A metamodel for the Unified Modeling Language: critical analysis and solution

Andrey Naumenko, Alain Wegmann

Systemic Modeling Laboratory,
Swiss Federal Institute of Technology – Lausanne.
EPFL-IC-LAMS, CH-1015 Lausanne, Switzerland
{Andrey.Naumenko, Alain.Wegmann}@epfl.ch

Abstract. Nowadays models, rather than code, become the key artifacts of software development. Consequently, this raises the level of requirements for modeling languages on which modeling practitioners should rely in their work. A minor inconsistency of a modeling language metamodel may cause major problems in the language applications; thus with the model driven systems development the solidness of modeling languages metamodels becomes particularly important. At its current state the UML metamodel suggests a significant area for improvement. In this work we present an alternative metamodel that was inspired by RM-ODP standard and that solves the problems of UML. RM-ODP was mentioned in UML specifications as a framework that has already influenced UML. Our metamodel was formalized, thus its resulting models can be simulated and checked for consistency. So, our proposed solution carrying a constructive potential towards improvement of the UML metamodel, may have a real practical impact on the UML specifications.

1. Introduction

Nowadays models, rather than code, become the key artifacts of software development. Consequently, this raises the level of requirements for modeling languages on which practitioners, such as software designers and IT system architects, should rely in their everyday modeling work. A minor imperfectness of a modeling language metamodel may cause major problems in the language applications, thus with the model driven systems development the solidness of modeling languages foundations becomes particularly important. These foundations include for example:

- the overall internal consistency of semantics of a modeling language,
- the coherency and unambiguity in semantics definitions presenting relations between a model constructed using the modeling language from one side and the corresponding to the model subject of modeling from the other side,
- the theoretical justifications of the semantics relevance (e.g. the necessity and sufficiency of the semantic constructs for a representation of the modeling scope targeted by the language).

At its current state the UML metamodel suggests a significant area for improvement with regard to the mentioned criteria. In addition, the UML metamodel is considered to be quite sophisticated by the modelers' community. Thus for UML in order to allow for more successful practical applications as well as for their facilitation, the current state of its metamodel should be improved.

In this work analyzing the problems of existing UML metamodel we present an alternative metamodel that was inspired by the RM-ODP (Reference Model of Open Distributed Processing [1]) ISO/ITU standard. We show how our proposed metamodel successfully resolves some of the existing problems of UML and present literature references supporting our solution. The example of our metamodel that we present in this paper, implements a formalization of RM-ODP conceptual framework. UML specifications mention RM-ODP as a framework that has already influenced UML metamodel architectures ([7] in *Preface: Relationships to Other Models*). It increases the probability for the constructive potential of our metamodel to influence future evolution of UML specifications.

This paper has the following organization. Section 2 will present an analysis of the situation with the UML metamodel highlighting three of its existing problems. Section 3 will introduce a detailed analysis for each of the three problems and then will define their three respective solutions. These solutions will frame the definition of our alternative metamodel. The analysis of possibilities to adapt UML concepts to our

defined metamodel will be done in Section 4. Section 5 will conclude the paper highlighting its most important results.

2. Problems Introduction

When developing any modeling language, the language designer needs to define a scope of the language applications and then to define a set of modeling concepts that would be necessary to represent the defined scope. For the language to be useful for a modelers' community practices, the modeling concepts need to have a clear, logically structured and consistent semantics. In other words, the better structured the semantics is, and the less internal inconsistencies it has – the more useful the language for the modelers that are interested to represent the identified modeling scope.

Unified Modeling Language (UML) was designed by Object Management Group (OMG) as "*a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems*" ([7] section 1.1). This identifies the scope of UML applications.

The experience of modeling practices in modern industries shows that UML was found useful by modelers nowadays. The amount of modeling projects that use UML, amount of books written about UML and the number of software tools that support UML are big in relation with the analogous practical achievements of other modeling languages. This proves that UML in its current state is more practical then other modeling solutions, however it doesn't mean that there are no problems with the current state of UML.

Consistently with the scenario explained in the first paragraph of this section, the UML specification [7] introduces a set of modeling concepts to represent the identified modeling scope. Section 2 of the specification defines UML semantics for these concepts.

The first problem that we can identify is that this UML semantics is considered to be complex and difficult for understanding by many modelers. OMG itself in its article "*Introduction to OMG's Unified Modeling Language (UML™)*" [8] confirms this saying that the UML specification [7] is "*highly technical, terse, and very difficult for beginners to understand*". This situation can be improved by analyzing the current state of UML semantics, understanding the reasons that caused its complexity and by proposing a better organization of semantics for modeling concepts. Particularly we will show that our solution by introducing a logically precise and internally consistent semantics structure that is based on Russell's theory of types [9] makes a positive difference in relation with the absence of such a structure in current UML semantics. The explicit presence of such a structure helps for understanding of how the modeling concepts should be used in practice, while its absence creates numerous possibilities for confusions in practical applications of modeling concepts.

While performing the analysis of the current UML semantics we can localize **the second problem**. Namely that current UML semantics is very ambiguous in presenting relations between models constructed using the language on one side and the subject that is being modeled on the other side. This is an important problem, because even an internally consistent model will not have much of practical sense in the case when its relations with subject that it is supposed to represent are undefined. This situation in UML is improved in our solution with introduction of a coherent and unambiguous set of modeling concepts definitions expressing a kind of Tarski's declarative semantics [10] for the mentioned relations between the model and the subject of modeling.

The third problem of UML semantics is the absence of any justifications in the UML specification that would explain why the presented set of UML modeling concepts is necessary and sufficient to represent the UML modeling scope. Without these justifications UML theoretical value is significantly diminished, since in this situation the language cannot prove the reasonableness of its ambitions to represent its modeling scope. In our solution the introduction of the set of modeling concepts is supported by the solid philosophical and natural science foundations providing such kind of justifications.

