Triune Continuum Paradigm:
a paradigm for General System Modeling and its applications for UML and RM-ODP

Andrey Naumenko

Thèse № 2581 (2002)

Thèse présentée au faculté informatique et communications pour l'obtention du grade de docteur ès sciences. Acceptée sur proposition du jury:

Prof. A. Wegmann
Directeur de thèse

Dr W. Frank
Rapporteur

Prof. E. Najm
Rapporteur

Prof. S. Spaccapietra
Rapporteur

Sous la présidence de M. le **Prof. M. Odersky**, *le 22 mai 2002.*

Ecole Polytechnique Fédérale de Lausanne – 2002

ABSTRACT

This thesis presents the structural organization, theoretical foundations and basic application principles of Triune Continuum Paradigm, an original paradigm applicable to object-oriented modeling.

The paradigm defines a metamodeling structure efficient in the scope of general system modeling, in particular for object-oriented frameworks. This structure is rigorous and at the same time flexible. It allows the definition of formal ontologies for various specific object-oriented frameworks, for example for "Unified Modeling Language" (UML) or for "Reference Model of Open Distributed Processing" (RM-ODP is an ISO/ITU standard). Thus different existing frameworks, like UML or RM-ODP, can benefit from the logical rigor, internal consistency, interpretation coherency, formal presentation and solid theoretical foundations of the defined paradigm. Adoption of this paradigm allows the resolution of crucial problems existing in these different object-oriented frameworks.

The paradigm is formally presented and realized in a computer-interpretable form on the example of ontology describing the RM-ODP conceptual framework. Thus the paradigm realizes an important result that was never achieved previously: a single consistent formalization of the RM-ODP standard conceptual framework. This formalization presents a concrete example of formal ontology for general system modeling.

The paradigm is also applied on UML. This application allowed the presentation of theoretical foundations that are necessary for the understanding and definition of the UML metamodel.

This thesis is useful to readers who are interested in the fundamentals of system analysis. It can be particularly interesting to the UML semantics specialists, to the RM-ODP experts, and to ontological engineers.

VERSION ABREGEE

Cette thèse présente l'organisation structurelle, les fondements théoriques et les principes de base pour les applications du Paradigme de Triune Continuum, un paradigme original applicable à la modélisation orientée objet.

Le paradigme définit une structure de meta-modélisation applicable à la modélisation générale des systèmes et en particulier aux systèmes orientés objet. Cette structure est rigoureuse et, en même temps, flexible. Elle permet la définition d'ontologies formelles applicable dans des contextes spécifiques comme "Unified Modeling Language" (UML) ou "Reference Model of Open Distributed Processing" (RM-ODP est un standard ISO/ITU). Ainsi les différents systèmes existants, comme par exemple UML ou RM-ODP, peuvent bénéficier de la rigueur logique, de la cohérence interne, de la cohérence des interprétations, de la présentation formelle et des fondements théoriques solides du paradigme. L'adoption de ce paradigme permet de résoudre des problèmes cruciaux existant dans ces différents systèmes orientés objet.

Le paradigme est présenté formellement et illustré par un exemple d'une ontologie qui décrit formellement les concepts présents dans RM-ODP. Cette ontologie est dans une forme interprétable par des ordinateurs. Ainsi le paradigme réalise un résultat important, qui n'a jamais été atteint auparavant : une formalisation unique et cohérente de RM-ODP. Cette formalisation présente un exemple concret de ontologie formelle pour la modélisation générale des systèmes.

Le paradigme est aussi appliqué à UML. Il permet de trouver les bases théoriques nécessaire pour comprendre et définir le meta-modèle de UML.

Cette thèse est utile pour les lecteurs qui sont intéressée par les fondements de la modélisation de systèmes. Elle peut être particulièrement intéressante pour les spécialistes de la sémantique de UML, pour les experts de RM-ODP et pour les ingénieurs spécialistes en ontologies.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I am thankful to the government of Swiss Confederation that provided funding for my PhD research work in the Swiss Federal Institute of Technology - Lausanne (EPFL). I highly appreciate not only the financial support but also the high level of organization of the working and living infrastructure that I have experienced in Switzerland in general and in particular while working in EPFL. Acknowledging this contribution I also thank the administration of EPFL. I am truly thankful to the Swiss society that maintains its high living standard that I enjoyed while living in Switzerland.

I would like to express my sincere gratitude to people who either through our personal contacts or through the results of their work or through both of these means have influenced the development of my research that produced the results presented in this thesis.

First of all, I am glad to address the gratitude to my contemporaries.

I thank my PhD thesis director, head of Laboratory of Systemic Modeling (LAMS), EPFL professor Alain Wegmann, who through the time of our collaboration was always providing me with creative questions that stimulated my research interest. The active interest in my work and its results that Professor Alain Wegmann was constantly expressing and his comments supported by the serious industrial experience that he possesses provided me with significant help in shaping my research ideas to the practical requirements of the outside world and in improving their presentation. Professor Alain Wegmann successfully managed both synchronizing our research interests and thus conducting my work in the broader scope of the LAMS laboratory on the one hand, and on the other hand providing me with the necessary freedom for creation that is difficult to overestimate in the research work.

I thank the members of my PhD thesis jury: Doctor William F. Frank, the founder of the Financial Systems Architects company (Jersey City, USA), Professor Elie Najm from ENST (Paris, France), and EPFL professor Stefano Spaccapietra from Database Laboratory (LBD). I highly appreciate their interest in my research and the time that they dedicated to become acquainted with its results. Especially I appreciate the usefulness of comments that I got from them either through our informal research contacts or after the preliminary defense of my dissertation. In particular, starting from the outset of my research of the PhD topic I have had the benefit of the attention of Doctor William F. Frank who has continuously kept his

interest in the PhD topic. Through our electronic correspondence and telephone conversations he gave me the benefit of his wide experience in logic: provided important comments on the content of my research ideas, suggestions for improvements of their presentation and valuable literature references. I am grateful to Doctor William F. Frank for this feedback information that every time was very relevant, precise and contained constructive potential for the research. Professor Elie Najm and Professor Stefano Spaccapietra have been personally involved in the PhD research since my preliminary defense of the thesis. Thanks to the comments of all the jury members on the preliminary defense, I was motivated to demonstrate the value of the thesis in the relation with current industrial practices that became an important supporting argument for the ideas presented in the dissertation.

I would like to express my gratitude to Doctor of Sciences Aleksandr Nikolaevich Maliuta from Lvov (Ukraine), the founder of International Academy "New Universum", the author of Theory of Hypercomplex Systems and its supporting Invariant Modeling methodology. I had a privilege to personally participate in series of lectures given by Doctor of Sciences Aleksandr Maliuta to introduce the Theory of Hypercomplex Systems. In fact Aleksandr Maliuta was the first person who introduced me to formalized patterns of systemic thinking. His highly inspirational research results have produced a significant constructive influence on the process of my PhD research.

I would like to thank the people who contributed to the RM-ODP ISO/ITU standard development. RM-ODP, the result of their work, was the initial source of inspiration for this PhD research.

I thank my colleagues: Pavel Balabko and Gil Regev from Laboratory of Systemic Modeling and Anastasiya Sotnykova from Database Laboratory, as well as my former colleagues: Doctor Guy Genilloud and Doctor Txomin Nieva. These people contributed to the excellent friendly working atmosphere that I enjoyed in EPFL. Our numerous discussions were very helpful for this PhD research.

