Electronic Communications of the EASST
Volume 30 (2010)

International Colloquium on Graph and Model Transformation - On the occasion of the 65th birthday of Hartmut Ehrig (GraMoT 2010)

Position Statement: Models in Software and Systems Development

Bernd Mahr

11 pages

Guest Editors: Claudia Ermel, Hartmut Ehrig, Fernando Orejas, Gabriele Taentzer
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Position Statement: Models in Software and Systems Development[1]

Bernd Mahr

mahr@cs.tu-berlin.de
Institut für Telekommunikationssysteme
Technische Universität Berlin

Abstract: The development of software and systems is, by its very nature, highly depending on models. In the software and system's design models play an important role, where they represent the constraining standards as well as the choices of ideas and perspectives applied in the systems modelling, implementation and technology. The general model of model-being, developed by the author, is briefly explained and is used to discuss model interconnections resulting from *model compositions* and *metamodel applications*. In this position statement it is claimed that the analysis of model interconnections, prerequisite to or underlying the software and systems design, should provide new insights into the designs architecture.

Keywords: software, design, model, model-being, model interconnections

1 Introduction

The choice of design in software and systems development is naturally constrained by mainly four factors: *first*, by the requirements and expectations on the properties and features of the intended systems future application and use, *second*, by the norms and standards to be met, *third*, by the quality and limitations of resources available for the intended systems modelling, implementation and technical realization, and *fourth*, by the reality of the social and technical environments, in which the intended system is embedded when being operated. However, the system's future applications, its modelling, implementation, technical realization as well as its environments are at the time, before the system is being developed, not directly accessible. At that time these constraining factors can only be identified and addressed by means of prospective models. And it is not only a fact that these constraints in the development process are to be mediated by models, it is also the likeliness of change, which affects the execution of the development task: it is not unusual that expectations and requirements on a system's

[1] This paper builds on and repeats some of the content of Bernd Mahr: *Information Science and the Logic of Models*, Journal of Software and Systems Modelling (2009) 8, Springer Verlag 2009, p. 365-383, (See also the German original: Bernd Mahr: *Die Informatik und die Logik der Modelle*, Informatik Spektrum, 32, 3, 2009, pp. 228 – 249). As a position statement it does not fully elaborate its concepts and assertions. Though what is said here is largely based on year long experience in software- and application development and on deep research on the question of modelling, the claims made do sometimes express expectations rather than experience.

application and use are being modified before the system is delivered; it is also common experience that resources for its modelling, implementation and technical realization do not stay stable while the system is being developed; and it is certain that the environments in which the system is being embedded, will not be in a constant state over time[2]. All models involved in the task of development are therefore to some degree unpredictable in regard to their future adequacy and trustworthiness. And there is yet another difficulty in system's development: when the system is being modelled and implemented, all matters of its functionality and design are to be expressed as features and properties of the system itself, which is to say that the models which capture requirements and expectations of the system's application and use and of all the other constraining factors in the development task, have to be encoded as features and properties of the system.

It would, however, be wrong to conclude from these observations, that the development of software and systems are impossible tasks. There are mainly two reasons why their development, despite of these difficulties, has a good chance of success: *first*, the fact that in practice systems behaviour is rarely judged on a predefined and completely rigorous basis. Users usually accept to adapt their expectations, activities and patterns of use to what the system is able to do, at least to a certain degree. And *second*, more than 60 years of experience in theory and practice of software development have lead to techniques and tools, which enable architects and developers to cope with the consequences of mediation and change. In the widest sense, these techniques and tools concern means of coding, abstraction and coordination, all being based on models and modelling techniques. It is, however, surprising that we have little knowledge about the principal conditions for something to be a model and about the activities constituent for model use, namely the activities implied by modelling and model application. And we can hardly say what in general counts as a good model and what does not. These deficiencies may not be severe if the modelling takes place in a disciplined context with a high level of standardisation, but they are definitely present with the general notions of model and modelling, and are also found to affect modelling in the fields of information and conceptual modelling[3], and in the many endeavours of computer based modelling.