3. Foundations of the UML Semantics: Problems and Solutions

As we see, all the three identified problems are related with nonoptimal semantics definition. Let us look at foundations of the UML semantics in order to localize the chapters in specifications from where the mentioned problems originate. The UML specification [7] in section 2.4 introduces the semantics Foundation package: "*The Foundation package is the language infrastructure that specifies the static*

structure of models. The Foundation package is decomposed into the following subpackages: Core, Extension Mechanisms, and Data Types." Analyzing the specification further we see for these three packages:

- Core: "*The Core package is the most fundamental of the subpackages that compose the UML Foundation package. It defines the basic abstract and concrete metamodel constructs needed for the development of object models.*" [7], section 2.5.1.
- Extension Mechanisms: "*The Extension Mechanisms package is the subpackage that specifies how specific UML model elements are customized and extended with new semantics by using stereotypes, constraints, tag definitions, and tagged values. A coherent set of such extensions, defined for specific purposes, constitutes a UML profile.*" [7], section 2.6.1.
- Data Types: "*The Data Types package is the subpackage that specifies the different data types that are used to define UML. This section has a simpler structure than the other packages, since it is assumed that the semantics of these basic concepts are well known.*" [7], section 2.7.1.

Thus we can conclude that the three identified problems are originating from the Core package of the UML. Consequently it is on this package that we will focus our further consideration.

3.1. Problem 1: Structural chaos of UML semantics

Let us now concentrate on the first of the identified problems, namely on the absence of a consistent structural organization of UML metamodel that leads to practical difficulties in understanding of semantics for particular modeling concepts as well as to the difficulties in understanding of semantically allowed application contexts for a particular modeling concept.

As we see from the diagram on Figure 2-5 from the Core package specification [7], the most general concept in the UML metamodel is called "Element". It is defined ([7], section 2.5.2.16) as following: "*An element is an atomic constituent of a model. In the metamodel, an Element is the top metaclass in the metaclass hierarchy. It has two subclasses: ModelElement and PresentationElement. Element is an abstract metaclass.*" Thus any atomic constituent of a UML model can be called as UML element.

As we can see from the diagrams 2-5,6,7,8,9 of the UML specifications, all the other modeling concepts are specializations of "Element". This defines a flat structure for the UML metamodel, where any of the concepts can be used as UML elements. End even if the elements obviously belong to different semantic categories (for example as "Operation" and "Class"), there is no explicit categorization defined to help a modeler to understand which concepts should be used in which context.

We may notice an introduction of "abstract" and "concrete" constructs categories in section 2.5.1 of the UML specification: "*Abstract constructs are not instantiable and are commonly used to reify key constructs, share structure, and organize the UML metamodel. Concrete metamodel constructs are instantiable and reflect the modeling constructs used by object modelers (cf. metamodelers). Abstract constructs defined in the Core include ModelElement, GeneralizableElement, and Classifier. Concrete constructs specified in the Core include Class, Attribute, Operation, and Association.*" However this categorization becomes quite confusing if it is compared with the actual terms definitions presented in UML specifications. For example, "Association" is defined ([7], section 2.5.2.3) relatively to "Classifier", which means that "Association" can be considered as both the abstract and the concrete construct. To summarize, the categorization of concepts into the abstract and the concrete constructs does not have a consistent implementation in the current UML specifications and cannot serve as a help to modelers who would like to understand the possible application context for a particular modeling concept.

An approximate sketch of another possible categorization can be found in section 2.5.2 of UML specifications. The section introduces the figures 2-5,6,7,8,9 as following: "*Figure 2-5 on page 2-13 shows the model elements that form the structural backbone of the metamodel. Figure 2-6 on page 2-14 shows the model elements that define relationships. Figure 2-7 on page 2-15 shows the model elements that define dependencies. Figure 2-8 on page 2-16 shows the various kinds of classifiers. Figure 2-9 on page 2-17 shows auxiliary elements for template parameters, presentation elements, and comments.*"

So a reader could guess that "Backbone", "Relationships", "Dependencies", "Classifiers" and "Auxiliary Elements" are probably different categories of the modeling concepts. Unfortunately these pseudocategories are neither defined in the relations between each other nor in some other theoretical or practical application context. In addition, if we check the described figures, we see that the same modeling concepts (e.g. "Classifier" or "Relationship") are present at the same time in several of the diagrams. Thus a potential differentiation between the pseudocategories is particularly difficult to understand.

So we can conclude that the current UML specification of the Core fails to introduce a practically useful categorization of concepts that would define different application contexts for different conceptual categories. Unfortunately this problem cannot be solved by a simple adoption of some categorization for the currently existing UML concepts. This is due to the absence of any explicitly mentioned consistent strategy of concepts introduction by UML. In fact judging from the specification, for us the strategy for introduction of particular concepts remains obscure even on an implicit level. Surprisingly some concepts seem to appear without a significant justification while other conventional object-oriented terms are omitted.

For example let us look at definitions of "ModelElement" and "PresentationElement", which are the two subclasses of UML element. We see that "PresentationElement" is defined ([7], section 2.5.2.33) as "*a textual or graphical presentation of one or more model elements*." Thus essentially a "PresentationElement" is a "ModelElement" presented in a textual or a graphical form. Here we may mention that in general a "ModelElement" from inside a model doesn't make sense to anybody or to anything if it is not presented in some form to somebody or to something who perceives the model in this form of presentation. Thus we may affirm that in general a "ModelElement", as soon as it is of interest to somebody or to something, is necessarily a "ModelElement" presented in some form. Thus, in fact, "PresentationElement" is a specialization of "ModelElement" where the forms of a possible presentation are known concretely (namely a textual and a graphical form). This specialization is the only value that is added to the semantics of "ModelElement" to obtain the semantics of "PresentationElement". Because of this minor significance of the added value we may consider "PresentationElement" as not important enough (not essential) to be a separate concept inside the UML metamodel. The elimination of "PresentationElement" accompanied by addition of the descriptions of possible ways of presentation inside the definition of "ModelElement" would simplify the metamodel without diminishing its value.

3.2. Problem 2: Absence of declarative semantics in UML

After having studied the complete UML metamodel we can note that the UML specifications define explicitly only two concepts whose definitions are made by referring (relating) to the subject (system) that is being modeled. The first concept is "ModelElement", it is defined ([7], section 2.5.2.27) as "*an element that is an abstraction drawn from the system being modeled.*" The second of the two concepts is "Component", it is defined ([7], section 2.5.2.12) as following: "*A component represents a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces*". All the other concepts that constitute the metamodel are defined as parts of a UML model **only** in the relation with each other and with the two mentioned concepts. That is the definitions of all the UML metamodel concepts, with exception of the two mentioned, do not make reference to the subject that is being modeled. This way of semantics definition is not optimal.