I thank LAMS secretary Holly B. Cogliati for her English language proofreading of my research papers. I also thank LAMS secretary Danielle Alvarez for her administrative support and LAMS computer systems manager Jean-Pierre Dupertuis for his high-quality professional support of the computer-related infrastructure.

I would like to thank the founders of the EPFL Doctoral School in Computer and Communication Sciences, in particular EPFL professors Martin Hasler, Jean-Pierre Hubaux, Jean-Yves Le Boudec, Martin Vetterli, and EPFL-I&C scientific adviser Jacques Bovay. These people actively participated in the organization of the Doctoral School, an excellent educational program in whose very first round I had a privilege to participate. The participation in this program convinced me to pursue

my PhD research in EPFL, thus predetermining the course of events that led to the realization of this PhD thesis.

After acknowledging the role of my contemporaries in this PhD research, I would like to express my gratitude to people who already passed away but whose research results were very important for this thesis.

In particular, I am grateful to Lao Tsu (604-531 BC), Zeno of Elea (490-425 BC), Hermann Minkowski (1864-1909), Bertrand Russell (1872-1970), and Alfred Tarski (1902-1983). The relation of this PhD research results to the fundamental findings of these people provided invaluable support for the results presentation.

After having expressed my gratitude to the people who have influenced the professional side of my PhD research, I would like to express my special gratitude to my family, in particular to my parents: Nina and Aleksandr Naumenko. These people through all my life, and especially through the years of my PhD research, have provided me with the invaluable moral support; their countenance and encouragements were the important factors that supported my motivation for the realization of this PhD work.

11.04.2002. Andrey Naumenko

INTRODUCTION

Reality, whether it is physical or an imaginary one (that is anything anyone could imagine), can be considered as an evolution of relations that are perceived between things of interest. The domain of general system modeling targets the development of adequate reality models. The problems of general system modeling require a definition of an efficient paradigm that would feature:

- theoretical foundations, defining:
    - a system modeling ontology that expresses relations between the model elements;
    - a meaning for the ontological constructs expressed by the relation between the reality being modeled and the model;
- logical rigor and internal consistency;
- coherency of interpretations of the reality being modeled.

This thesis defines Triune Continuum Paradigm that exhibits all the described characteristics. The paradigm allows to define a consistent object-oriented framework of modeling terms. Such framework is necessary, for example, for the modeling of distributed systems within the IT research and development projects. Often in the software systems development community the notions for basic object-oriented terminology are understood from practical experience with programming languages. The basic terms like "object", "instance", "action", "state", etc. are rarely defined in a formal way. With the aid of the aforementioned paradigm we defined the framework that fills this gap by presenting formal theoretical foundations for the object-oriented terminology. The framework is expressed in a computer-interpretable form and preserves the generic nature of object-oriented terminology that supports the framework applications for heterogeneous contexts of general system modeling.

Let us describe the history of our research and its most important results that we present in this thesis.

The initial source of our inspiration for this work was the RM-ODP (Reference Model of Open Distributed Processing) ISO/ITU standard [21]. RM-ODP introduced a conceptual framework for modeling of ODP-systems. This framework consists of traditional object-oriented terms and thus attempts to define semantics for the terms used by different object-oriented modeling languages, in particular by Unified Modeling Language (UML [40]).

We started the work by trying to solve one very concrete problem of this standard. This was the following problem:

- RM-ODP introduced a conceptual framework for modeling of ODP-systems;
  - → RM-ODP has positioned a definition and an implementation of a single consistent formalization of the conceptual framework as one of its primary goals (in fact as the goal of one of its four parts);
  - → RM-ODP neither defined nor implemented a single consistent formalization of the conceptual framework; no such formalization existed in the standard-related research; in fact this kind of formalization was never defined and implemented.

Thus the definition and implementation of a single consistent formalization of the RM-ODP conceptual framework was the concrete problem to be solved. The resolution of this problem became the goal of our work.

The problem-related analysis of the RM-ODP standard showed that within the standard:

- there were no factors that would prevent us from solving the problem;
- there was a lack of the factors that would allow for a solution based purely on the standard (without an additional information being introduced);
- there were some factors providing hints on the definition of the additional information (the information that would allow for the accomplishment of the solution).

The first of these three points encouraged us to continue the research. The second of the three points was probably the reason explaining why a solution for the problem was never achieved previously. And the third of the three points predetermined the future scenario of the evolution of our research. Let us present some details about the hints mentioned in this third point.

From the very first time when we acquainted ourselves with the conceptual framework of RM-ODP, by analyzing its definitions, we understood that the definitions contain within themselves a considerable constructive potential from which there could emerge a modeling framework that not only would allow a formalization of the standard conceptual framework, but also would bring a positive difference compared to the existing frameworks used by people from the system modeling community (in particular, for the object-oriented systems analysis and design). This potential could have been understood implicitly from the essence within the RM-ODP definitions and thus it was necessary to realize the potential in an explicit form, that is, to define this emerging modeling framework. This definition thus became the refined goal of our research.

To achieve this refined goal, we reinforced the RM-ODP standard reference model by adding strong logical and philosophical foundations as constraints for its interpretation. After completing this task we understood, that our proposed

theoretical foundations not only allow the complete formalization of the RM-ODP conceptual framework but also they are fundamental enough and general enough to systematically describe the whole domain of general system modeling problems. Hence we defined Triune Continuum Paradigm.

The definition of the paradigm has two parts:

- the definition of the paradigm's metamodeling structure,
- and the definition of the paradigm's theoretical foundations supporting the metamodeling structure.

The metamodeling structure is defined with the aid of two theoretical techniques:

- Russell's theory of types [43];
- Tarski's declarative semantics [50].

The foundations of the paradigm are defined as a philosophically supported generalization of fundamental frameworks of natural science. In particular:

- in classical (Newtonian) mechanics the observer-relational reference frames exhibit the relational nature in space, while time and material objects remain invariant for different observers;
- in relativistic mechanics the observer-relational reference frames exhibit the relational nature in space and in time, while material objects remain invariant for different observers;
- in Triune Continuum Paradigm we define the observer-relational reference frames that exhibit the relational nature in space, in time, and in constitution of models that represents different subjects of modeling (including material objects) in the models. So, representations of material objects are observer-relational here.

Thus we extend the two frameworks of mechanics by defining an additional dimension for the observer-relational reference frames. This dimension presents concepts that constitute the content of models. Concepts constituting the content of models in our paradigm are observer-relational in opposition to the observer-invariant material objects constituting the content of the natural science models. This additional to mechanics observer-relational dimension defines Model Constitution Continuum.

By positioning space-time and model constitution as independent dimensions in the same frame of reference, we automatically introduce the third conceptual category that emerges out of the first two: information about the mutual relation of model constitution and space-time. Defined in this way observer-relational SpaceTime Continuum, Model Constitution Continuum and Information Continuum are the three intrinsic features of our paradigm, the three foundations that ensure its efficiency for general system modeling. We show that these three continuums are necessary and sufficient to represent the unlimited richness of

general system modeling scope. Because of these reasons we call our paradigm as "Triune Continuum Paradigm".

So, the paradigm was defined. In particular, the definition of the paradigm provided a rigorous and at the same time flexible metamodeling structure. This structure can be adapted by different object-oriented conceptual frameworks as soon as these frameworks are internally consistent (destitute of self-contradictions). Thus different existing frameworks can benefit from the logical rigor, internal consistency, interpretation coherency, formal presentation and solid theoretical foundations of our defined paradigm.