Having observed this, the question comes up of what benefit it might be to gain deeper knowledge about the notion of model in general, namely how the tasks of software and systems development can benefit from it. And one might also ask, if there is a chance at all to clear the general notion of model, which is widely assumed to be indefinable. To respond to the second question first, there is much more to know about models and modelling than there is known today. But to acquire this knowledge one has to give up some of the epistemological customs of explanation: when thinking about models one can no longer avoid to constructively treating their subject and context dependency; and in order to seek for an answer to the question of *what is a model* in ontology or in some formal theory, one has to rephrase this question as to *what justifies the judgement that a given object is a model*, and answer it in the

[2] If application and use of a system makes a difference, which is what is intended by its provision, it will change its environment.
[3] Bernhard Thalheim: *Entity Relationship Modeling – Foundations of Database Technology*, Berlin und Heidelberg: Springer, 2000.

realm of logic; and finally one has to focus on the structure of contextual relationships characteristic for models rather than to try to find the kind of similarity which relates an original with its model[4]. To respond to the first question, it has first of all to be observed that it is generally impossible to restrict the notion of model to only certain familiar types or to ask for computer science owned notions of model and modelling. Software systems are applied in almost any field of science, engineering and daily life; the design of software, from an overall perspective, has therefore to deal with the modelling cultures and disciplines in all these fields. As a consequence, if software design is not wanted to be split into separate disciplines, we cannot avoid to either know about the different conceptions of model and modelling in these fields, or to develop a general conception of universal applicability. It is the authors position in this statement that a model of model-being can be conceptualized, which not only covers all known conceptions of models as particular fragments or specializations of the general conception, but which should at the same time be useful as a sensitive tool for analysis and design, not only in the case of modelling in general, but also in individual situations of model use.[5] It is the expectation that such an analytical tool should also yield deeper insights into the structure of model interrelations on which software is being built. And from these insights, it is expected, it should be possible to derive new techniques and tools for the tasks of software and systems development.

2 The epistemic pattern of model-being

The question of *what justifies a judgement that a given object is a model*, presupposes that the model-being of this object is the conclusion of a judgement for which there are grounds. If it is possible to phrase general conditions which are necessary and sufficient for such a judgement to be *acceptable*, one might take these conditions as an argument form, which in individual cases if properly instantiated justifies the judgement of model-being. The argument form of model-being is then seen to exhibit the logic of models in general, while its instantiations determine the logic of individual models. Since judgements are actions undertaken by subjects, the model-being of an individual object is not a property of the object in itself but is relativised by the subject dependency of the judgement which asserts it. And since the argument form of model-being yields necessary and sufficient conditions for acceptability and not conditions for defined or objective truth, the notion of model, conceptualized in terms of this argument form, is relativised by the subject dependency of accepting.

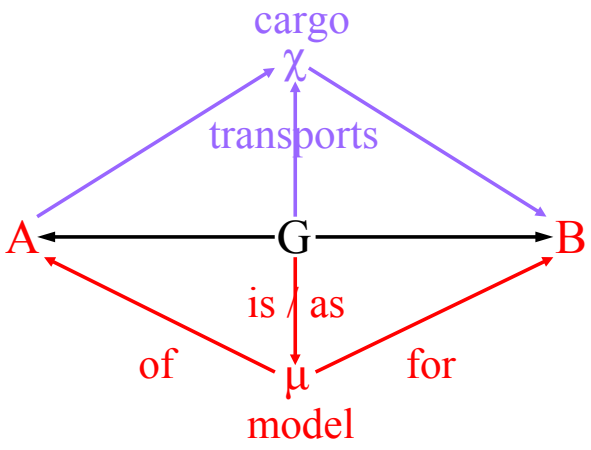
The conditions implied by the argument form for model-being cannot be based on inherent properties of the object judged to being a model. This is obvious from the fact that any object can be acceptably judged to being a model if it is only positioned into a proper context, for

[4] Herbert Stachowiak: *Allgemeine Modelltheorie*, Wien / New York: Springer, 1973.