Indeed, as we said, only two concepts used in UML models are defined by a reference to the subject being modeled. More then that, the UML metamodel doesn't define why these two concepts and why only these two (and not some other) were designated for this purpose. This means:

a. that this choice of these two concepts does not have a tenable reason defined in the UML specification;
b. that UML specification does not define a tenable relation between a subject that needs to be modeled and its model.

The b conclusion is particularly important, because it means that for the UML concepts the specification does not define any kind of formal declarative semantics that was introduced by Alfred Tarski [10]. Indeed, Tarski's declarative semantics for concepts used inside models is supposed to introduce mappings between the agreed conceptualizations of a subject that is being modeled and the concepts inside its model. The UML metamodel never presents an agreed conceptualization of the subject of modeling. Thus the specification has no choice but to define modeling concepts exclusively in their interrelations inside the model. In the general case this approach is not an optimal one for the following two main reasons:

1. The overall complexity of relations between concepts in the UML metamodel is bigger than it would have been if part of the concepts were defined in the relations with the subject of modeling. Indeed, the quantity of concepts is the same in both cases, but in the latter case some concepts would be defined in a self-sufficient way, while in the former the corresponding concepts for their definitions will need relations to other concepts from the metamodel.

2. Since there is no tenable relation defined between a subject that needs to be modeled and its model and since there is no any agreed conceptualization of the subject, there cannot be a formal proof (such as in the case of Tarski's declarative semantics) that a given modeler's interpretation (that is a model) represents the subject of modeling in a logically consistent way[1]. In other words, in this case several mutually contradicting models can represent the same subject of modeling and all of them may be affirmed as adequate; or, from the other side, one single model may be related with the same degree of adequateness to different mutually incompatible subjects of modeling.

To illustrate the second point let us take Tarski's original example [10] for declarative semantics definition: 'It snows' is true (in the model) if and only if it snows (in the subject of modeling). So if we decide to take the subject of modeling where it snows, without the declarative semantics, then both 'It snows' and 'It does not snow' can be considered true in the model if it snows in the subject of modeling. From the other side, without the declarative semantics the model where "'It snows' is true" may represent equally well both the subject of modeling where it snows and the subject of modeling where it does not snow.

In the case of UML specification as we said there are only two concepts that make the direct relation to a subject of modeling: "ModelElement" and "Component". Parts of their definitions can be considered as introducing the declarative semantics. For example according with [7], sections 2.5.2.27 and 2.5.2.1 we can write for "ModelElement": 'A ModelElement exists' is true in the model if and only if a subject of modeling ("system being modeled") is. And for "Component" according with [7], section 2.5.2.12 we can write: 'A Component exists' is true in the model if and only if there is a modular, deployable, and replaceable part of the subject of modeling ("of system"). But as we see, these definitions are too abstract: they do not give a possibility for differentiation of modeling concepts, thus the choice of these concepts to be defined using the declarative semantics is not practical.

3.3. Problem 3: Absence of theoretical justifications for UML metamodel to represent the targeted modeling scope

As we said, the third problem of UML semantics is the absence of any justifications in the UML specification that would explain why the presented set of UML modeling concepts is necessary and sufficient to represent the UML modeling scope. This problem can be considered as natural for the current state of UML, because from its outset the language was constructed by OMG as a result of integration of the best existing industrial modeling practices, but these practices were never really linked with the existing scientific theories. While the "best practices" strategy can be considered as an attempt for practical justification of UML, the theoretical justification was never defined in the language specifications and still needs to be provided.

Thus as we can see, this third problem of UML metamodel is a theoretical problem, comparing to the first two identified problems which are practical. So the third problem is important as soon as UML would pretend to be standardized as a modeling technique by some international standardization committee (such as ISO), which would normally assume solid scientific foundations rather then just results of practical experience to support the language.

3.4. Solution to Problem 1: Categorization of concepts based on Russell's theory of types

So, with the fact that UML doesn't define any explicit logically consistent strategy for the introduction of modeling concepts, in order to solve the problem of the structural chaos of UML semantics we needed to define such kind of strategy ourselves.

To define the structure of our metamodel we took the basic conceptual structure of RM-ODP [1] standard part 2: Foundations and reinforced it by means of the strong theoretical foundations of Russell's theory of types [9], as well as by means of the structural principles of Tarski's declarative semantics [10].

As it was proposed in RM-ODP part 2 clause 6 that defines "Basic Interpretation Concepts" conceptual category, we call the subject of modeling (which is the subject that has some modeling interest to a

[1] Here we mean the logical consistency in an interpretation of subject of modeling. The internal consistency of a model (the model being a result of the interpretation) is a different thing. The internal consistency of a model can be insured by the consistency of the UML metamodel, while to ensure the logical consistency of the interpretation a kind of consistent Tarski's declarative semantics is necessary.

modeler) as “Universe of Discourse”. In RM-ODP “Universe of Discourse” was constituted by entities (defined in [1] 2-6.1 as “*any concrete or abstract thing of interest*”) and propositions that can be asserted or denied as being held for entities (defined [1] 2-6.2).

This notion of the “Universe of Discourse” organization is compatible with Russell’s theory of types [9] defined by Bertrand Russell in 1908, that introduces individuals and propositions over individuals. Particularly, [9] explains:

“*We may define an individual as something destitute of complexity; it is then obviously not a proposition, since propositions are essentially complex. Hence in applying the process of generalization to individuals we run no risk of incurring reflexive fallacies.*

Elementary propositions together with such as contain only individuals as apparent variables we will call first-order propositions. We can thus form new propositions in which first-order propositions occur as apparent variables. These we will call second-order propositions; these form the third logical type. [while individuals form the 1st logical type and the first-order propositions form the 2nd logical type (note by A. Naumenko)] *Thus, for example, if Epimenides asserts "all first-order propositions affirmed by me are false," he asserts a second-order proposition; he may assert this truly, without asserting truly any first-order proposition, and thus no contradiction arises.*

The above process can be continued indefinitely. The (n + 1)th logical type will consist of propositions of order n, which will be such as contain propositions of order n - 1, but of no higher order, as apparent variables. ”

Analogously, in our case we have “entity” corresponding to Russell’s “*something destitute of complexity*”, because the only intrinsic meaning of an entity in RM-ODP definition is to be “something” that can be qualified by means of propositions. An entity has no other meaning without the propositions associated with it. Thus, by mapping Russell’s “individual” and “proposition” to our “entity” and “proposition”, respectively, we can use Russell’s suggestion in the context of our universe of discourse. This allows us to differentiate the propositions with regard to their subject of application:

- if a proposition is applied to an entity it is considered as the first-order proposition;
- if a proposition is applied to a proposition it is considered as the higher-order proposition.

