

Towards a Theory of Models in Systems Development Modeling

Andrea Hillenbrand^a

Computer Science Division, University of Applied Sciences, Darmstadt, Germany

Keywords: Modeling, Model Theory, Model-driven Development, Development Methodology, Informatics, Logics.


Abstract: Despite decades of gaining experience in the development of software systems, the controversies on competing methodologies have not subsided. The pivotal element of reasoning and justification of any perspective taken thereon is arguably the recourse to models. Specifically, by means of a logical conceptualization of the notion of model-being, judgments on models can then be assessed whether they are justified, because as model judgments they are based on typical contextual constructive relationships. With such a conceptualization the potential lies in the realization of an epistemic architecture that emerges as a systematic structure of relations between models at object and meta levels. Each model application is then embedded in a systematic process during which a software system is developed. In this article, the combinatorics of model interweavements of such an epistemic architecture is presented, thereby providing the means to assess the development of a particular system as well as best practices and methodologies of systems development in general.

1 INTRODUCTION

The discourse on development methodologies continues unabated. Currently it seems to be dominated by the stance towards the agile paradigm as there are entire conferences dedicated to this topic (Stray et al., 2020). Large software companies involved in innovative product development usually have the operational luxury of implementing best practices of a conceptual methodology in unison with agile practices, like *DevOps* and its toolchain (Macarthy and Bass, 2020). Yet, many startups adopt speed-related agile practices exclusively, in a *lean startup* approach (Pantiuchina et al., 2017). Despite purported inadequacies, however, these startups do not necessarily sacrifice quality for speed more than other startups do. Oftentimes the model of profit margin works in favor of startups, arguably because agility draws in capable junior software developers whose value is not yet reflected on their payroll. In contrast to product development, the scope of project development also appears less risky as there exists more experience with regard to software system requirements and thus, speed-related agile practices are sufficient (Pantiuchina et al., 2017). Of course, a concrete answer whether agile developers are more prone to erroneous reasoning and biased decision-making (Mohanani et al., 2020) or the first to realize and reverse flawed thinking when focusing in-

cessantly on interactions, working software, and customer wishes (Dingsyr et al., 2010) has to be given by domain experts. Regardless, any concrete answer depends on the role *models* play during the development of a system, either their explicit development and conscious, logical usage contributing to *model adequacy*, or their implicit, though through experience routinely established usage, or their rather vague, unconscious usage, in which case *model adequacy* is potentially in jeopardy. Thus, any perspective is taken with regard to the role of models in systems development. It is claimed in this article that justified reasoning and taking a substantiated perspective are only guaranteed to be meaningful with recourse to models. This article explains why decision-making during the development of a software system should be justified along the lines of judgments about models in respect of their *adequacy and intent*—and in retrospect, decisions are defended or corrected better in that same respect.

Be the current frontline of development methodologies as it may, in any case judgments on how to develop systems need a common conceptualization of the underlying notion of models in order to discuss, evaluate, and compare concrete development decisions as well as competing methodologies. With the *Model of Model-being* (Mahr, 2009), Bernd Mahr put a logical conceptualization forward by which model judgments can then be assessed whether they are justified, that is, if they are based on typical contextual constructive relationships. Sadly, he was not able to

^a  <https://orcid.org/0000-0002-1063-5734>

complete the research on the *Model of Model-being* in his lifetime. In Section 3 of this article, this conceptualization is revisited. The potential of the conceptualization of Mahr's *Model of Model-being* now lies in the realization of the epistemic architecture of constructive relationships which emerges as a systematic structure of relations between models at object and meta levels. The epistemological question, which remained open and is now addressed in this article, is how these contextual constructive relationships of the involved models can, and can only, be interwoven. This clarification facilitates the discussions how any one model application is embedded in a systematic process during which a software system is developed. In this article, the combinatorics of model interweavements of an epistemic architecture is presented, thereby providing the means to assess the development of particular systems as well as best practices and development methodologies in general.

Contributions. This article contributes:

- The logical conceptualization of systems development modeling based on the *Model of Model-being* is completed through the epistemic architecture of the constructive relationships between models at object and meta levels.
- This epistemic architecture is clarified by means of discussing the combinatorics of interweavements with abstract logical means, including the composition and superimposition of models, and the application of meta models.
- This facilitates a methodological conceptualization of systems development modeling as a systematic process of creating, applying, and interweaving models, thereby forming a methodological basis of the best practices and methodologies of systems development in general.