[5] The model of model-being, for which this strong claim is made, has been developed in the last decade in interdisciplinary studies and projects by the author and his co-workers. See for example Bernd Mahr: *Modellieren. Beobachtungen und Gedanken zur Geschichte des Modellbegriffs*, in: Sybille Krämer, Horst Bredekamp (ed.): *Bild-Schrift-Zahl*, München: Fink, 2004, p. 59-86; Bernd Mahr: *Ein Modell des Modellseins. Ein Beitrag zur Aufklärung des Modellbegriffs*, in Ulrich Dirks, Eberhard Knobloch (ed.): *Modelle*, Frankfurt am Main: Peter Lang, 2008, p. 187-218; Reinhard Wendler: *Die Rolle der Modelle in Werk- und Erkenntnisprozessen*, Dissertation am Kunstgeschichtlichen Seminar der Philosophischen Fakultät III der Humboldt-Universität zu Berlin, Juli 2008; and (Mahr, 2009).

example into a context of production in which it takes the role of a prototype. And because there is no object which is a model by necessity, since it is always possible to position an object into a context in which there is no meaning of models at all, one has to conclude that the conditions implied by the argument form of model-being are context dependent.

If an object is judged to being a model, it is necessarily conceived of as a model by the judging subject. Assuming in general that to conceive of an object means nothing else but to identify the object's involvement in context relationships[6], it seems justified to defining the argument form for model-being as to being a complex of context relationship types. The pattern of these relationship types is then taken to characterise the situations, in which an object has the role of a model. This diagram depicts this pattern[7].



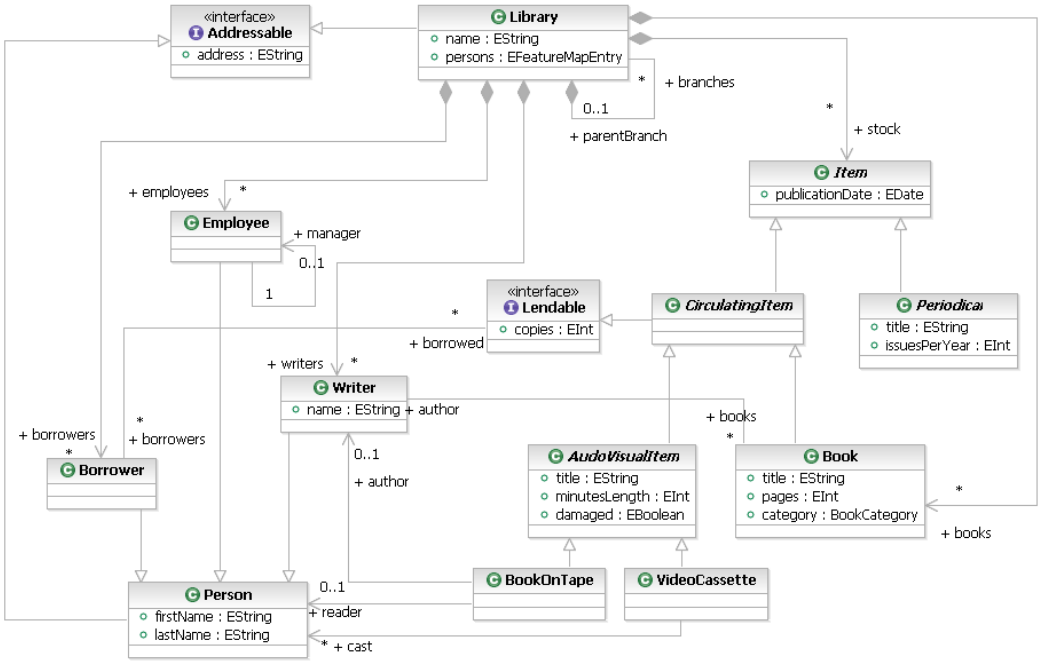
The diagram defines the argument form of model-being and is composed of epistemic object and object relationship types. It is therefore also called the *epistemic pattern of model-being*. The term *epistemic* is used here to indicate that objects and relationships of an instantiated argument form have existence only as intentional objects, i.e. as being conceived of by the judging subject. Objects of type A, G and B may be conceived of to be concrete or abstract, but the objects of type μ and χ as well as all relationships are necessarily abstract, i.e. their existence is independent from space and time.