Thus from this point we were able to return to the concrete problem that was the origin of our research.

So, as a particular example of its application, the paradigm allowed our targeted formalization of the RM-ODP conceptual framework. Thus we defined the RM-ODP-based formal ontology for general system modeling featuring:

- an internally consistent metamodeling structure that is based on Russell's theory of types [43];
- an object-oriented kind of Tarski's declarative semantics [50] for the standard terminology (e.g. for the terms like "object", "environment", "action", "state", etc);
- formal semantic relations for the modeling terms within the model.

The formalization was expressed with Alloy, the language for description of structural properties of a model [24]; thus the associated software tool (Alloy Constraint Analyzer) allows the internal consistency check of models that use the formal object-oriented terminology of our ontology.

So with the aid of Triune Continuum Paradigm we accomplished an important result that was never achieved previously: a single consistent formalization of the RM-ODP standard conceptual framework. This formalization presents a concrete example of formal ontology for general system modeling.

Thanks to Triune Continuum Paradigm, the metamodel that is realized by the formal ontology is internally consistent, introduces logical coherency of interpretation of a subject of modeling, defines formal semantics for the modeling concepts, its models are verifiable with the aid of computer tools and its conceptualization of a subject of modeling is supported by solid philosophical and natural science foundations. All these features make positive differences for the metamodel being compared with different other object-oriented metamodels that are used by modeling community nowadays. To prove the last statement we analyzed the current state of the Unified Modeling language (UML [40]) metamodel.

This analysis showed that the UML metamodel (that was designed by OMG with purposes and terminology that are similar to those of our metamodel) is destitute of all the described advantages provided by our paradigm. So we applied the paradigm

on UML demonstrating how the described advantages help to define a better metamodel for UML and help to understand the semantics of the UML concepts.

The presentation of these research results in the dissertation is organized in three parts:

- **Part I** presents the definition of Triune Continuum Paradigm;
- **Part II** presents an application of the paradigm on the example of UML;
- **Part III** presents an application of the paradigm on the example of RM-ODP.

Let us describe in details the structure of content in these three parts.

Part I consists of Chapters 1 and 2. This is the principal part of this thesis; it is dedicated to the definition of Triune Continuum Paradigm. The discussed metamodeling structure is defined in **Chapter 1**. The aforementioned theoretical foundations are presented in **Chapter 2**. At the end of Part I readers can find a summary of the results that were presented in this part of the thesis. This part is potentially useful to the readers who are interested in fundamentals of system analysis.

Part II consists of **Chapter 3** and positions Triune Continuum Paradigm that was defined in Part I in the relation with UML [40]. In this chapter several important problems of the UML metamodel are analyzed and their solutions (provided by the paradigm) are presented. A summary of these results can be found at the end of Part II. This part of the thesis can be particularly useful for the readers who are interested in semantics for the UML terminology.

Part III presents an application of Triune Continuum Paradigm on the example of RM-ODP [21] and consists of Chapters 4, 5, and 6.

As we already mentioned, Triune Continuum Paradigm allowed us to achieve an important result: a consistent formalization of the RM-ODP conceptual framework. **Chapter 4** introduces all the background information that is necessary to present the computer-interpretable form of this formalization. The computer-interpretable formalization itself is defined and explained in **Chapter 5**. These two chapters are potentially interesting for the readers who want to see a concrete implementation of Triune Continuum Paradigm, and they are particularly important to the readers who are interested in RM-ODP and its formal foundations.

Chapter 6 is dedicated to application issues of the RM-ODP-based formal ontology that was defined and explained in Chapters 4 and 5 as a concrete implementation of Triune Continuum Paradigm. Chapter 6 also introduces a UML-based Systemic Notation that supports the ontology applications. This chapter is potentially interesting to the readers who want to see the justifications of practical advantages that the formal ontology provides to modelers.

The main results of the last three chapters are summarized at the end of Part III. Apart from being particularly useful to the RM-ODP users and researchers this part of the thesis, presenting a concrete object-oriented ontology for system modeling, can also be useful for the readers who are interested in ontological theories and practical applications of system modeling ontologies.

Conclusion chapter provides the summary of the most important results that were presented in this thesis.

Appendix A is presented for the readers who are interested to see the correspondence of Triune Continuum Paradigm to the axiomatic method that is "*a method for reorganizing the accepted propositions and concepts of an existing science in order to increase certainty in the propositions and clarity in the concepts*" (see [1] on "axiomatic method").

Appendix B contains an extract from the RM-ODP standard. This extract helps the readers to check the original RM-ODP definitions when we refer to them in different parts of the thesis.

Appendix C is presented for the readers who are interested to see what kind of modifications would there be necessary to introduce in the RM-ODP standard part 2, for "RM-ODP part 2: Foundations" to fully conform to Triune Continuum Paradigm.

Appendix D contains the complete Alloy ([23], [24], [25]) code of our computer-interpretable formalization of the RM-ODP conceptual framework.

PART I

Triune Continuum Paradigm Definition

1. DEFINITION OF THE METAMODEL

Part I, consisting of chapters 1 and 2, is the principal part of this work. It is dedicated to the definition of Triune Continuum Paradigm, - an object-oriented modeling paradigm that presents a rigorous technique, efficient for modeling problems in the scope of general system modeling. In Chapter 1 the reader will:

- understand the metamodeling structure of the paradigm;
- understand the origins and definitions of the principal object-oriented terms (e.g. "object", "state", "action", "type", "instance", etc);
- see the logical rigor, internal consistency and interpretation coherency of the defined paradigm;
- see an example of how an ISO/ITU standard modeling framework fits into the defined paradigm.

Part I is potentially useful to the readers who are interested in theoretical foundations of system analysis.

1.1. Metamodel Structure

In this section we will explain the structure of the metamodel for our object-oriented modeling paradigm. A metamodel structure should explicitly define the contexts of applications for different categories of modeling concepts that are introduced by the metamodel ontology. To define the structure of our metamodel we took the basic conceptual structure of RM-ODP [21] standard part 2: Foundations and reinforced it by means of the strong theoretical foundations of Russell's theory of types [43], as well as by means of the structural principles of Tarski's declarative semantics [50].

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 modeler) as "Universe of Discourse". In RM-ODP "Universe of Discourse" was constituted by entities (defined in [21] 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 [21] 2-6.2).

This notion of the "Universe of Discourse" organization is compatible with Russell's theory of types [43] defined by Bertrand Russell in 1908, that introduces individuals and propositions over individuals. Particularly, [43] 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 ([21] 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 [50].

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 [50]: 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 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.

A computer-interpretable formalization of this categorization can be found in Chapter 5. Here let us present Figure 1.1 explaining the structure of the introduced categorization (in all the figures in Part I, excluding Figure 1.6, we use the UML notation [40]).