Outline. Section 2 recognizes related work. Section 3 revisits Mahr's *Model of Model-being* as a logical conceptualization of the notion of model-being. Based on this, the combinatorics of model interweavements is presented in Section 4, completing the conceptualization of systems development modeling. Section 5 concludes this article.

2 RELATED WORK

Bernd Mahr's late work focused on the notion of judgments (Mahr, 2010b), the *Model of Model-being* (Mahr, 2009; Mahr, 2015; Mahr, 2010c), and a model of conception (Mahr, 2010a). An issue

of a journal was dedicated to the *Model of Model-being* (Mahr, 2015) including many critiques as well as responses. The complete works of Mahr are in the process of being published (Robering, 2020).

On the subject of conceptual modeling in computer science Thalheim has published prominently, for instance in (Thalheim, 2013; Thalheim, 2010; Thalheim, 2011). The semantics of models is discussed in (Kralemann and Lattmann, 2013), among many other noteworthy works left out here for the sake of brevity. The notion of design rationale in software engineering is investigated in (Burge et al., 2008), which instantiates the presented conceptualization consistently as a methodological basis of systems development. In (Boronat et al., 2009), semantically well-founded notions of a multi-modeling language and of semantic correctness of model transformations are proposed, to which the presented systematic process how model are embedded during the systems development fits in. In (Pastor and Ruiz, 2018), the sound software production process based on conceptual modeling is detailed going from the initial requirements model to the final application code through a well-defined set of conceptual models and transformations between them, with which the following discussion is consistent.

3 MODEL OF MODEL-BEING

Teaching computer science relies on the use of models. Generally speaking, models serve as conveyors of knowledge shared by those who take an interest in the facts of the matter being conveyed. It is argued here that models appear constitutive as they form the methodological basis of a science. But how do models do this? It turns out that what justifies perceiving something as a model provides an insightful answer, because a model judgment to be justified through logical reasoning allows that models are being *consulted* as conveyors of knowledge. In the tradition of philosophy, the *the act of a model judgment* is determined by that which in a model judgment is being modeled, the *intentionality*. It addresses the age-old question of the relationship between the outside world and the human perception of it. Although it seems far-fetched, intentionality is of crucial importance to modeling, because any scientific statement can only be understood as qualified networks of judgments about models in respect of their *adequacy* and *intent* (Mahr, 2009, p. 366). Mahr's *Model of Model-being* sheds light on what the judgment of model-being implies, thereby equipping us with the understanding of what it is that we do in systems development modeling.

In this section, the role of models in systems development is introduced in 3.1, followed by the discussion of the judgment of model-being in 3.2, and concluding with the logic of models in 3.3.

3.1 Model-driven Systems Development

In the tradition of systems development, models appear constitutive in particular as they provide the means to assert statements of possibility in theory and in practice, for instance, when prototypes are being developed as a proof of concept. This, broadly speaking, corresponds to the notion of validity in model theory, where certain propositions are considered valid with regard to certain models. It seems that for any kind of development, any constructive act toward a realizing a goal, we need models to put our thoughts in order and to *reason* in a certain direction. What this reasoning entails is going to be discussed in 3.3. In any case, the direction is naturally given through the intent when developing something, as this connects the factuality of the past experiences and the possibility of a future reality, both becoming accessible in the present through models.

In the context of systems development, any serious activity is directed toward providing an answer to the leading question: *Does the system S comply with the requirements for its application?* (Mahr, 2009, *ibid.*) An answer that addresses the future reality of the system application fulfilling the requirements can only be given using models, because models incorporate adequacy and intent referred to by the leading question in systems development. For the activities during the systems development, aimed at fulfilling the requirements for its application, the future system is accessible as a model only. In other words, the key characteristic in the development of a system is that the requirements to be complied with can only be formulated in respect of models anticipating the scenarios of the prospective system application. Even when a system has been put into operation, the models still remain models in respect of the system requirements. As such, many models live on, so to speak, and form the methodological basis of a science.