The intended meaning of the epistemic pattern of model-being, which guides the instantiation of its object and relationship types, is the following:

[6] This assumption forms the basis of the *model of conception* developed by the author. The first thoughts on this model have been described in Bernd Mahr: *Gegenstand und Kontext – Eine Theorie der Auffassung*, in: K. Eyferth, B. Mahr, R. Posner, F. Wysotzki (Ed.): *Prinzipien der Kontextualisierung*, KIT Report 141, TU Berlin, 1997, p. 101 - 119. A set-theoretical study of the model is given Tina Wieczorek: *On Foundational Frames for Formal Modelling – Sets, epsilon-sets and a model of conception*, Aachen: Shaker, 2009. For a philosophical justification of the model see Bernd Mahr: *Intentionality and modelling of conception*, 2009, in: Sebastian Bab, Klaus Robering (Ed.): *Judgements and Propositions – Logical, Linguistic and Cognitive Issues*, Berlin: Logos, 2010, pp 61 – 87.

[7] For justification of this pattern see (Mahr, 2009) as well as (Mahr, 2008) and (Mahr, 2004).

1. In the context of this pattern an object of type G is called *model object* and an object of type μ is called *model*. An object of type G is not by its identity a model. It has to be distinguished from the model *as* which it is seen, because different model objects can represent the same model. The fact that an object is seen *as a model* assigns it the role of a model object and determines the relationship between itself and the model it represents.



Instance: A model, for example depicted as the above class diagram[8], which is its model object, can have different other diagrammatic representations as model objects.

2. A model is always a model of something, the type of which is here denoted by A. And the fact that a model is seen to be a *model of* something, determines the relationship between the model and that of which it is a model.
   Instance: The model which has the above mentioned class diagram as its model object, is a model of the university's library system.

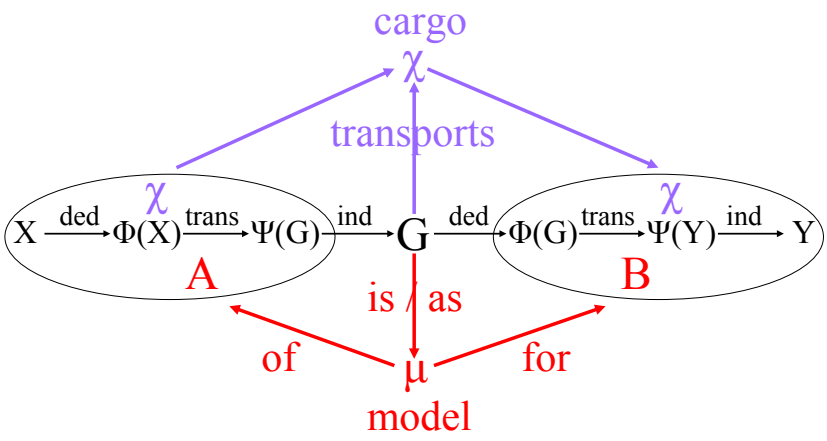
3. Composing the relationships *as-a-model* and *model-of* yields the fact that the model object is seen to be a model of A. This fact determines the relationship between a model object and that of which it is a model.
   Instance: The above mentioned class diagram is a model of the university's library system.