Of course, in an application of these propositions there may be a situation when a higher-order proposition is applied on another higher-order proposition, which in its turn is applied on yet another higher-order proposition and so on, until the overall structure of the higher-order propositions is finally applied on the first-order proposition. Hence for simplification, we will refer to the combination of several higher-order propositions, which is applied on a first-order proposition, as a single higher-order proposition.

So we ordered the entities and propositions that constitute a universe of discourse in agreement with the Russell’s theory of types. Now we can look at models that should represent an arbitrary universe of discourse. A model is the place where modeling language constructs should be applied. Thus it is for the model part of our metamodel that we should provide a useful structure of the categorization of concepts, which would explain the different contexts of practical applications for the concepts from different categories.

We suggest organizing the modeling concepts structure in such a way that there would be a straightforward correspondence between the model and the corresponding represented universe of discourse. That is, we suggest having a structure of concepts in the model constructed in agreement with Russell’s theory of types, which would correspond directly to the universe of discourse organization we presented earlier.

According to our suggestion, within the model we will be able to identify “Model Elements” that will be analogous to the Russell’s “*individuals*” defined “*as something destitute of complexity*”. Also, under this assumption, in the model we will have some concepts that are analogous to the Russell’s “*first-order propositions*” (we will call them “Basic Modeling Concepts”), and some concepts – analogs of the “*higher-order propositions*” (we will call them “Specification Concepts”). With this approach to the construction of a model it would be necessary to qualify “Model Elements” with the aid of “Basic Modeling Concepts”, which in their turn could be qualified by means of “Specification Concepts”.

Thus we are able to define the correspondence of the conceptual categories from within the model to the entities and propositions that form the universe of discourse that should be modeled. The correspondence was defined as following:

- *Entities* from the Universe of Discourse are modeled by *Model Elements* in the Model.

- *First-order Propositions* from the Universe of Discourse are modeled by *Basic Modeling Concepts* in the Model.
- *Higher-order Propositions* from the Universe of Discourse are modeled by *Specification Concepts* in the Model.

So, model elements are defined in the model as one to one counterparts to entities from the universe of discourse. Let us consider more closely the two other conceptual categories from within the model. As we showed, in correspondence with Russell's definitions, basic modeling concepts (essentially the first-order propositions) contain model elements as "*apparent variables*"; and specification concepts (the higher-order propositions) contain the basic modeling concepts as "*apparent variables*".

In fact, these two conceptual categories were introduced by RM-ODP specifications ([1] part 2, clauses 8 and 9); up to this point in our presentation we only reinforced logical justifications for this categorization with the support of Russell's theory of types and with explicit definitions of the application contexts for concepts from the two categories. For further explanation of difference between concepts from the two conceptual categories we will use the principal structure of relations between a universe of discourse from one side and its model from the other side; this structure was defined by Alfred Tarski in 1935 for the introduction of his formal declarative semantics [10].

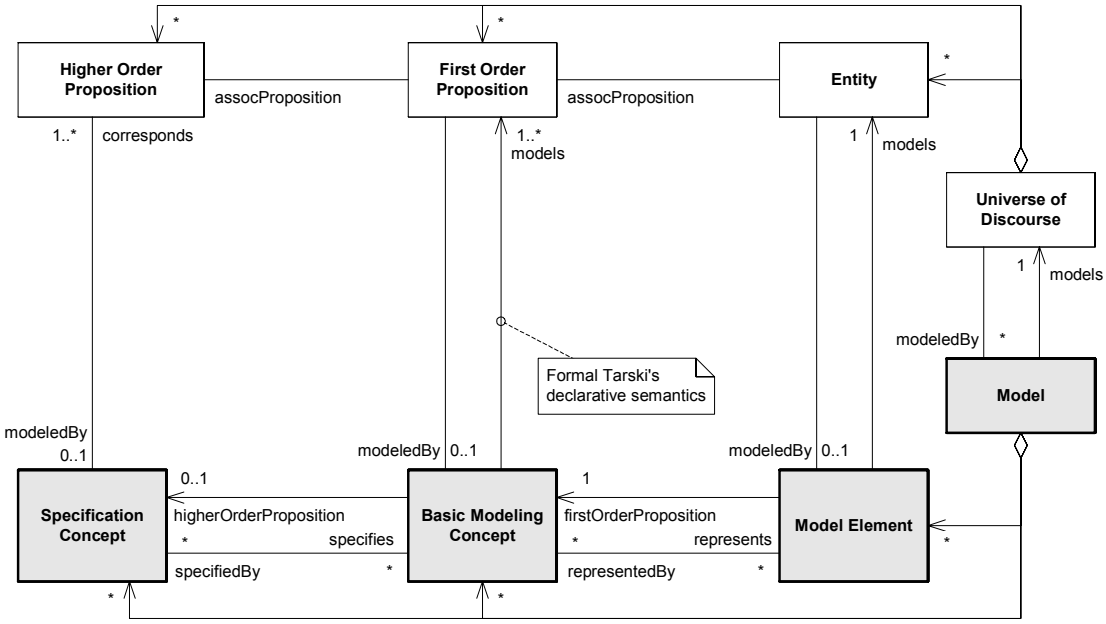

Figure 1. Categorization of concepts for the proposed metamodel (UML diagram)[2].