For complex systems to be developed, the process of creating models in respect of fulfilling the system requirements is oftentimes inaccurate. The crux lies with a dilemma: Concluding from a necessarily finite system description and from a necessarily finite number of application tests to a potentially infinite future system behavior is an uncertain inductive conclusion. Furthermore, the context of the future system application is oftentimes unknown. Not to put too fine a point on it, the problem can be aggravated through biased

or self-serving decision-making or questionable experience of the developers. Yet, inductive reasoning, through its generalizing conclusion from premises of particular incidents, has enormous potential and is constitutive in the creation and application of models, but it also has a potential to introduce errors as its conclusion cannot be formalized within a language rich enough for the purposes at issue here—a dilemma system developers should be aware of.

3.2 The Judgment of Model-being

A model judgment is indivisible, i.e., a subject decides whether to perceive of an object as a model, or *conceive* of it as a model consistent with Mahr's terminology (Mahr, 2010a). This judgment depends on a subject's conception of the object in a certain context. Thus, Mahr distinguishes in the *Model of Model-being* between the purely mental model μ and the model object M by which μ is represented (cf. to Figure 1). Now, the identity of an object as a model depends on two constructive relationships, depicted in Figure 1 horizontally, into which the model object enters according to the conception of the judging subject: First, a model object is *created* via a constructive act of production, selection, role assignment, abstraction, or mapping, on the basis of an initial object, for instance, a prototype, a set of requirements, or something that is *observed* (Mahr, 2009, pp. 377/8). This constructive act can be thought of or actually performed, also through an act of speech when a role of an object is assigned. Once the model object has been created, the second constructive act of a model judgment is the *application* of the model. As depicted in Figure 1 from left to right, in the act of model creation from an initial object A , the model object M is the resulting object, whereas in the act of model application to a resulting object B , the model object M is the initial object. Hence, a model μ can be viewed as both a model *of something* and *for something*. This dual role of M justifies viewing it as a model μ , the perspective of which is highlighted in red in Figure 1.

Through the application of the model, certain qualities are transferred with which M is loaded in its model capacity, so to speak. The *cargo* χ of the model, which was worked into M , can be unloaded through the model application, in Figure 1 highlighted in blue. The identity of χ depends on M as well as it depends on μ (Mahr, 2009, p. 378). The judging subject has *decided* for M that it carries the cargo χ adequately when serving as a model. Hence, one can distinguish two *modeling perspectives*, the perspective of model creation μ and the perspective of model application χ . Note that even when a model has not

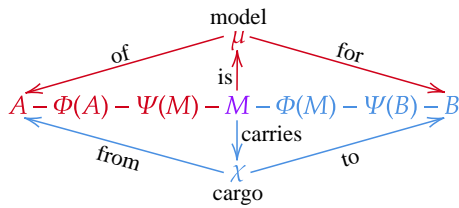


Figure 1: The logical structure of the constructive acts of the model judgment: Deduction Φ , transformation, and induction Ψ ; adapted from (Mahr, 2009, pp. 383).

yet been applied, it can still be a model, because it has been created with a certain model capacity intended to be used in order to transfer certain qualities. This is the case, because M has been created from the perspective of μ as a model *of and for* something. However, if it is logically impossible for an object M to be applied, for instance, in case of contradictions, then it cannot be judged a model in this conceptualization.

3.3 The Logic of Models

The two constructive acts in the judgment of model-being can be characterized as follows. Both constructive relationships, $A-M$ and $M-B$, are similar in the sense that whatever influence is exerted on a resulting object, here M and B , must be identifiable in an observation on the initial objects, here A and M . In this subsection, M and B are abbreviated by Y , and A and M by X . $\Phi(X)$ refers to a set of valid assertions *observable on X* . Accordingly, whatever has emanated from X exerting an influence on Y must be identifiable in an observation on Y , henceforth called $\Psi(Y)$. The model object M has the role of a mediator between the two constructive relationships. The set of valid assertions $\Phi(X)$ emerges from the initial object X through an observation on X , the relationship of which is denoted by $X-\Phi(X)$. The set of valid assertions $\Psi(Y)$ emerges from $\Phi(X)$ through a transformation, denoted by $\Phi(X)-\Psi(Y)$. $\Psi(Y)$ can be understood as *requirements* directed towards the resulting object Y , which emerges through a realization of the requirements $\Psi(Y)$ in Y , denoted by $\Psi(Y)-Y$. In Figure 1, the complete sequence of the two constructive acts of model creation and application, $A-M$ and $M-B$, are instantiated as follows.