[8] http://help.eclipse.org/galileo/topic/org.eclipse.ocl.doc/references/examples/extlibrarymode, October 26, 2010.

4. A model is always a model for something, the type of which is here denoted by B. And the fact that a model is seen to be a *model for* something, determines the relationship between the model and that for which it is a model.
   Instance: the model which has the above depicted class diagram as its model object, is a model for the object architecture of the planned implementation of the university's e-library system.

5. Composing the relationships *as-a-model* and *model-for* yields the fact that the model object is seen to be a model for something.
   Instance: the above depicted class diagram is a model for the planned implementation of the university's e-library system.

6. In the context of this pattern an object of type χ is called *cargo*. For a model to make sense there must be something, named *its cargo*, which carries over from that *of which* a model object is a model, to that, *for which* it is a model. The cargo of a model is seen to be *transported* by the model object *from* that of which it is a model *to* that for which it is a model. This fact determines the three relationships between objects of type G and χ, A and χ, and χ and B.
   Instance: the above depicted class diagram transports a structure of components, component relationships and component owned operations, which are identified in the university's library system, to the implementation of the university's e-library system.



Looking more carefully at situations of model-being, it becomes apparent that the relationship types between A and G and between G and B show the same sequential structure. In combination they are derived from the following sequence of action types (also called *action sequence of modelling*):

1. an *observation* on an *initial object* of type X, resulting *observed facts* of type Φ(X),
2. a *transformation* transforming the facts of type Φ(X) to *requirements of type* Ψ(G) imposed on a model object of type G,

3. a *realization* of the requirements of type Ψ(G) by a *model object* of type G,
4. an *observation* on the *model object*, resulting *observed facts* of type Φ(G),
5. a *transformation* transforming facts of type Φ(G) to *requirements* of type Ψ(Y) on a *terminal object* of type Y,
6. a *realization* of the requirements of type Ψ(Y) by the *terminal object* of type Y.

An object of type A may now be of type X, Φ(X), or Ψ(G), and an object of type B may now be of type Φ(G), Ψ(Y), or Y. We can then speak of a *model of* an object, *of* an observation, or *of* requirements, and accordingly of a *model for* an object, *for* requirements or *for* an observation.

Coming back to the example of a university library system, the relationship between the model object (depicted as the class diagram above) and the object of type A (the existing library system), of which the diagram represents the model, can be explained as follows: It is derived from some kind of systems analysis in which the existing library system (instantiating the *object of type X*) has been studied (*observation*) in respect to what matters in view of the model to be created. This analysis results in the identification of relevant types of entities, relationships, attributes and actions (*observed facts*). These facts are then read (*transformation*) as items to be referred to (*requirements*) in the model object to be created. The class diagram (*model object*) is then developed to show these items (*realization*).

The relationship between the class diagram and the model object and the university's e-library system is similarly structured, in the sense that it also results from a sequence of observation, transformation and realization. The only differences are that the object on which the observation is made, is the model object and that the terminal object of type Y may, at the time when the judgement of model-being is made, be only a vision and not yet exist in reality. If the latter is the case, the sequence of observation, transformation and realization is only a thought. But if this thought was not justified to possibly be reality, it was difficult to argue that the class diagram was a model at all. It would then rather be an image.

In any real circumstance in which a compliant judgement of model-being is made, the epistemic pattern of model-being is instantiated by the judging subject. No matter if the judging subject[9] refers in his instantiation to existing things in reality or only to thoughts about a possible reality, it identifies objects which instantiate the types X, Φ(X), Ψ(G), Φ(G), Ψ(Y), and Y, and it identifies object relationships which result from the two observations, say α and β, the two transformations, say σ and τ, and the two realizations, say π and ρ[10], thereby

[9] The judging subject may be a human individual, but it may also be a group of individuals, a community or, even more abstract, a culture. On the other hand, a judging subject may also be a machine or a mathematical definition. Generally speaking, the judging subject is something by the authority of which the judgement in question is affirmed.