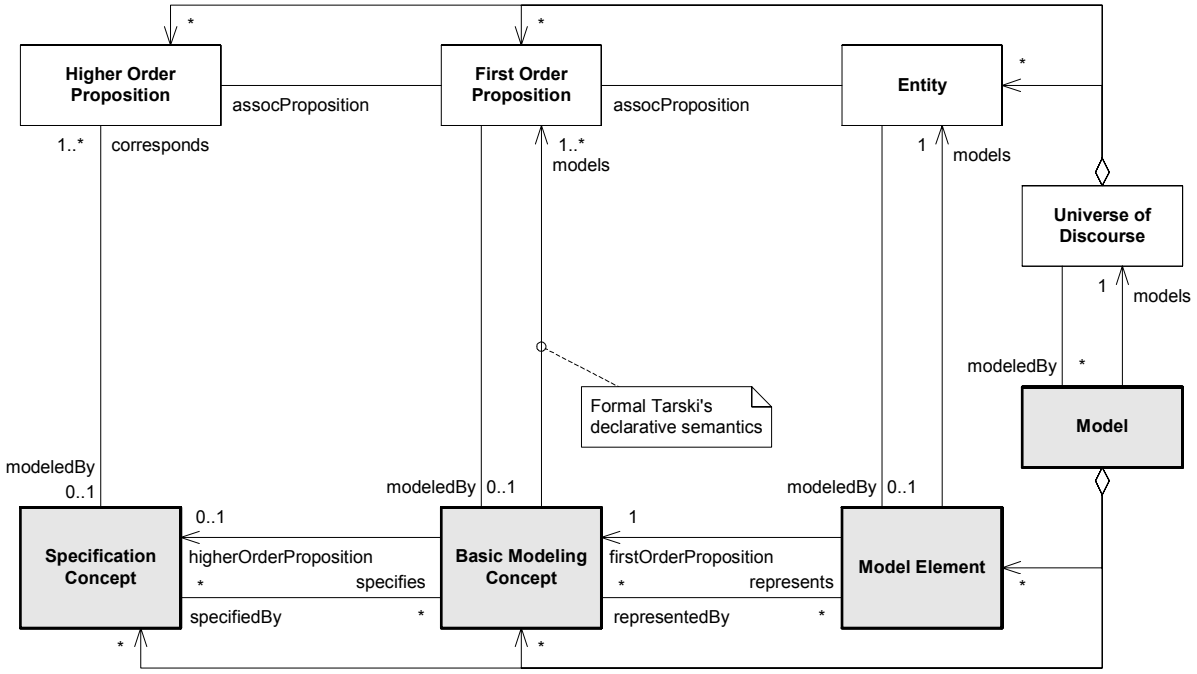

Figure 1.1: Categorization of concepts for the proposed metamodel[1].

1.2. Basic Modeling Concepts

In this section we will introduce and define the concepts for the "Basic Modeling Concepts" conceptual category of our metamodel. As we already mentioned, the name of this conceptual category is the same as the name of a category from RM-ODP ([21] 2-8). And as we will see, the results of our definitions for the basic modeling concepts set will have a lot in common with the RM-ODP concepts from their basic modeling concepts category. However, in the part of basic modeling concepts there is a difference between our metamodel and RM-ODP. Our metamodel provides a systematic reasoning for the introduction of the basic modeling concepts set, as well as for the definition of the concepts semantics. And as it will be presented in the next section of the paper, the universe of discourse conceptualization that is needed for the semantics definition is supported by fundamental philosophical and natural science foundations; whereas the RM-ODP

[1] On the diagram from Figure 1.1, 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, and 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 that will be explained in Section 1.3.2.

standard doesn't provide any reasoning to support their defined basic modeling concepts set. This is an important difference that favors our metamodel compared to RM-ODP.

As we showed in the previous section, here our goal is to determine the necessary and sufficient set of concepts that should have Tarski's declarative semantics for their definitions. Recall that, we call these concepts "Basic Modeling Concepts", whereas all the other concepts in the model (those that will not have declarative semantics but will have their semantics defined in the relations with themselves as well as in the relations between themselves and the basic modeling concepts) we will call "Specification Concepts".

During the introduction of the basic modeling concepts we will need to understand a couple of general principles[2] that support conceptual reasoning (modeling). By conceptual reasoning (modeling) here we mean a framework of reasoning that results in construction of conceptual models.

One of the foundations of conceptual reasoning supports the possibility to localize a single concept in the overall conceptual multitude of a given scope and thus to differentiate concepts between themselves. We see the nature of this foundation in duality of two essences: continuum and discontinuity. Continuum as soon as it is introduced, automatically allows for discontinuity to appear. Discontinuity allows the definition of limiting points within a continuum, which consequently allows defining the interval between limiting points and the space outside the interval within the continuum.

For instance, in plane geometry a line exhibits the continuum essence and allows for points to appear on it; a point exhibits the discontinuity essence, which allows the definitions of a segment between two points on the line and of the corresponding space on the line outside the segment. This example illustrates the definition of discrete segment within the scope of the line continuum.

In fact, the continuum-discontinuity foundations fit the conceptual reasoning in a general case; this means that the mechanisms that were illustrated in the example of the plane geometry work as well for any conceptual domain. Indeed, we can consider any concept as something that is discrete within its conceptual scope. At the same time we can consider the conceptual scope to be continuum in which discrete concepts can exist. Thus we can define a concept as the discrete interval in the corresponding conceptual dimension.

The philosophical basis of the presented continuum-discontinuity foundations of conceptual reasoning will be discussed in Chapter 2. Here we will just mention that application of these foundations can be considered as a natural modeling approach, as soon as the need to differentiate things in a given domain is considered natural.

---

[2] A theoretical justification for these principles will be provided in Chapter 2.

To use this approach we need to introduce definitions for the two basic essences, for Continuum and for Discontinuity. This will start our semantics definitions for "Basic Modeling Concepts".

def. 1: (declarative). **Continuum** (in the model) is an extent representing a subject of modeling.

def. 2: (non-declarative). **Discontinuity (also Point, Limit or Limiting Point)** in the model is a nil-extended entity that can be imagined within a **Continuum**.

For the declarative semantics definitions (the definitions that relate declaratively a term in the model with its corresponding conceptualization in the subject of modeling) we will put the word "*declarative*" when introducing these definitions. For the non-declarative semantics definitions (those definitions that do not declare any relation between the model and the subject of modeling) we will put the word "*non-declarative*" when introducing the definitions.

Thus, according with *def. 2*, "Discontinuity" may serve as a reference within a conceptual continuum in the model, which allows to define a concept as an interval within the conceptual continuum in the model, as well as to define the space outside the interval within the continuum. All the definitions of declarative semantics for the basic modeling concepts will require these very basic notions that allow us to distinguish a concept within its corresponding conceptual continuum. Thus we present here Figure 1.2 demonstrating that Tarski's declarative semantics definitions for the basic modeling concepts are composed on the most general level from continuum and discontinuity definitions. Further in this section we will present all the necessary specializations of these two most general definitions.

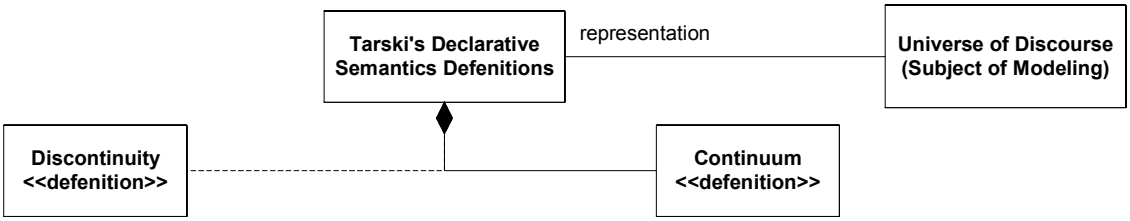

Figure 1.2: Basic Modeling Concepts: Continuum and Discontinuity definitions as the most general definitions with declarative semantics.[3]

And now, taking into account these foundations, we are ready to begin with introduction of the necessary and sufficient set of basic modeling concepts.