These constructive relationships, according to which a resulting object emerges from an initial object, can be described further by using *abstract logic* (Beziau, 2005). A *logical theory* is a set of propositions in a formal language that is closed under the consequence relation. If the objects X and Y are replaced by their theories $\text{Th}(X)$ and $\text{Th}(Y)$, i.e., by the sets of propositions valid in X and Y , resp., then $\Phi(X) \subseteq \text{Th}(X)$, and $\Psi(Y) \subseteq \text{Th}(Y)$. Thus, the

object X and its theory $\text{Th}(X)$ correlate in the sense that within the scope of the logical language, nothing else can be known about X than what is already entailed in $\text{Th}(X)$. The same holds true for Y and $\text{Th}(Y)$. Hence, every observation in $\Phi(X)$ is a consequence of $\text{Th}(X)$ and valid in X , and every requirement in $\Psi(Y)$ is a consequence of $\text{Th}(Y)$ and valid in Y . Therefore, an interpretation of the relationship $X-\Phi(X)$, in Figure 1 occurring as $A-\Phi(A)$ and $M-\Phi(M)$, can be identified as a logical *deduction*. Analogously, the realization $\Psi(Y)-Y$, in Figure 1 occurring as $\Psi(M)-M$ and $\Psi(B)-B$, is identifiable as a logical *induction*. Hence, the logical structure of the relationships between $A-M$ and $M-B$, created through two constructive acts, can be viewed as a sequence of an act of deduction comprehended as an observation, a transformation, and an act of induction comprehended as a realization, for each constructive act. This is what is called the logic of models in this context (rf. to (Mahr, 2009, pp. 380-385) for an extensive discussion).

By this notion of a logic of models, a double sequence of two constructive acts is inherent in the process of modeling. Obviously, this is a powerful concept, because an intermediate constructive act of a model creation facilitates the exploration of possibilities, the conception of scenarios of future applications, and the demonstration of a possible solution to be implemented. Earlier in this section, it is referred to this as activities during the systems development aimed at fulfilling the requirements for its application where the future system is accessible as a model only. An intermediate constructive act of a model creation also makes decisions in the development process comprehensible, the outcomes of which predictable and defensible as decisions are based on judgments about models in respect of their *adequacy* and *intent*. In this sense, Mahr's Model of Model-being, as a general model, serves as a condition of correctness justifying a model judgment. (Mahr, 2009, p. 385). This can be extended to justify decision-making in the development process, where decisions, that are based on justified model judgments, are themselves justified, which is currently being researched by the author.

In order to motivate the following Section 4, imagine that there is a shortcut possible from observations on A , that is from $\Phi(A)$, to requirements on B , that is to $\Psi(B)$. A shortcut seems feasible if B is realized from A in one constructive act, but all the complexity usually carried by a model cargo would have to be worked into the transformation from $\Phi(A)$ to $\Psi(B)$. Though not impossible, it seems much more plausible that models would be present implicitly facilitating this transformation so

that the complexity of $\Phi(A) - \Psi(B)$ is manageable. Then, this transformation step would integrate a transformation model in the development process, i.e., $\Phi(A) - \Psi(M) - M - \Phi(M) - \Psi(B)$. Thereby, it would be reconstructed what modeling is about: It makes the reflective and anticipating activities more manageable and provides the possibility to formulate the observations and requirements in a common language. The implicit modeling would be made explicit. The explicitness of the model judgment allows an independent consultancy of the model from the perspective of χ thereby facilitating a validation of the model judgment in terms of its condition of correctness. Now, imagine further, that models are being created and applied at different levels of abstraction. Modeling at different levels of granularity, thereby creating a structure of constructive relationships, seems not feasible but rather prone to errors. While implicit routine modeling may be conceivable with regard to the usual canon of models, which is commonly taught in computer science, however, a discussion of the combinatorics of the model interrelationships sheds light on what happens at different levels of abstraction in case that it is not routine.