[10] These identifications need not necessarily be conscious. They constitute part of the context in which the model object takes its role as a model. The objects and object relationships identified also need not to have existence independent from the judging subject, as they may just be thought of or generated in the moment of judgement. Their identification is assumed to be valid as long as the judgement is valid in the subject's eye. Examples show that in practice this may be a fragment of a second or, in the other extreme, if the subject is not an individual, last for centuries, or even longer.

producing a sequence of actions (which is to be distinguished from the presupposed sequence of action types):

$$X - \alpha - \Phi(X) - \sigma - \Psi(G) - \pi - G - \beta - \Phi(G) - \tau - \Psi(Y) - \rho - Y$$

The judgement of model-being is then acceptable, if this sequence of actions is an adequate view on a defined or an identified situation of model-being.

Assuming that all objects with object types in this sequence are being replaced by their logical theories[11], one observes that observations are *deductions*, and that realizations are *inductions*. A model object is therefore at the same time the result of an induction and the source of a deduction. This dual nature of model objects can be seen as one of the most typical characteristics of objects which are being conceived of as models.

The epistemic pattern of model being is not a formal definition of what a model is. Such a definition would make sense in certain modelling disciplines, like the disciplines of set formation or graphs, but due to the mathematical nature of its constituents it would hardly meet the needs of practical model use in most of the sciences, engineering and daily life in a direct way. If a formal definition was to be given it would have to follow the structure provided by the pattern, and would have to define object and object relationship types as well as their possible instantiations as mathematical entities in some foundational theory, like category theory or the theory of sets. A mathematical definition of models would replace the judging subject by written formal conditions and would have to explicitly determine the criteria for relationships to be observations, transformations and realizations. Namely for transformations these criteria would either be restrictive, like the criteria for a function to be a homomorphism[12] or the criteria for a relationship to encode similarity, or they would be very general and therefore be weak in their expressiveness and determination. But the question of what the epistemic pattern conceptually is, if it is not a definition, can nevertheless clearly be answered: it is a model itself, i.e. it is an object which is to be judged as something by using the same argument form that is used for other judgements of model-being.[13] Its epistemological foundation is therefore not different to that of a mathematical definition which, to be strict, is also to be based on a (mathematical) definition of what a definition is.

3 Complexes of interconnected models

Anything constructed can be questioned for *what* it is, *how* it was constructed, and *by what means*. In the case of software, the question '*what*' asks for the systems architecture, its functionality and its technical realization; the question '*by what means*' asks for the techniques and tools applied in its modelling, implementation and installation; and the question '*how*'

[11] The theory of an object is the set of all sentences which are true of this object. A theory is therefore language dependent. For reasons of simplicity one might assume that the choice of this language is determined as part of the subject's judgement and also that it is the same for all objects involved.
[12] (Stachowiak, 1973), p. 140 – 159.
[13] Justification for the epistemic pattern of model-being as to being a model itself can indeed be given by using the epistemic pattern of model-being as an argument form. Evidence for this is provided in the articles (Mahr, 2004), (Mahr, 2008), (Mahr, 2009) and in other texts cited there.

asks for the concepts and models underlying its design. The distinction between the last two questions, however, is not exclusive, since also models are tools, and tools, like formalisms and languages, are representations of descriptive models. As a clarification it is therefore assumed that the question '*how*' is here intended to ask for the conceptual basis on which the system was built as a solution and for the ideas and perspectives taken when this solution was found. Choices of ideas and perspectives are choices of models. So to ask for ideas and perspectives is to ask for the models which have been applied in the systems modelling, implementation and use of technology. For example, access to a set of stored data can be implemented as an array, a list, a record, a stack, a tree, or the like; a system which integrates patient data distributed in a hospital and makes these data available in a hospital network to all having the right to their access, can be implemented as an information system, a communication system, an open distributed system, a service, an agent network, a peer to peer system or a cloud; and a component in a component based system can be realized as an object, a module, or a service. This is to say that prerequisite to or underlying any existing software and system is a complex of interconnected models which conceptualize the ideas and perspectives put together and combined to determine the software or systems design. In their relation to the design which itself is a model, these models are metamodels of the implemented software or system.