[3] The question whether the discontinuity definition is indeed declarative or not will be discussed in Section 2.2. For now we have put the dashed line in the figure keeping this question open.

First we start by considering a notion of the space-time continuum. This notion is in correspondence with the conventional models of space-time in the universe that were proposed by natural science and mathematics (e.g. Galilean space-time and Minkowski's space-time)[4]. As explained, we can introduce a notion of a discrete interval within the space-time continuum as a space-time within some spatiotemporal limits in the continuum. Then, in the relation with the interval we can introduce the space-time outside the interval as a part of the space-time continuum that does not contain the interval. These notions of the space-time continuum, of the spatiotemporal limits, of the space-time interval and of the space-time outside the interval give us a possibility to define the four respective basic modeling concepts:

def. 3: (declarative). **SpaceTime Continuum** (in the model) is a space-time (perceived in the subject of modeling).

def. 4: (non-declarative). **Point in SpaceTime** (in the model) is a **Point** in **SpaceTime Continuum**.

Thus a "Point in SpaceTime" defines a spatiotemporal limit.

def. 5: (declarative). **SpaceTime Interval** (in the model) is a space-time within some spatiotemporal limits (perceived in the subject of modeling).

def. 6: (declarative). **SpaceTime outside a SpaceTime Interval** (in the model) is a space-time outside some spatiotemporal limits (perceived in the subject of modeling).

Thus we can extend the diagram from Figure 1.2 as it is presented in Figure 1.3.

By making reference to the perceived conceptualization of the subject that is being modeled (to the universe of discourse) definitions *def. 3, 5, 6* introduced Tarski's declarative semantics for the three respective concepts; *def. 4* is a non-declarative semantic constraint because it is defined without a reference to a conceptualization of subject of modeling, but only referring to the concepts from inside the model. To complete the definitions we need an additional non-declarative semantic constraint that will insure an identity of the relations between the three concepts within a model and the relations that are perceived between their respective conceptualizations of the subject of modeling:

[4] Galilean space-time is used in classical (Newtonian) mechanics; there it contributes to the definition of observer-relational reference frames with the observer-relational space and the observer-invariant time. Minkowski's space-time is used in relativistic mechanics; there it contributes to the definition of observer-relational reference frames with the observer-relational space and time.

def. 7: (non-declarative). **SpaceTime outside a SpaceTime Interval** is a complement of the corresponding **SpaceTime Interval** within the **SpaceTime Continuum**. That is: the union of the "SpaceTime outside a SpaceTime Interval" and of the corresponding "SpaceTime Interval" gives the "SpaceTime Continuum", while the intersection of the "SpaceTime outside a SpaceTime Interval" and of the corresponding "SpaceTime Interval" gives nil.

This semantic constraint defines that in the model a "SpaceTime outside a SpaceTime Interval" and the corresponding "SpaceTime Interval" are two disjoint (non-overlapping) parts complementing each other within the "SpaceTime Continuum".

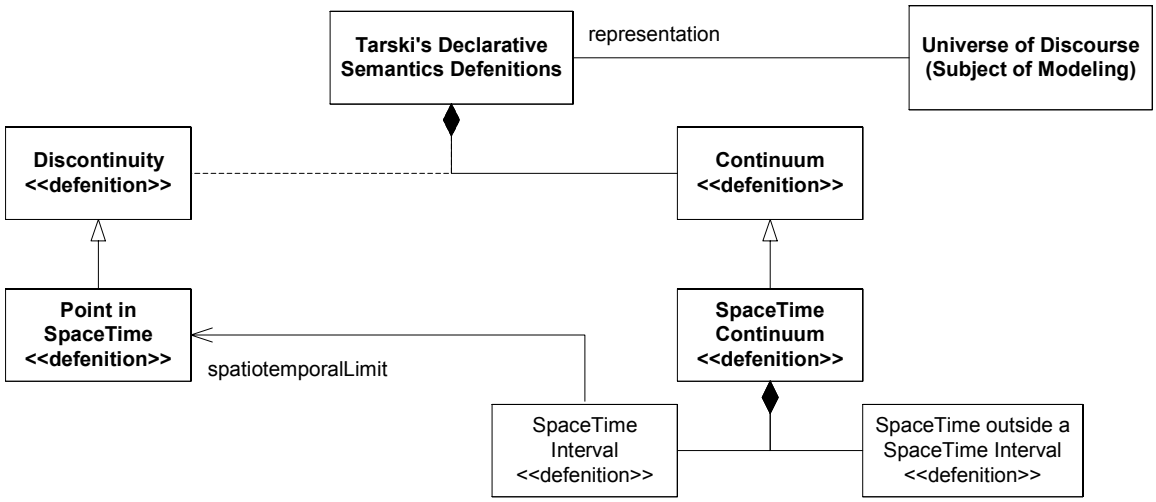

Figure 1.3: Basic Modeling Concepts: definitions of declarative semantics for the SpaceTime concepts.

We have introduced the four concepts whose purpose is to model spatiotemporal essence perceived in the subject of modeling. In natural science the spatiotemporal essence is traditionally considered as a primordial feature of the universe, - that's why we decided that for the general system modeling purposes it is also essential to distinguish the spatiotemporal part in the conceptualization of the universe of discourse (of the subject of modeling).

Naturally, as soon as we distinguished the spatiotemporal essence within the universe of subjects of modeling, its complement in this universe will be unambiguously the non-spatiotemporal essence. To represent the non-spatiotemporal essence in the model, we should have some concepts that, being considered without space-time, do not exhibit spatiotemporal characteristics. Thus essentially, the purpose of these non-spatiotemporal concepts will be to constitute the invariant to the space-time essence of the model. Because of this purpose we

decided to refer to this non-spatiotemporal conceptual dimension of the model as "Model Constitution".

As we explained earlier in this section, we can consider any concept as something that is discrete within its conceptual scope. And we can consider the conceptual scope to be continuum in which discrete concepts can exist. Thus we can define a concept as the discrete interval in the corresponding conceptual dimension. Applying this mechanism to the model constitution, we will obtain a discrete constitution interval and its complement within the model constitution continuum. Traditionally in modern system modeling (particularly as defined in RM-ODP [21] 2-8.1 and 2-8.2) the principal piece constituting models is called "Object" and the constitutional part of the model that is not that "Object" is called as "Environment of the object". Hence for us there is no need to change terminology when introducing the constitution-related part of the basic modeling concepts. We can define:

def. 8: (declarative). **Model Constitution Continuum** (in the model) is the non-spatiotemporal conceptual scope (perceived in the subject of modeling) that is complementary with regard to space-time within the subject of modeling (perceived in the subject of modeling).

def. 9: (non-declarative). **Point in Constitution** (in the model) is a **Point** in **Model Constitution Continuum**.

Thus a "Point in Constitution" defines a constitutional (non-spatiotemporal) limit.

def. 10: (declarative). **Object** (in the model) is a constitutional entity (perceived in the subject of modeling) within some non-spatiotemporal conceptual limits.

def. 11: (declarative). **Environment of an Object** (in the model) is a conceptual scope outside some non-spatiotemporal conceptual limits (perceived in the subject of modeling).

Figure 1.4 adds these definitions to the diagram from Figure 1.3.