4 INFORMATICS OF MODELS

In order to fully understand how models are used in systems development, the combinatorics of what Mahr hinted at by the *interweavements of model interrelationships* is discussed in this section. As explained, models appear constitutive as they form the methodological basis of a science. Based on this, the *informatics of models* can be viewed as the methodological practice of interweaving canonical models as used in the development of software systems. Having realized what a judgment of model-being implies, the potential of the informatics of models now lies in a realization of the *epistemic architecture* of the constructive relationships which emerges as a systematic interweavement of relations to other models at object and meta levels (Mahr, 2009, p. 397). The application of one model is then embedded in a systematic process of applying a range of models on all levels during the systems development.

The interweavement of model interrelationships can come about in three ways, that is the *composition* and *superimposition* of models, and the *application of meta models*. An analysis of the combinatorics of model interweavements is presented here for the first time. The influences of the combinatorics of model interweavements on decision-making is currently being researched by the author. In the following, the

three possibilities of interweavements of model interrelationships are discussed, the *composition* of models in 4.1, the *superimposition* of models in 4.2, and the *application of meta models* in 4.3.

4.1 The Composition of Models

The composition of models represents the linear development and can be distinguished into two cases depending on whether the constructive relationships of the involved models overlap or not. In the latter case, the resulting object of a model application is, in turn, the initial object of a model creation, which is depicted as *Case 1* of Figure 2. This combination is referred to here as *complete model composition* and can be viewed as generic development process. If the system to be developed is particularly complex, then many consecutive steps of this process may be necessary to finally implement a system in its production setting. Some of steps can be intermediate models serving as auxiliary models. In an evaluation process, an auxiliary model could represent the changes to be made in order to improve the preceding model in this development. One example of such a normative-actual comparison in software engineering is the *system analysis*.

In *Case 2* of Figure 2, the application of one model and creation of a subsequent model overlap, which is possible because their constructive acts match in terms of their deduction, transformation, and induction. Note that the result of a model application is again a model. The transformation $\Phi(M_i) - \Psi(M_{i+1})$ here is a process converting observations of one model object to requirements of another model object. Precisely, it transforms the set of assertions $\Phi(M_i)$ valid in M_i to the set of assertions $\Psi(M_{i+1})$ valid in M_{i+1} . In a transformation, the expressiveness of $\Psi(M_{i+1})$ remains the same or becomes more or less expressive, consistent with the claim of universality of the *Model of Model-being*. Note that neither set is required to be closed under the consequence relation since this is not required of Φ or Ψ in general. The case is canonical when M_{i+1} is a *better* model (better in respect of the conception μ_{i+1}) compared to M_i , but despite the transformation M_i is still recognized as predecessor of M_{i+1} .

The *overlapping model composition* can come about by two motivations: First, the application of M_i is intended to be used in the constructive process of creating another model M_{i+1} . Here, applying M_i facilitates creating M_{i+1} . The observations made on the preceding model object are the observations on the initial object of a model creation. The preceding model's cargo allows such usage or even intended the

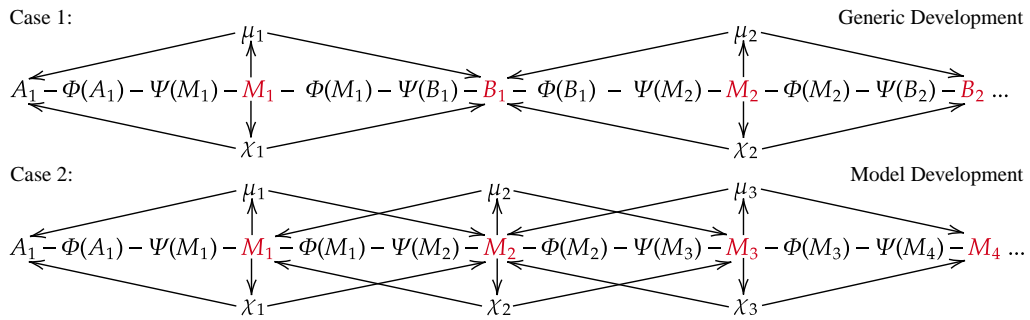


Figure 2: Complete and Overlapping Model Composition: Generic Development and Model Development.

usage of applying the model to result in a succeeding model. Another motivation could be that M_{i+1} is an improvement on M_i , where M_i is the precursor of M_{i+1} . Epistemically, though, this can both be viewed as model development. The ambivalence to view M_i as a facilitator for and a precursor of M_{i+1} also applies to the complete model composition, where the applicate B_i is the initial object in the creation of a subsequent model object M_{i+1} .