The basic modeling concepts set, as it aims to model the first-order propositions from the universe of discourse, should contain the concepts expressing the qualities that are considered as primary and intrinsic for the universe of discourse entities. This fundamental nature of the primary qualities belonging to the universe of discourse doesn't allow their modeling representations to be defined exclusively within the model. Hence the only possibility for a definition of the basic modeling concepts is to define them using Tarski's declarative semantics [10]: the semantics that defines equivalence of an agreed conceptualization of the universe of discourse to a concrete concept in the model. The set of basic modeling concepts

[2] On the diagram from Figure 1 in addition to all the explained particularities of the categorization structure we also showed that a specification concept can specify any of the basic modeling concepts, while a basic modeling concept can be specified by any of the specification concepts. In fact this is true only for the generic specification concepts – the subcategory of specification concepts whose definition is beyond the scope of this paper and can be found in [3], or [4].

constructed in this way is the necessary, sufficient and limited set representing a limited amount of intrinsic qualities from the universe of discourse.

The set of specification concepts contains all the other concepts that can be found in models. These concepts aim to model the higher-order propositions from the universe of discourse; thus they do not represent the primary qualities of the universe of discourse entities and hence they do not need to have Tarski's declarative semantics for their definitions. So these concepts will be defined only in the relations between themselves and in the relations with the basic modeling concepts, but not in the relations with the universe of discourse. In a general case, the set of specification concepts is not limited because of the same quality of the higher-order propositions set. As new higher-order propositions can be constructed by applying one higher-order proposition on another, new specification concepts can similarly be constructed by applying one specification concept on another.

So it becomes clear that there is a significant semantic difference between the two conceptual categories. Basic modeling concepts are defined using Tarski's declarative semantics, but specification concepts are not. This is the consequence of difference in their design purposes, which explains the clear difference in their corresponding applications within a model.

Additional details on this categorization can be found in [3]. Here let us present a UML diagram explaining the structure of the introduced categorization (see Figure 1).

3.5. Solution to Problem 2: Tarski's declarative semantics definitions for basic modeling concepts

The complete analysis of definitions for concepts from the basic modeling concepts category that was introduced in the previous section can be found in [4]. Here we will just briefly explain the overall structure of basic modeling concepts, present and present this structure in the form of UML. Later in the paper we will also explain the mappings of the introduced basic modeling concepts with the key UML concepts.

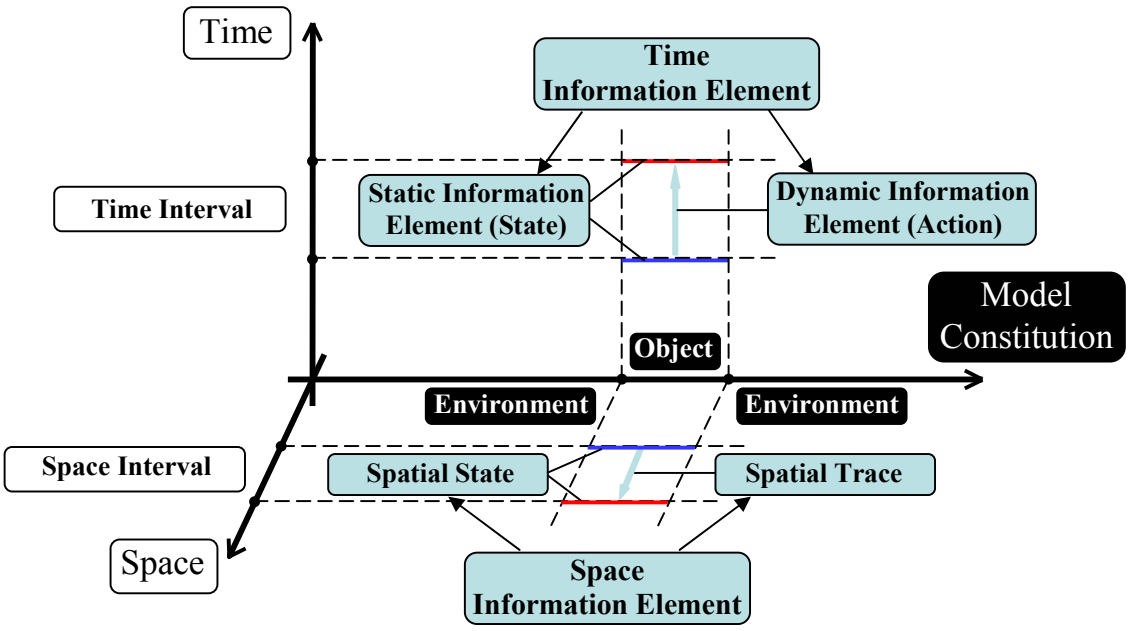

Figure 2. Three-dimensional framework with the dimensions of "Space Continuum", "Time Continuum" and "Model Constitution Continuum", which allows for the emergent "Information" continuum.

Figure 2 presents the idea of general organization for the basic modeling concepts category. Essentially the set of concepts is determined by consideration of the spatiotemporal conceptual continuum and the non-spatiotemporal conceptual continuum. The former represents in the model a space-time in the universe of

discourse, while the latter represents in the model the non-spatiotemporal conceptual entities that constitute the universe of discourse. Correspondingly to their ancestors from the universe of discourse, the former is presented by the Space and the Time dimensions on Figure 2, while the latter by the Model Constitution dimension. Being considered in the same context of a model the two introduced conceptual continuums necessarily give birth to the third one, that is essentially the Information about the Model Constitution within Space-Time.

Defining limiting points within Space-Time and Model Constitution dimensions we get concepts of Space Interval, Time Interval for the Space and the Time and concept of Object with the concept of its Environment for the Model Constitution. Also, with definition of these limiting points we are able to consider Object and its Environment

- at a single moment in Time, and thus to define the concept of Static Information Element (State) within the Information continuum;
- at an interval between two moments in Time, and thus to define the concept of Dynamic Information Element (Action) within the Information continuum.

Thus we arrive to the structure of basic modeling concepts presented on the UML diagram from Figure 3.