Generally, a judgement of model-being is never isolated. It is always embedded into a network of intentional relations which determine its meaning[14]. Examples show that the contextual environment of a model object is typically not only the complex of objects and object relationships, which the pattern of its model-being indicates, but that this environment also includes other models which are interconnected with the model at hand in specific ways. There are at least two elementary types of model interconnections, which can be distinguished: *model compositions* and *metamodel applications*.

Model compositions capture situations in which, for example, the initial object of a modelling action sequence is itself a model object, or in which a terminal object is the initial object of another sequence, or a model object is related to more than one initial object, observation or requirement, and the like. If the above depicted class diagram, for example, is not produced in a single step, but in a sequence of steps with intermediary results, like in model driven architecture (MDA), the situation of modelling is better described as a model composition. And the idea of model composition also applies if one wants to capture a situation in which the class diagram is not only representing a model of the University's existing library system alone but also includes concepts from other more innovative such systems.

M*etamodel applications* capture situations in which items in an action sequence, like an observation or the model object itself are constrained by other models in the sense that the constrained items result from an application of these other models. In the above sketched modelling of the library system this is the case with the use of UML as description formalism. UML constraints the model object of the library system by its concepts and expressiveness.

[14] John R. Searle: *Intentionality – An Essay in the Philosophy of Mind*, Cambridge: Cambridge University Press, 2004, p. 20 – 21 and 26.

And if some automatic programming tool is applied, which takes the class diagram as its input, this tool plays the role of a metamodel for the sequence of observation, transformation and realization in the relationship between the class diagram and the implemented e-library system.

Referring to the action sequence of modelling, which was introduced above, i.e. to

$$X - \alpha - \Phi(X) - \sigma - \Psi(G) - \pi - G - \beta - \Phi(G) - \tau - \Psi(Y) - \rho - Y$$

and using some self explaining notational conventions, the following examples abstractly denote such situations of model compositions and metamodel applications:

$$X–\alpha–\Phi(X)–\sigma–\Psi(G)–\pi–G–\beta–\Phi(G)–\tau–\Psi(Y)–\rho–Y–\alpha'–\Phi(Y)–\sigma'–\Psi(Z)–\pi'–Z$$

depicts the situation of a terminal object which is a model object itself,

$$\{(X_i–\alpha_i–\Phi(X_i)–\sigma_i–\Psi(G)_i–\pi_i) \mid i \in I\}–G–\beta–\Phi(G)–\tau–\Psi(Y)–\rho–Y$$

depicts the situation in which a model object G is a model having a set of initial objects

$$X–\alpha–\Phi(X)–\sigma–\Psi(G)–\pi–G–[X'–\alpha'–\Phi(X')–\sigma'–\Psi(G')–\pi'–G'–\beta'–\Phi(G')–\tau'–\Psi(\beta)–\rho'–\beta]–\Phi(G)–\tau–\Psi(Y)–\rho–Y$$

depicts the situation in which the observation β on the model object G is obtained by applying a metamodel with model object G'.

The complex of interconnected models, which conceptualizes the ideas and perspectives taken, namely by employing model compositions and metamodel applications, and which determines the design of a given software or system, may be seen as to being the software or system's *model architecture*. Frameworks, like security frameworks or specification frameworks, can then be identified as prescriptions of model architectures. They prescribe which metamodels are to be used in a modelling action sequence and how models are to be composed. By using the sequence of action types in the pattern of model-being it should not be difficult to build a formal setting for model compositions, which allows to specify particular model architectures and to define types of complexes of interconnected models in a systematic way.