Analogously to the way it was defined in the case of spatiotemporal modeling, we can introduce a non-declarative semantic constraint that will insure the adequateness of relations between the non-spatiotemporal basic modeling concepts inside models to the relations that are perceived between their respective conceptualizations of the subject of modeling:

def. 12: (non-declarative). **Environment of an Object** is a complement of the corresponding **Object** within the **Model Constitution Continuum**. That is: the union of the "Environment of an Object" and of the corresponding "Object" gives the "Model Constitution Continuum", while the intersection of the "Environment of an Object" and of the corresponding "Object" gives nil.

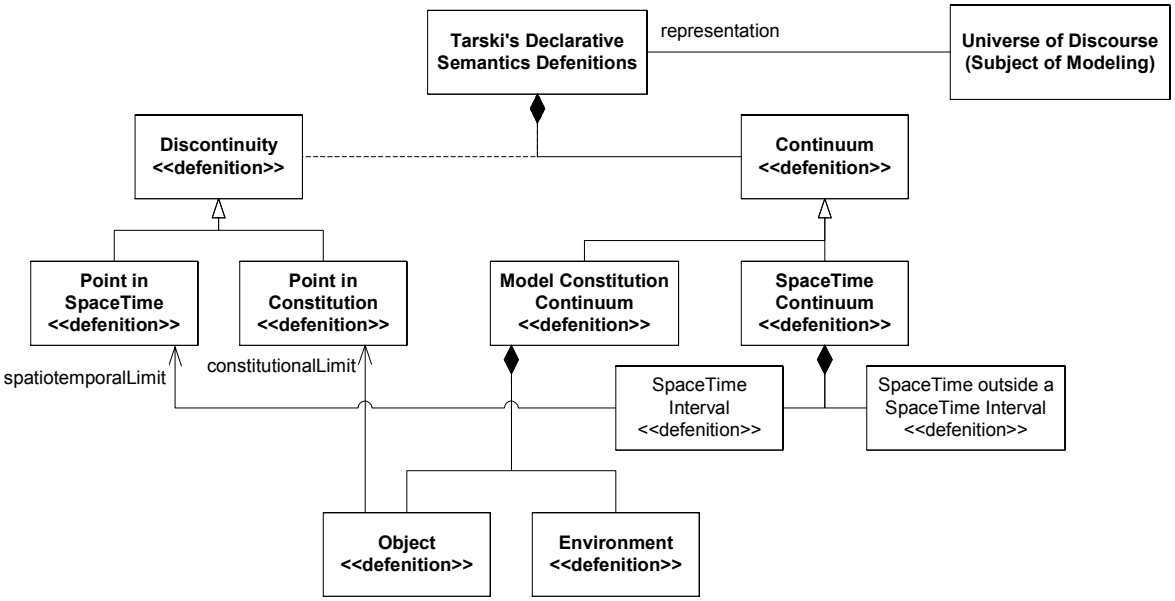

Figure 1.4: Basic Modeling Concepts: definitions of declarative semantics for the Constitution concepts are added.

We would like to note here, that if considered exclusively under these definitions, "Object" and "Environment of an Object" have no relation to "SpaceTime Continuum", which is natural because as *def. 12* explains, so far "Object" and "Environment of an Object" are only defined in the relation with each other within the "Model Constitution Continuum", and this continuum is defined as complementary to the "SpaceTime Continuum". Thus these definitions imply absence of relations with the "SpaceTime Continuum" for the semantics of "Object" and "Environment of an Object".

Note that this does not introduce a cyclic definition (e.g. the definition where two terms are defined exclusively in a relation with each other, so that such definition cannot be interpreted outside of the context of this relation), because both "Object" and "Environment of an Object" make explicit reference to the scope where they are defined – to the "Model Constitution Continuum", and this scope is formally defined in the relation with the "SpaceTime Continuum".

Now that we have introduced the two groups of basic modeling concepts (spatiotemporal and non-spatiotemporal concepts) we are ready to apply another foundation of conceptual modeling (reasoning), the one that explains the origins of emergent concepts.

Emergence is a well known property of systems that is closely considered in general system theory. Emergent concepts in the general case appear as a consequence of two conceptually different categories being put in the common scope of consideration, thus sharing a common context. For the emergent concept

to appear, it is necessary that the two conceptually different categories are defined independently from each other and cannot be deductively reduced to (or derived from) a common conceptual category without loosing their defined essential features.

In our case we just defined the appropriate two conceptually different and mutually independent parts of basic modeling concepts, spatiotemporal concepts and non-spatiotemporal concepts, and because they were defined with the purpose to be considered in the common context (namely in the model) the third conceptual part of basic modeling concepts will, necessarily, emerge out of the first two being related with regard to each other.

So, now we will consider the space-time continuum and the model constitution continuum in the same context. This will give us a possibility to consider the constitution in the space-time and the space-time in the constitution. Essentially the third emergent conceptual part will represent the information about the second category with regard to the first, as well as the information about the first category with regard to the second. That is, we will get the spatiotemporal information about the non-spatiotemporal part of model and the non-spatiotemporal information about the spatiotemporal part of model. The former will describe how the model constitution evolves through the space-time continuum and the latter will indicate how the space-time evolves through the model constitution continuum.

Usually evolution of the space-time through the model constitution continuum is not considered in the general system modeling. Thus we also will exclude it from the scope of our consideration, nevertheless mentioning that modeling of the space-time evolution in the relation with the constitution is an interesting issue that can be positioned in the scope of consideration of the general theory of relativity in the theoretical physics.

So, let us concentrate on conceptualizing the information about model constitution in space-time. For the beginning, taking into account the above mentioned arguments, let us define the term of Information Continuum:

def. 13: (declarative). **Information Continuum** (in the model) is a spatiotemporal information about the non-spatiotemporal conceptual scope (perceived in the subject of modeling).

Now we are able to apply the first of the previously described techniques of conceptual modeling (reasoning), which allows to differentiate an "Information Element" within the "Information Continuum":

def. 14: (declarative). **Information Element** (in the model) is a spatiotemporal information about a constitutional entity (perceived in the subject of modeling) within some non-spatiotemporal conceptual limits.

We see that for information-related concepts, contrary to what was done in cases of spatiotemporal and model constitution concepts, there was no need to define an explicit informational limit. This is because we construct information-related concepts as an emergent category and so for the definition of "Information Element" we can refer to the previously defined non-spatiotemporal limit.

Analyzing *def. 14* we can note that the declarative semantics for "Information Element" from the model make reference to the conceptualization of subject of modeling that corresponds to the "Object" concept (as it was defined in *def. 10*), also it refers to the spatiotemporal continuum in the subject of modeling (which participates in *def. 3*). Thus we can define the following semantic constraint:

def. 15: (non-declarative). **Information Element** (in the model) is a **SpaceTime** information about an **Object** (in the model).

