently is used to organize theoretical knowledge or concepts. This, however, is not a typical use of models. Their characteristic of being a vehicle for empirical knowledge makes that use sort-of difficult. It is, however, true that abstraction is a mechanism for getting hold of models. In algebra, for example, one can ignore certain aspects of a group and end up with a semigroup. One can then use the group as a model of that semigroup. Doing so may contribute to an improved understanding of the impact of variations of the group axioms.

Another related example takes us back to modeling languages: In object modeling it is common to model the behavior capacity of an object as a state chart. A state chart essentially is a finite directed and labeled graph. The vertex set of that graph is the set of so-called object states. The state chart has edge labels. These represent events and activities, respectively. An event-activity pair $(e, a)$, that is the label of an arc $(s, t)$, means that if event $e$ occurs and the object is in state $s$ then the object attains state $t$ and while doing so performs activity $a$. First of all one notices that in general an object will not have any such vertices, directed arcs or labels. Therefore in general one cannot consider such a state chart as a mere abstraction of an object. The state chart certainly has characteristics of an abstraction of an object. It, however, also has characteristics of a representation of an object. I can make it more obvious that a choice took place of the things that were added so as to simplify modeling and model use. It is, for example, possible to consider an object type $O$ as an entity type $E$ to which a weak entity-type “behavior capacity” has been associated by means of a dominating “has” relationship type. The key attribute of the type “behavior capacity” would be a set-valued composite attribute whose components are “state”, “event”, “action” and “follow up state”. The meaning of an instance $(s, e, a, f)$ of that attribute easily follows from the explanation of the state chart given prior. Obviously any further aspects such as “initial”, “current” or “terminal” state can be added to that weak entity type. There is no information concerning the modeled thing one can get out of the state-chart-model that cannot also be obtained from the weak-entity-type-model and vice versa. Consequently the state-chart-model just adds “syntactical sugar” to the weak-entity-type-model. That syntactical sugar thus has nothing to do with the model original. It has to do with how modelers and model users want to work on and with such models. The visual formalism
used for creating and representing state charts makes the state-chart-model more appealing for human modelers than the weak-entity-type-model.

Also the object identity may be modeled in the ERM. Only a weak entity type is needed that represents the object value and a dominating entity type with a single attribute, i.e., an abstract identifier representing the object identity. Also message passing processes can be represented in the ERM. The same goes for methods. A method (in perhaps a somewhat simplified form) is a four-place-predicate \( \text{method} \times \text{object} \times \text{input} \times \text{output} \): It is true with regard to a method \( m \), an object \( o \), an input \( x \) and an output \( y \), if \( o.m(x) = y \), i.e., the method \( m \) of the object \( o \) on input \( x \) responds with the output \( y \). Taking the object identifier \( o \) as an implicit parameter the method \( m \) can be specified as a set-valued composite attribute, i.e., the set \( \{ (x, m(x)) : x \text{ is admissible} \} \). It follows that object modeling languages do not improve on the expressivity of the ERM. Consequently the UML surplus over the ERM is syntactical sugar. The superiority of, say, the UML over the ERM is one of convenience of modeling and model use. This also suggests that models are not mere abstraction of their originals.

\[
\begin{align*}
\binom{n + 1}{k} &= \binom{n}{k} + \binom{n}{k-1} \\
&= \left(\frac{n + 1}{k}ight) = \left(\frac{n}{n-k} \right)
\end{align*}
\]

To illustrate further the importance of the representation I briefly discuss Pascal’s identity, i.e., for integers \( n \) and \( k \). It is not difficult to prove it. My point here is not about proving it. Rather I point out how representation can aid to discover and memorize it. As is well-known, the so-called binomial coefficient \( \binom{n}{k} \) equals the number of \( k \)-element subsets of an \( n \)-element set. It also is well known that \( \binom{n}{k} \) satisfies the binomial theorem: \( (a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k \). If one, for whatever reason, lists instances of the binomial theorem below each other centered in the right way then Pascal’s identity becomes so apparent that it is hard to overlook:

\[
\begin{align*}
0 : & \quad 1 \\
1 : & \quad a + b \\
2 : & \quad a^2 + 2ab + b^2 \\
3 : & \quad a^3 + 3a^2b + 3ab^2 + b^3 \\
4 : & \quad a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4 \\
5 : & \quad a^5 + 5a^4b + 10a^3b^2 + 10a^2b^3 + 5ab^4 + b^5 \\
6 : & \quad a^6 + 6a^5b + 15a^4b^2 + 20a^3b^3 + 15a^2b^4 + 6ab^5 + b^6
\end{align*}
\]