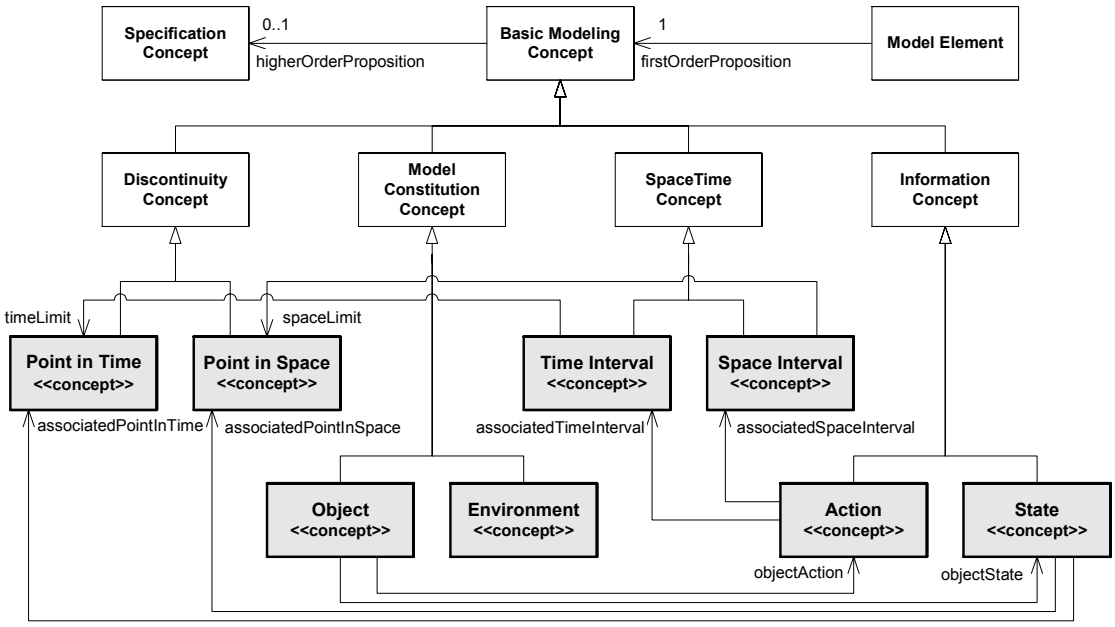

Figure 3. Basic Modeling Concepts: Conceptual Specialization (UML diagram).

In correspondence to explanations from the previous section all the basic modeling concepts have formal definitions in the form of Tarski's declarative semantics. For all the concrete definitions we recommend to check [4]. Just as an example, Action in [4] was defined as following: "***Action (Dynamic Information Element)*** *(in the model) is information about a constitutional entity perceived in the subject of modeling within some non-spatiotemporal conceptual limits in an interval in time. <…> That is: "Action" in the model is something that happens with a discrete constitutional part at a time interval in the subject of modeling*".

The definitions of basic modeling concepts as well as the definitions for all the other concepts proposed in our metamodel have a lot in common (and even identical in many cases) with the definitions of corresponding concepts given by RM-ODP. We formalized the overall RM-ODP foundations framework ([3], [5], [6]) including the basic modeling concepts part using Alloy [2] formal description technique. To sketch an idea of this formalization let us present a small piece of its Alloy code presenting a part of the described basic modeling concepts structure:

```
partition Constitution, SpaceTime, Information, : static BasicModellingConcepts
```

partition Object, Environment : static Constitution
environment (~object) : Object! -> Environment!
partition State_, Action : static Information
partition SpaceInterval, TimeInterval: static SpaceTime

Thus the basic modeling concepts semantics introducing a coherent set of Tarski's declarative semantics for relations between the concepts and the subject that is being modeled (universe of discourse) present a formally justified logical structure.

3.6. Solution to Problem 3: Philosophical and natural science foundations of our proposed metamodel

As we explained in the analysis of Problem 2, even for the choice of two modeling concepts that were linked in their definitions with the subject of modeling, UML specification does not define any tenable reason. And the set of modeling concepts that are defined using declarative semantics could be exactly the source of justifications for the ambitions to represent a given modeling scope with the modeling concepts of the language. Indeed, if the declarative semantics concepts cover all the possibilities of the agreed conceptualizations of the modeling scope then, the set of concepts can be considered as sufficient for the modeling purposes. And from the other side, exactly the set of declarative semantics concepts that would cover all the agreed conceptualizations of the modeling scope can be considered as necessary due to the necessity of the scope representation.

So, as wee see the approach to solution of the third indicated problem of UML metamodel is in the scientific justification of an agreed conceptualization of the modeling scope and in a formally defined unambiguous and logically consistent correspondence of the conceptualization to the modeling concepts that are designated to represent the conceptualization in the model.

The complete scientific justifications of the universe of discourse conceptualization (that was introduced in the previous section to support the introduction of basic modeling concepts) can be found in [4]. Here we will just mention that:

- the possibility to define limiting points and thus discrete concepts within a conceptual continuum is justified by mereology, that is a branch of philosophy studying whole-part relationships;
- the possibility to consider the constitution of models as an independent conceptual continuum realizing the dimension that is independent with regard to the spatiotemporal continuum is an original idea; the idea is a generalization of fundamental natural science foundations such as principles of both classical and relativistic mechanics. It is a generalization because in our case the modeling scope is not restricted only to space-time and material entities as it is done in natural science, but also includes any imaginary conceptual entities. The resulting space-time-constitution framework can be considered as an extension of the traditional Minkowski's space-time framework;
- the vision of defining information as a continuum emerging out from the space-time and the model constitution continuums being positioned in the same context of consideration is an original idea that however has an analogy found in Taoist philosophy.

The important result was to demonstrate that our conceptualization of the universe of discourse is in agreement with fundamental philosophical and natural science foundations. This demonstration (that can be found in [4]) allows us to rely on the introduced conceptualization and thus to define the set of Tarski's declarative semantics for the basic modeling concepts of our metamodel as not only having the logical consistency in the interpretation, but also being justified as a generalization of scientific experience. And as we explained in the beginning of this section, the definition of this Tarski's declarative semantics set for introduced by the conceptualization limited modeling scope of the universe of discourse provided a straightforward logical proof that the resulting limited set of basic modeling concepts is necessary and sufficient for the modeling scope representation.

4. Proposed metamodel and existing UML concepts

Our metamodeling solution described in this paper is an original proposition that doesn't have direct analogs in any of the known to us modeling frameworks. However, different modeling frameworks can benefit from the presented advantages of our metamodel. It is possible because our metamodel reserves only two things: the structure of its organization and the basic modeling concepts set that consists of six concepts (the concepts presented as gray rectangles in the diagram from Figure 3: Point in Time, Point in Space, Time Interval, Space Interval, Object, Environment, Action and State). At the same time the

metamodel allows for an arbitrary construction of the specification concepts set. So, different object-oriented frameworks could fit their terminology in our defined metamodel structure making use of its internal consistency, logical coherency of interpretation, formalized semantics and theoretically justified foundations.