For instance, *Case 2* allows these interpretations: *Recursion* as such could be viewed as model development, where M_{i+1} is the model that emerges from one step of a recursive application process prescribed in M_i . This recursive application process continues until the last model M_i is reached when the termination condition applies and the recursion ends with the applicate B_i , the result of a recursively applied function defined in M_1 . A second, degenerated form of modeling would be *stagnation*, where the transformation of $\Phi(M_i) - \Psi(M_{i+1})$ is an isomorphism so that M_i would equal M_{i+1} in so far that no cargo would be gained except for a potential renaming.

4.2 The Superimposition of Models

Two cases of the superimposition of constructive relationships can be distinguished depending on how the constructive relationships overlap. In *Case 1* of Figure 3, a model object is the resulting object of two (or more) model creations. This can happen during a constructive act when an integration process takes place

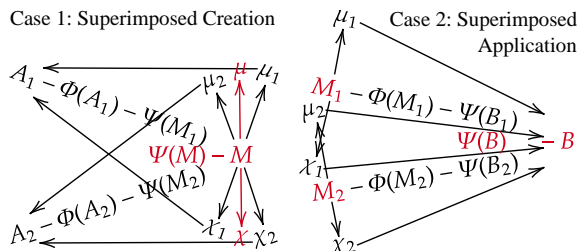


Figure 3: Model Creation and Application Superimposition.

of different observations, $\Phi(A_1)$ on A_1 and $\Phi(A_2)$ on A_2 . Instead of being transformed to different sets of requirements for different models, they are transformed to the set of valid assertions $\Psi(M)$ which shall be observable of M , i.e., to the requirements $\Psi(M)$ for the integrated model object M . This transformation has to be consistent, which means that contradictory requirements have to be resolved during the transformation so that the set of valid assertions $\Psi(M)$ can be realized into M . Accordingly, the modeling perspectives μ_i and χ_i both change to be model and cargo of an integrated M , implying integrated model perspective μ and integrated cargo χ .

Integration can also occur as a superimposed application, as depicted in *Case 2* of Figure 3, when an applicate B is the resulting object of two (or more) model applications. In this case, the integration comes about as transformation when two different sets of observations $\Phi(M_1)$ and $\Phi(M_2)$ are transformed to that which shall be observable of B , the integrated requirements $\Psi(B)$. This transformation has to be consistent as well, which means that contradictory observations on the model objects $\Phi(M_1)$ and $\Phi(M_2)$ have to be resolved during or after the transformation so that the set of transformed and integrated valid assertions $\Psi(B)$ can be realized into B . If the models M_1 and M_2 are already mutually consistent submodels of M , then the integration M is straightforward and has probably been anticipated. Therefore, the two forms of model superimposition are both interpretable as integration processes.

4.3 The Application of Meta Models

A meta model is a model whose resulting object of its application is an element of at least one of the two constructive relationships. This can occur in two forms depending on whether the resulting object of a meta model application is an *object* of the constructive relationships (*Case 1* of Figure 4) or whether it is used during a *transition from one object to the subsequent object* (*Case 2*). *Case 1* appears *constitutive* in

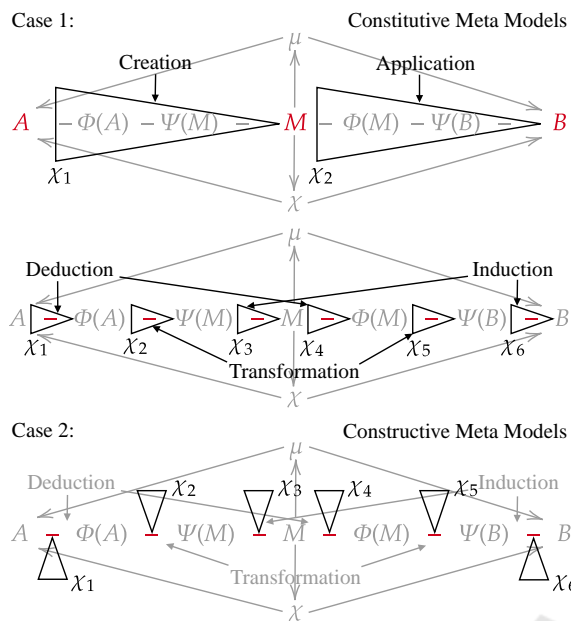


Figure 4: Applications of Meta Models.