On the left-hand-side of this table there, in each row, the power \( n \) is listed (followed by a colon) to which the sum \( a + b \) has to be raised to obtain the sum in the \( n \)-th row of the triangular schema. The diagram makes it easy to focus on the binomial coefficients that have to be put in relation to get Pascal’s identity for a given exponent \( n \). Obviously the coefficient of \( a^{n+1-k}b^k \) is the sum of the coefficient on the left and on the right of \( a^{n+1-k}b^k \) in the row preceding it.
That, however, means: \((n+1 \choose n+1-k) = n \cdot (n\choose n-k) + \left(\begin{array}{c} n \\choose \ n-k\end{array}\right)\), at least as far as the above table goes. Certainly that is an indicator for Pascal’s identity to hold generally. Of course, as soon as one suspects the formula one can prepare further rows of the diagram in order to verify the related hypothesis. It is, bye the way, a quite natural idea to consider the instances of the Binomial Theorem that way. This is because of the commutativity of the “+” operation in these instances of the theorem there should be apparent a symmetry in terms of \(a\) and \(b\). To check out this symmetry it is a natural idea to arrange the formula so they are centered around the column headed 1, as this is the column in which the exponents of \(a\) equal the exponents of \(b\) in the terms \(a^x b^y\). The diagram suggests the in fact well-known identity: \(\left(\begin{array}{c} n \\choose \ k\end{array}\right) = \left(\begin{array}{c} n \\choose \ n-k\end{array}\right)\). By its use one can transform the Formula (1) into the more frequently used form: \(\left(\begin{array}{c} n+1 \\choose \ k+1\end{array}\right) = \left(\begin{array}{c} n \\choose \ k\end{array}\right) + \left(\begin{array}{c} n \\choose \ k\end{array}\right)\). I summarize that the discovery of Pascal’s identity simply can result from playing around a little with a particular representation of instances of the binomial theorem. Combinatorics is full of examples that could be used to provide additional support for my thesis that representation is key to models.

Modeling for computerization usually requires an inter-subjective meaning to be constituted by the models. Therefore the models must be represented. They must be represented in a medium that is accessible to the stakeholders. It moreover must be represented in a way such that the stakeholders can sufficiently make sense out of them. This implies that the models must have characteristics that depend on the stakeholders. With regard to these it is immaterial whether or not they are characteristics of the model original. Consequently, in general a model cannot be regarded as a mere abstraction of its original. Quite to the contrary a model is a representation that is tailored towards its users. It is in fact one of the key difficulties in modeling to know how to best aid those users and to find the appropriate representations.

Modeling for computerization at the first glance might seem to contradict the above argument that models are an abstraction injected into a certain material. It might not be that obvious what that material could be. However, models for computerization must be implementable on a computer. Thus they need to have certain logical characteristics that allow them to be implemented on a computer. These certainly, in general, are not characteristics of the original. In fact it is quite possible to implement models of non-computable phenomena on a computer. The material in this case are languages, formalisms, hard and orgware that permit the intended way of processing the model.

If one follows the above line of thought then one finds that the database area strongly depends on models being abstractions injected into a suitable material. First of all, for example, instead of representing UoD-things by patterns of microscopical size they could be represented by macroscopic marbles onto which, of course with an erasable pen, the information was written that should be recorded about these UoD-things. Obviously with regard to the concerned UoD the abstraction is exactly the same. The difference is the material into which that abstraction has been injected. The records, even if they are many, can be handled efficiently by a suitably defined computer program. In contrast to this it is not
clear how a large number of such marbles could be handled efficiently. Second, the well-known ANSI/SPARC-architecture of database management systems introduces logical and physical data independence. These independencies mean that, as far as a related database application goes, the information content of the model is independent of how that model actually is implemented on a computer. Therefore, of all the applicable representations, those can be chosen that are best at enabling the desired ways of handling the related information. The more appropriate representation may be the one that leads to queries being answered faster or the data quality being higher or similar.