At this point we could have defined the concept corresponding to *the complement of an "Information Element" within the "Information Continuum"*, - the same way as it was defined in the cases of the SpaceTime and the Model Constitution continuums. However, *for the simplification we will omit this definition.*

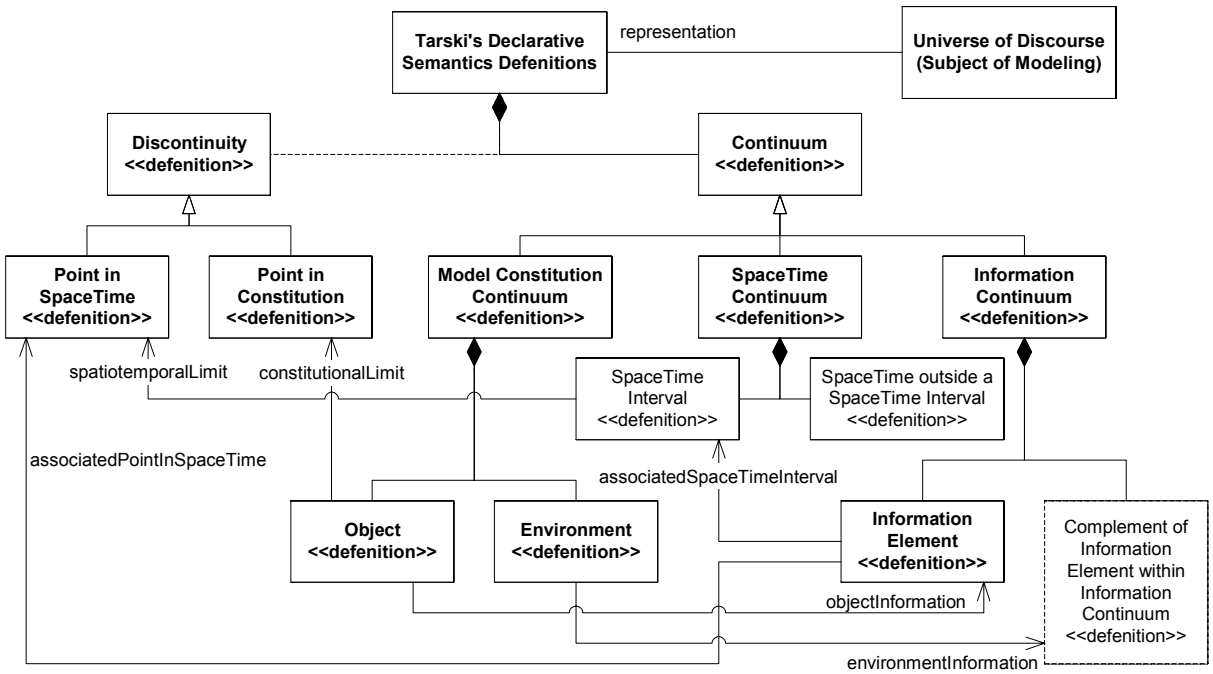

Figure 1.5: Basic Modeling Concepts: definitions of declarative semantics for SpaceTime, Constitution, and Information concepts.[5]

[5] "Complement of Information Element within Information Continuum" is drawn with the dashed rectangle because, as we mentioned, for simplification we omitted its definition.

Thus we have defined the fundamental structure for basic modeling concepts that includes "SpaceTime", "Model Constitution" and "Information" about "Model Constitution" within "SpaceTime". The concept of "Information Element" crowns the structure presenting the information about spatiotemporal evolution of objects within models. Figure 1.5 presents the resulting structure of definitions.

To define this fundamental structure of SpaceTime, Constitution and Information concepts we assumed that:[6]

- Continuum/Discontinuity foundations are applicable to system modeling. This allowed to distinguish a concept within its corresponding conceptual scope;
- spatiotemporal essence is perceived in the Universe of Discourse. This automatically allowed to distinguish this essence from its complementing (within the Universe of Discourse) non-spatiotemporal essence;
- evolution of the spatiotemporal essence through the non-spatiotemporal essence is irrelevant for the general system modeling scope. Thus we have excluded one half of the general Information scope from the models.

In fact, the elimination of the third of these assumptions would not change the structure that is presented in Figure 1.5. Simply in this case there would be necessary to extend the definitions *def. 13, 14* in a way that the defined terms would represent also the excluded part of the information.

So at this point of our discussion we have achieved an **important result**: having defined the SpaceTime, Model Constitution and Information concepts we introduced a logically consistent generic structure of basic modeling concepts that on the highest level of abstraction covers completely the representation needs for the most general modeling scope specialized only by the aforementioned three assumptions. To define SpaceTime, Model Constitution and Information concepts we did not make any other assumption, so this defined structure (presented in Figure 1.5) can be considered as generic because it can be specialized for the variety of particular models that comply with the three assumptions.

In particular, for the further exploration of the Information-related concepts in this thesis, let us consider space and time separately (in the Universe of Discourse and respectively in the model), as two continuums within the space-time continuum.

This assumption, considering space and time as two entities, is not very strict in the relation with modern modeling practices. Different possible models (e.g. Galilean space-time, Minkowski's space-time) of the traditional practices are compatible with this assumption. The assumption just differentiates space and time within the spatiotemporal continuum, but it doesn't impose any particular model for the properties of space and time interrelations. Under this assumed condition we may define the following concepts:

---

[6] The detailed justification of these assumptions will be given in Chapter 2.

def. 16: (declarative). **Space Continuum** (in the model) is a space (perceived in the subject of modeling).

def. 17: (non-declarative). **Point in Space** (in the model) is a **Point** in **Space Continuum**.

Thus a "Point in Space" defines a spatial limit.

def. 18: (declarative). **Space Interval** (in the model) is a space within some spatial limits (perceived in the subject of modeling).

def. 19: (declarative). **Space outside a Space Interval** (in the model) is a space outside some spatial limits (perceived in the subject of modeling).

def. 20: (non-declarative). **Space outside a Space Interval** is a complement of the corresponding **Space Interval** within the **Space Continuum**. That is: the union of the "Space outside a Space Interval" and of the corresponding "Space Interval" gives the "Space Continuum", while the intersection of the "Space outside a Space Interval" and of the corresponding "Space Interval" gives nil.

def. 21: (declarative). **Space Information Element** (in the model) is space-dependent information about a constitutional entity perceived in the subject of modeling within some non-spatiotemporal conceptual limits.

def. 22: (non-declarative). **Space Information Element** (in the model) is **Space**-dependent information about an **Object** (in the model).

def. 23: (declarative). **Time Continuum** (in the model) is a time (perceived in the subject of modeling).

def. 24: (non-declarative). **Point in Time** (in the model) is a **Point** in **Time Continuum**.

Thus a "Point in Time" defines a temporal limit.

def. 25: (declarative). **Time Interval** (in the model) is a time within some temporal limits (perceived in the subject of modeling).

def. 26: (declarative). **Time outside a Time Interval** (in the model) is a time outside some temporal limits (perceived in the subject of modeling).

def. 27: (non-declarative). **Time outside a Time Interval** is a complement of the corresponding **Time Interval** within the **Time Continuum**. That is: the union of the "Time outside a Time Interval" and of the corresponding "Time

Interval" gives the "Time Continuum", while the intersection of the "Time outside a Time Interval" and of the corresponding "Time Interval" gives nil.

def. 28: (declarative). **Time Information Element** (in the model) is time-dependent information about a constitutional entity perceived in the subject of modeling within some non-spatiotemporal conceptual limits.

def. 29: (non-declarative). **Time Information Element** (in the model) is **Time**-dependent information about an **Object** (in the model).

These definitions are straightforward analogs of *def. 3 – def. 7* and *def. 14 - def. 15* for the case when we consider space and time as two continuums within the space-time continuum.

Now let us explore the nature of "Information Element" within the three-dimensional framework where the dimensions are "Space Continuum", "Time Continuum" and "Model Constitution Continuum". This three-dimensional framework is presented on Figure 1.6.