nature, i.e., this kind of meta model is necessarily applied during modeling. Two subcases of *Case 1* can be distinguished: First, as such, model creation and application themselves are constitutive meta models for the construction of models, because their applications result in the final objects of the constructive relationships M and B . Second, the meta models representing notions of deductive and inductive reasoning as well as a notion of transformation make the creation and application of models possible, in which they appear as meta models. Here, the resulting object of the application of this constitutive meta model coincides with an object of the model construction in three ways: (1.) It coincides with the result $\Phi(A)$ of the observation on A or the result $\Phi(M)$ of the observation on M both via deductions unloading the cargoes χ_1 and χ_4 , resp. (2.) It coincides with the result of the realization M of the requirements $\Psi(M)$ or the result of the realization B of the requirements $\Psi(B)$ via inductions (χ_3 and χ_6), or (3.) with the results of the transformations into $\Psi(M)$ or $\Psi(B)$ (χ_2 and χ_5).

As depicted in *Case 2* of Figure 4, meta models are used during a transition from one object to the subsequent object of the constructive relationships. These meta model applications are constructive in nature, i.e., applying these meta models contributes their cargoes during the application of constitutive meta models as explained above. The model of reference (highlighted in gray in Figure 4) then gains the weights of the cargoes χ_i in the respective parts of the constructive relationships. For instance, the cargo of the meta model χ_1 is contributed during the deduc-

tive reasoning observing A and leading to $\Phi(A)$ while deducing assertions valid in A . In general, the resulting objects of the applications of constructive meta models coincide with the observation $A - \Phi(A)$ on A or the observation $M - \Phi(M)$ on M , the transformations $\Phi(A) - \Psi(M)$ or $\Phi(M) - \Psi(B)$, or the realization $\Psi(M) - M$ of the model object M or the realization $\Psi(B) - B$ of the applicate B . Consequently, in the creation and application of the model of reference, multiple meta models can be applied and thus contribute to the cargo χ of the model of reference.

5 CONCLUSION

Decision-making during the development of a software system can only be justified along the lines of judgments about models. The pivotal element of reasoning and justification of all perspectives taken is always their recourse to models, because models incorporate *adequacy* and *intent* through the typical contextual constructive relationships. With the presented conceptualization all aspects of systems development can be discussed in a common language and with the same underlying notion of models. Systems development can then be assessed by the way models are created, applied, and interwoven, and whether these model judgments are justified.

Taking up the discussion in the introduction on the usage of agile practices, the justification of which can be assessed based on the adequacy and intent of the involved models, whose detailed analysis is beyond the scope of this article. However, consider some other examples when model judgments are not justified due to lacking model adequacy: First published in 1975, Brooks et al. had already recognized that decision biases are quite common in software engineering (Brooks, 1975). They proved that initial inquiries are often based on inadequate information and that requirements *emerge* when prototypes are already developed, a classical case of disregarding the constructive relationships of modeling. Cross et al. coined the notion of the *solution-first bias* stating the tendency of designers to prematurely decide on a prototypical solution, before having fully grasped the problem, and to use this premature solution to explore the problem further (Cross, 2001). What he calls *fixation in design* fits into Nobel laureate Kahneman's notion of *cognitive ease*, the reason for the manifold biases in human decision-making (Kahneman and Tversky, 2000).

In the context of systems development, any serious activity is directed toward creating a system that complies with the requirements for its application. The future reality of a system application can only

be addressed using models, because models incorporate adequacy and intent referred to by the compliance of the requirements for the application of the system. Such a conceptualization has been introduced in this article. The above discussions constitute the potential of the informatics of models realizing the epistemic architecture of the constructive relationships which emerges as a systematic structure of relations between models at object and meta levels. The application of one model is then embedded in a systematic process of applying a range of models on various abstraction levels during the systems development. Then systems development modeling form a methodological basis and practice in respect of all abstraction levels throughout the informatics of models. Models, that have become canonical in this process, serve as conveyors of knowledge and systematize it into different levels of abstraction. With such a general methodological conceptualization based on the same underlying notion of a model a wide variety of topics can be discussed related to systems development, such as its best practices, business process modeling, the integration of multiple views in management systems, adequate or biased decision-making, or the conceptualization of fairness in machine learning.