4 Kinds of normative references as aided by models

Muller et al., [17], show that it is not uncommon for authors concerning conceptual modeling to consider models as descriptive or prescriptive. That way of conceptualizing models ascribes to them the capacity to describe or prescribe something else. This is at least a surprising stance since models are systems of concepts and thus by themselves incapable of enforcing anything. By implication that view denies modeling communities the capacity to govern the relationship between models they use and the originals thereof. A theory of conceptual modeling that puts the modeling community in charge cannot really live with that.

Wieringa has introduced the direction of fit concept [22]. The direction-of-fit concept is about the implied change in case an intolerable difference between model and original is detected. As Wieringa wrote for “descriptive models” the direction of fit is from the model to the original. That means if model and original do not fit each other well enough then the model must be changed so a fit of required quality is going to be established. He wrote further that the direction of fit for “prescriptive models” is from the original to the model, i.e. exactly reversed. The conception of these directions of fit is valuable. The conception of a model being descriptive or prescriptive is insufficient as I have shown above. These terms do not address a model property. Rather, they address the use that is made of the model. A model may be used to describe or prescribe something. That something may even be something other than its original. I can easily exemplify the point by considering a typical descriptive model such as a photograph of some important thing such as an ancient temple.\(^6\) In case that temple would be destroyed one might want to rebuild it. Then one might find it helpful to take that photograph as a prescription. It is the same thing, that photograph, that prior might have served as a description of the temple that due to such unlucky event could become the prescription of what to restore or rebuild. It is not an ontological status of model to be descriptive or prescriptive. It is instead a convention inherent in the reference established by means of that model. It is a simple extension of the example that shows that a model can be prescriptive with regard to one original and descriptive with regard to another. For example, the photograph of that temple might be used to build another

\(^6\) So I can be very clear about the problem I tolerate that the model considered in this example, strictly spoken, is not a conceptual model.
temple that looks just like the original temple. Moreover, design processes often require this in fact. In software development often a prototype is created. At some development stage that prototype may be taken as a specification of the product to be produced. Prior to that the prototype might just have been used as a description of what could be implemented.

To focus on the different kinds of reference that are exploited in software development the term reference mode [11] has been coined. In that paper a number of such modes has been identified. The idea of the direction of fit extends quite largely into determining the various reference modes. For example, the idealizing reference mode is how software process models are related as a pre-image to the activity of software developers. Under ideal circumstances one would proceed exactly as specified by that model, however, the world is not ideal and therefore there can be good reason for deviating from the ideal way of doing things. Therefore even if there is a vast difference between what the model instructs to do and what was actually done no corrective action might going to be taken. In fact the deviation might not even be a problem. And, if there is such a problem then it might have to be resolved on an entirely different level. Another example is the constitutive reference mode. In this mode a model of an essentially unexplored part of the reality of a group of people is taken to replace in their reality that pre-discourse version of the thing. This is the mode typical for philosophical ontologies.

So far in this paper three dimensions of modeling have been identified, namely expressivity of the modeling language, reference mode and material, i.e., the medium of the representation. These three do not exhaust the dimensions of modeling. The modeling system (rules for model construction) and the representation system (rules for model representation) need to be mentioned too. Furthermore the object of modeling, i.e. that which qualifies for being referred to, is another dimension of modeling. The same goes for who or what qualifies as modeler or as model user and the purpose of the model, respectively. These dimensions have already been pointed out by Stachowiak [20]. He has also identified the intended usage scenario, circumstances and time-span of use as further dimensions of modeling.

5 Resume

In this paper an activity-oriented view of conceptual modeling has been used. It was argued that theories of conceptual modeling may be important despite the fact that they at least partially belong to metaphysics and thus neither fully can be proven nor rejected. It was also pointed out that key tasks of modeling-communities are to refer to things in a universe of discourse, to characterize such things and to relate such things to each other. It was put forward that modeling in part is an experimental pursuit that aims at acquiring world-oriented (i.e., empirical) and modeling-community-oriented knowledge. The point made above in

7 Both these dimensions go back to [14].
the discussion of the importance of abstraction and representation for conceptual modeling is that this empirical knowledge only partially is obtainable by purely deductive capacities of human thought. It was then derived that representation, rather than abstraction, is key for modeling. The paper concludes with a discussion of kinds of normative references that are aided for by conceptual models. Doing so was justified by the observation that it is not uncommon to authors in the field to consider models as such as being prescriptive or descriptive.

References