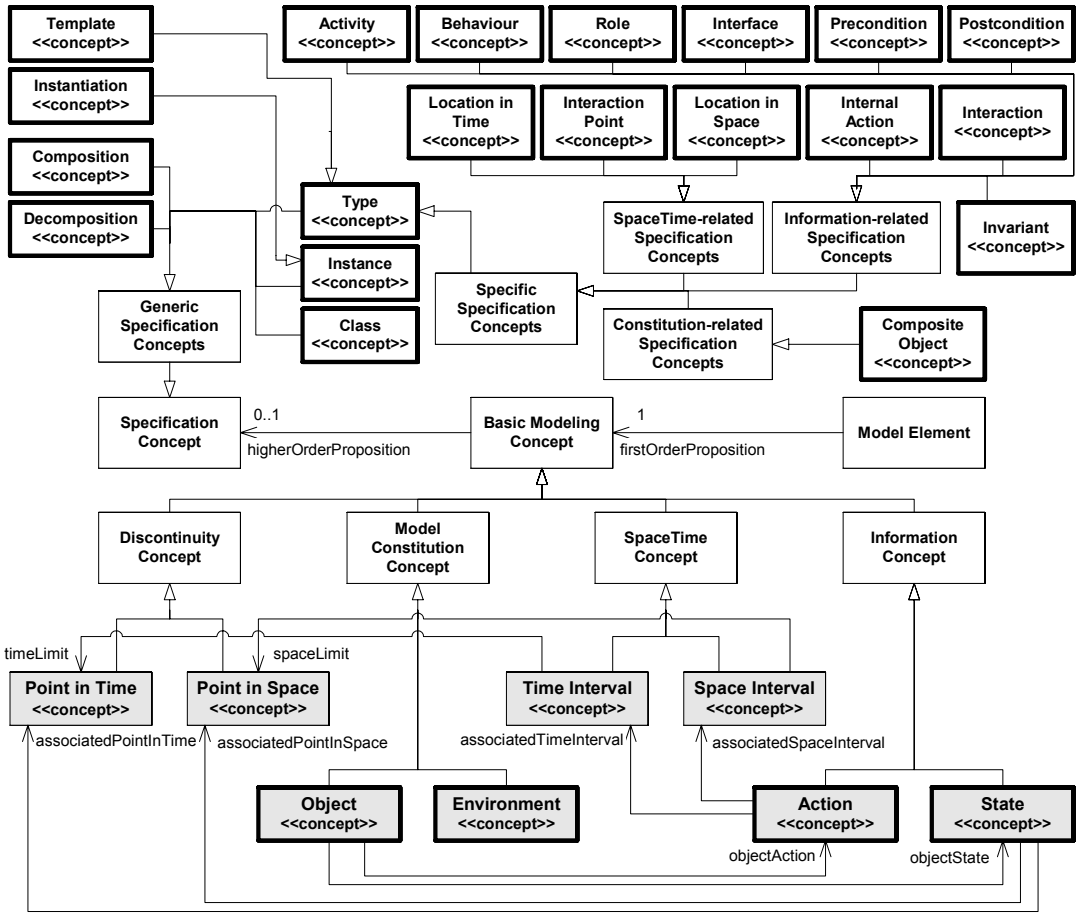

Figure 4. Example of RM-ODP conceptual framework presented using our defined metamodeling structure (UML diagram).

For example, Figure 4 shows how RM-ODP conceptual framework that is defined in [1] fits successfully our presented metamodeling structure. Original RM-ODP concepts are presented on Figure 4 with the aid of rectangles having the thick-lined borders; our reserved concepts are presented as gray rectangles. We see, that realizing the appropriate categorization that is implied by the RM-ODP definitions we can construct the specification concepts structure for our metamodel containing all the defined RM-ODP concepts. This example allowed us to formalize the RM-ODP conceptual structure ([3], [5], [6]), thus now it is possible to verify RM-ODP models with the aid of computer tools.

Since we are proposing our metamodel as a potential solution for UML, it would have been nice to present how existing UML concepts fit the structure of our proposed metamodel. Indeed, in the case of UML a priori we could have expected that it would be possible to construct the specification concepts structure with the UML terminology (analogously with how it was done in the RM-ODP case), and thus to propose the UML terminology solution analogous to the diagram from Figure 4. Unfortunately these expectations were proved to be naive by the research that we have done on UML specifications.

After having done a detailed study of UML specifications we have to affirm that with the UML terminology in its current state of definitions it is impossible to construct a reasonable modeling

framework. Unfortunately, the definitions for terminology from the Core package of UML semantics as they are currently presented contain multiple logical contradictions that can only be resolved either with the complete absence of the terms interpretation or with a free meaningless interpretation for the terms.

For example, one of the key confusions in the UML terminology definitions from the Core package is related with the word "object". Definitions of terms like "Class" ([7], section 2.5.2.9), "Flow" ([7], section 2.5.2.22), "Node" ([7], section 2.5.2.29), "Operation" ([7], section 2.5.2.30) are referring to "object", while "object" itself is not defined in the Core package and what exactly was meant by "object" in these definitions remains impossible to deduce.

Let's look for example on the definition of "Class": "*A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. <…>*". From this phrase we can understand that "object" for UML specification is something that is supposed to have some semantics. Omitting for the moment the question about this concrete semantics definition, from the fact that some semantics is defined we may conclude that "object" is a modeling construct. That is, "object" is something which exists in the model and controlled by a modeler with the aim to represent in accordance with the defined semantics something from the universe of discourse that is a subject (system) being modeled.

Further in the definition of "Class": "*In the metamodel, a Class describes a set of Objects sharing a collection of Features, including Operations, Attributes and Methods, that are common to the set of Objects. <…> A Class defines the data structure of Objects, although some Classes may be abstract; that is, no Objects can be created directly from them. Each Object instantiated from a Class contains its own set of values corresponding to the StructuralFeatures declared in the full descriptor. <…>*". The first phrase from the last quote still supports the vision that object is a modeling construct, that is a part of model. However, the last part of the quote assumes that "Object" can be "*created*" or "*instantiated from a Class*". Referring to the definition of semantics for "Instantiation" ([7], section 2.5.4.5), we see: "*The purpose of a model is to describe the possible states of a system and their behavior. The state of a system comprises objects, values, and links. Each object is described by a full class descriptor. The class corresponding to this descriptor is the direct class of the object. <…>*". Here in opposition to the previous conclusions we may clearly see that "object" is a part of the system that is represented by a model (as well as "value" and "link", which would have analogous interpretation problems by the way).