ACKNOWLEDGEMENTS

This work has been funded by the German Research Foundation/Deutsche Forschungsgemeinschaft (DFG) – grant 385808805.

The author would like to thank Klaus Robering for constructive criticism of the manuscript.

REFERENCES

- Beziau, J.-Y. (2005). From Consequence Operator to Universal Logic: A Survey of General Abstract Logic. In *Logica Universalis*. Birkhäuser, Basel.
- Boronat, A., Knapp, A., Meseguer, J., and Wirsing, M. (2009). What Is a Multi-modeling Language? In *Proc. WADT'16*, volume 10644 of LNTCS, pages 71–87. Springer, Berlin.
- Brooks, F. (1975). *The Mythical Man-Month: Essays on Software Engin.* Addison-Wesley, Boston.
- Burge, J. E., Carroll, J. M., McCall, R., and Mistrik, I. (2008). *Rationale-Based Software Engineering*. Springer, Berlin.
- Cross, N. (2001). Design Cognition: Results from Protocol and other Empirical Studies of Design Activity. In *Design Knowing and Learning*, pages 79–103. Elsevier, Oxford.
- Dingsyr, T., Dyb, T., and Moe, N. B. (2010). *Agile Software Development: Current Research and Future Directions*. Springer, Berlin.
- Kahneman, D. and Tversky, A. (2000). *Choices, Values, and Frames*. Cambridge University Press.
- Kralemann, B. and Lattmann, C. (2013). The Semantics of Models: A Semiotic Philosophy of Science Approach. In *Proc. SDKB'13*, volume 4925 of LNCS, pages 50–69. Springer, Berlin.
- Macarthy, R. W. and Bass, J. M. (2020). An Empirical Taxonomy of DevOps in Practice. In *Proc. Euromicro SEAA'20*, pages 221–228.
- Mahr, B. (2009). Information Science and the Logic of Models. *J. SoSyM*, 8(3):365–383.
- Mahr, B. (2010a). *Intentionality and Modeling of Conception*, pages 61–87. Logos Verlag, Berlin.
- Mahr, B. (2010b). On Judgements and Propositions. *ECEASST*, volume 26.
- Mahr, B. (2010c). Position Statement: Models in Software and Systems Developm. *ECEASST*, 30.
- Mahr, B. (2015). Modelle und ihre Befragbarkeit. Grundlagen einer allgemeinen Modelltheorie. *Erwägen Wissen Ethik*, volume 26.
- Mohanani, R., Salman, I., Turhan, B., Rodríguez, P., and Ralph, P. (2020). Cognitive Biases in Software Engineering: A Systematic Mapping Study. *J. TSE (IEEE)*, 46(12):1318–1339.
- Pantiuchina, J., Mondini, M., Khanna, D., Wang, X., and Abrahamsson, P. (2017). Are Software Startups Applying Agile Practices? In *Proc. XP'17*, volume 283 of LNBIP, pages 167–183. Springer.
- Pastor, O. and Ruiz, M. (2018). From Requirements to Code: A Conceptual Model-based Approach for Automating the Software Production Process. *EMISAJ*, 13(Special):274–280.
- Robering, K., editor (2020). *Schriften zur Modellforschung*. To be published.
- Stray, V., Hoda, R., Paasivaara, M., and Kruchten, P., editors (2020). *Proc. XP'20*, volume 383 of LNBIP. Springer, Berlin.
- Thalheim, B. (2010). Towards a theory of conceptual modelling. *J. UCS*, 16(20):3102–3137.
- Thalheim, B. (2011). *The Theory of Conceptual Models, the Theory of Conceptual Modelling and Foundations of Conceptual Modelling*, pages 543–577. Springer, Berlin.
- Thalheim, B. (2013). The Conception of the Model. In *Proc. BIS'13*, volume 157 of LNBIP, pages 113–124. Springer, Berlin.