So, in which domain "object" is? The first half of "Class" definition suggests that "object" is in the model, while the second half of "Class" definition as well as "Instantiation" definition suggests that "object" is in the system which is being modeled. These are clearly two different domains, and they cannot contain the same constructs: the model is under a complete modeler's responsibility and control (which allows for the modeler's definitions of a concrete formal semantics for the modeling constructs), while the system which is being modeled is not under modeler's control (which only allows for an experiential conceptualization of the system).

Thus the references to "object" in the analyzed definitions introduce contradiction. This concrete contradiction makes impossible a logical interpretation of the concerned definitions. Unfortunately this is just one of many examples.

A concrete semantic definition for "object" cannot be found in the chapter 2 of UML specifications that is called "UML Semantics" and that as defined ([7], section 2.1.1) "*provides complete semantics for all modeling notations described in UML Notation Guide" (Chapter 3).*". However surprisingly a semantic definition for "Object" can be found in UML Notation Guide ([7], section 3.39.1). It defines: "*An object represents a particular instance of a class. It has identity and attribute values. <…>*" Which if being put in the same context with previously quoted ([7], section 2.5.2.9) "*A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. <…>*" makes the following construction: "An object represents a particular instance of a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics." Does this pretend to be a tautology? We cannot know the answer since semantics of "instance" that is heavily used as a modeling concept in UML was never defined in UML specifications.

We may also note the question on whether "object" and "Object" from ([7], section 2.5.2.9) and "run-time physical object" from ([7], section 2.5.2.29) are three identical things or not; and if not, what the differences between them are.

These are just few of many analogous examples. Other examples, such as the analogous analysis that would try to examine what UML specification see by "instance" would show even more of despair. All these things make a very unpleasant research experience that practically shows that UML specifications in

the current state present a structure of definitions which are self-contradictory and contain many baseless relations.

Thus we showed that UML terminology in its current state can be considered as undefined. In this situation the best what we can do is to wait until OMG provides logically interpretable definitions for UML terminology. Then it would be possible to position the definitions in the frame of our introduced metamodeling structure and UML would get a metamodel destitute of its current fundamental problems. Another solution for UML would be to adopt an existing example of a more successfully defined object-oriented terminology. In this case the mentioned RM-ODP conceptual framework would be a particularly interesting option, since the research results on its formalization are already available ([3], [5], [6]), and as we showed here its adaptation for the metamodeling structure is already completed.

5. Conclusions

In this paper we identified and analysed three important problems of UML metamodel. These were the following problems:

- Structural chaos of UML semantics;
- Absence of a formal declarative semantics in UML metamodel;
- Absence of theoretical justifications for UML metamodel to represent the modeling scope that is targeted by UML.

We solved these problems by defining an alternative metamodel that:

- has an internally consistent structure supported by Russell's theory of types [9];
- defines a kind of Tarski's declarative semantics [15] for the basic modeling concepts, thus it is coherent and unambiguous in the interpretations of subjects of modeling;
- is applied on a concrete example of the RM-ODP conceptual framework that is formalized in a computer-interpretable form [5];
- provides philosophical and natural science foundations to justify that its proposed modeling concepts set is necessary and sufficient to represent its identified modeling scope.

So, our metamodel defines concrete improvements for the current state of UML, and it can have a constructive influence on the evolution of UML specifications by providing the language designers with its logical rigor, its formal presentation and its solid theoretical foundations. The concreteness of our solutions and the fact that we implemented them formally on the example of RM-ODP (the framework that was mentioned by UML specifications as influential for UML metamodel architectures) are two strong points that may attract OMG's attention to the results of our research.

References

[1] ISO, ITU. *ISO/IEC 10746-1, 2, 3, 4 | ITU-T Recommendation X.901, X.902, X.903, X.904. "Open Distributed Processing - Reference Model"*. 1995-98.

[2] Jackson D.: "*Alloy: A Lightweight Object Modelling Notation*". Technical Report 797, MIT Laboratory for Computer Science, Cambridge, MA, February 2000. http://sdg.lcs.mit.edu/~dnj/pubs/alloy-journal.pdf

[3] A. Naumenko, A. Wegmann. "A Formal Foundation of the RM-ODP Conceptual Framework" *EPFL-DSC technical report No. DSC/2001/040*, http://icawww.epfl.ch/publications/naumenko/TR01_040.pdf July 2001.

[4] A. Naumenko, A. Wegmann. "Formal Foundations of General System Modeling" *EPFL-IC Technical Report No. IC/2002/012*, http://icawww.epfl.ch/publications/naumenko/TR02_012.pdf March 2002.

[5] A. Naumenko, A. Wegmann. "RM-ODP part 2: Foundations in Alloy". *EPFL-DSC Technical report No. DSC/2001/041*, http://icawww.epfl.ch/publications/naumenko/TR01_041.pdf ; Swiss Federal Institute of Technology, Lausanne, Switzerland, August 2001.

[6] A. Naumenko, A. Wegmann, G. Genilloud, W. F. Frank. "Proposal for a formal foundation of RM-ODP concepts". *Proceedings of ICEIS 2001, Workshop On Open Distributed Processing - WOODPECKER`2001*, J. Cordeiro, H. Kilov (Eds.), Setúbal, Portugal, July 2001.

[7]. OMG. *Unified Modeling Language Specification*. Version 1.4, September 2001, http://www.omg.org/uml

[8]. OMG. *Introduction to OMG's Unified Modeling Language (UML™)*. January 2002, http://www.omg.org/gettingstarted/what_is_uml.htm

[9] B. Russell, “Mathematical logic as based on the theory of types”. *American Journal of Mathematics*, 30, 1908, pp. 222-262.
[10] Tarski, A. “*Logic, Semantics, Meta-mathematics.*” Oxford University Press, 1956.