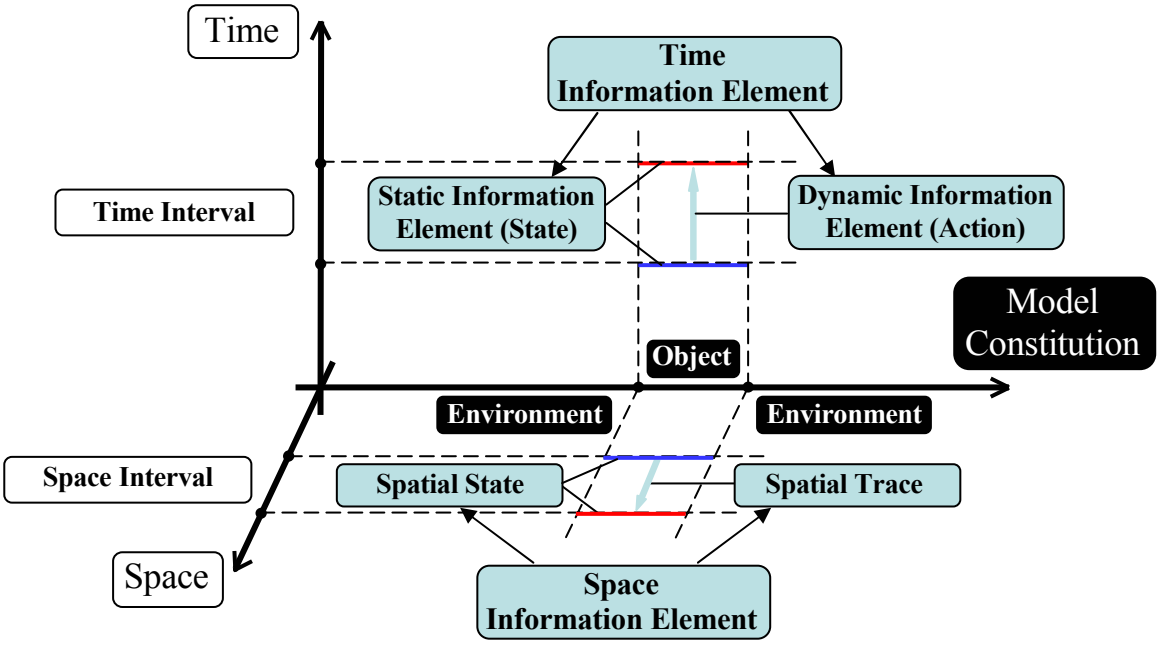

Figure 1.6: Three-dimensional framework with the dimensions of "Space Continuum", "Time Continuum" and "Model Constitution Continuum", which allows the emergent "Information Continuum".

As we saw, by considering an "Object" (the discrete part of "Model Constitution Continuum") in the relations with "Time Continuum" and with "Space Continuum" we are able to define "Time Information Element" (*def. 28, 29*) and "Space

Information Element" (*def. 21, 22*) respectively. Further, for both of the information elements we can define two conceptually different parts: the information about an object related to a single point of the time (or space) continuum, and the information about an object related to an interval within the time (or space) continuum.

Thus in the case of "Time Continuum" we will have a specialization of "Time Information Element" giving the information about an "Object" related to a single point in time, and a specialization of "Time Information Element" giving the information about an "Object" related to an interval in time. Traditionally in system modeling the former information is called *static* (or sometimes *structural*), representing the *state* of an object at a point in time, while the latter information is called *dynamic* (or sometimes *behavioral*), representing the *behavior* (or the *action*) of an object in an interval in time.

So, we can define another two basic modeling concepts as the specializations of "Time Information Element":

def. 30: (declarative). **State (Static Information Element)** (in the model) is information about a constitutional entity perceived in the subject of modeling within some non-spatiotemporal conceptual limits at a point in time.

def. 31: (non-declarative). **State (Static Information Element)** (in the model) is information about an **Object** (in the model) at a point in **Time Continuum** (in the model).

def. 32: (declarative). **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.

def. 33: (non-declarative). **Action (Dynamic Information Element)** (in the model) is information about an **Object** (in the model) in a **Time Interval** (in the model). That is: "Action" in the model is something that happens with a discrete constitutional part at a time interval in the subject of modeling.

Analogously in the case of "Space Continuum", we will have a specialization of "Space Information Element" giving the information about an "Object" related to a single point in space, and a specialization of "Space Information Element" giving the information about an "Object" related to an interval in space. Thus we can define the information about a perceived spatial state of an object depending on a point in space and the information about a perceived spatial trace that an object follows within an interval in space:

def. 34: (declarative). **Spatial State** (in the model) is information about a constitutional entity perceived in the subject of modeling within some non-spatiotemporal conceptual limits at a point in space.

def. 35: (non-declarative). **Spatial State** (in the model) is information about an **Object** (in the model) at a point in **Space Continuum** (in the model).

def. 36: (declarative). **Space Trace** (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 space.

def. 37: (non-declarative). **Space Trace** (in the model) is information about an **Object** (in the model) in a **Space Interval** (in the model). That is: "Space Trace" in the model is something that happens with a discrete constitutional part at a space interval in the subject of modeling.

This concludes the set of definitions of basic modeling concepts relevant for the general case specialized by the following four assumptions:

- Continuum/Discontinuity foundations are applicable to system modeling;
- Spatiotemporal essence is perceived in the Universe of Discourse;
- Space and Time are perceived as distinguishable from each other within the spatiotemporal essence in the Universe of Discourse;
- evolution of the spatiotemporal essence through the non-spatiotemporal essence is irrelevant for the general system modeling scope.

The first three assumptions shaped the presented definitions of the basic modeling concepts, while the fourth one didn't influence the presented definitions and just restricted the scope of information that can be represented in the models

Thus, the defined set of basic modeling concepts is necessary and sufficient for a specification of basic structure of any subject of interest in system modeling. Indeed, as it was constructed - it gives the possibility to represent any spatiotemporal or non-spatiotemporal entity, either discrete or continuum. The necessity of all the concepts in the set is justified by the need of modeling the mentioned scope. As for the sufficiency, if we assume that subjects that exhibit neither spatiotemporal nor non-spatiotemporal nature, and neither discrete nor continuum essence, are irrelevant for system modeling, then the sufficiency of the presented basic modeling concepts set is justified. We may consider this assumption as reasonable, because currently we are not aware of any presented system modeling example that wouldn't be in agreement with the assumption.

Thus we don't need other concepts for a representation of the basic essence of subjects of modeling. As we said in the beginning of this section, all the other concepts used in models will form the specification concepts category; since these concepts don't represent the fundamental nature of subjects of modeling, they will not have declarative semantics in their definitions but will be defined in the relations with themselves as well as in the relations between themselves and the basic modeling concepts.

The defined structure of basic modeling concepts is presented in Figure 1.7. As we can see, regardless of the relatively big number of introduced definitions, the structure itself is quite simple. All the introduced definitions were necessary to support a rigorous logical method of the framework presentation, which included Tarski's declarative semantics definitions, as well as definitions of non-declarative semantic constraints that interrelate basic modeling concepts within a model. However for the practical applications of the framework a modeler should keep in mind the following essential set of 8 basic modeling concepts: "**Point in Space**", "**Point in Time**", "**Space Interval**", "**Time Interval**", "**Object**", "**Environment**", "**State**" and "**Action**".

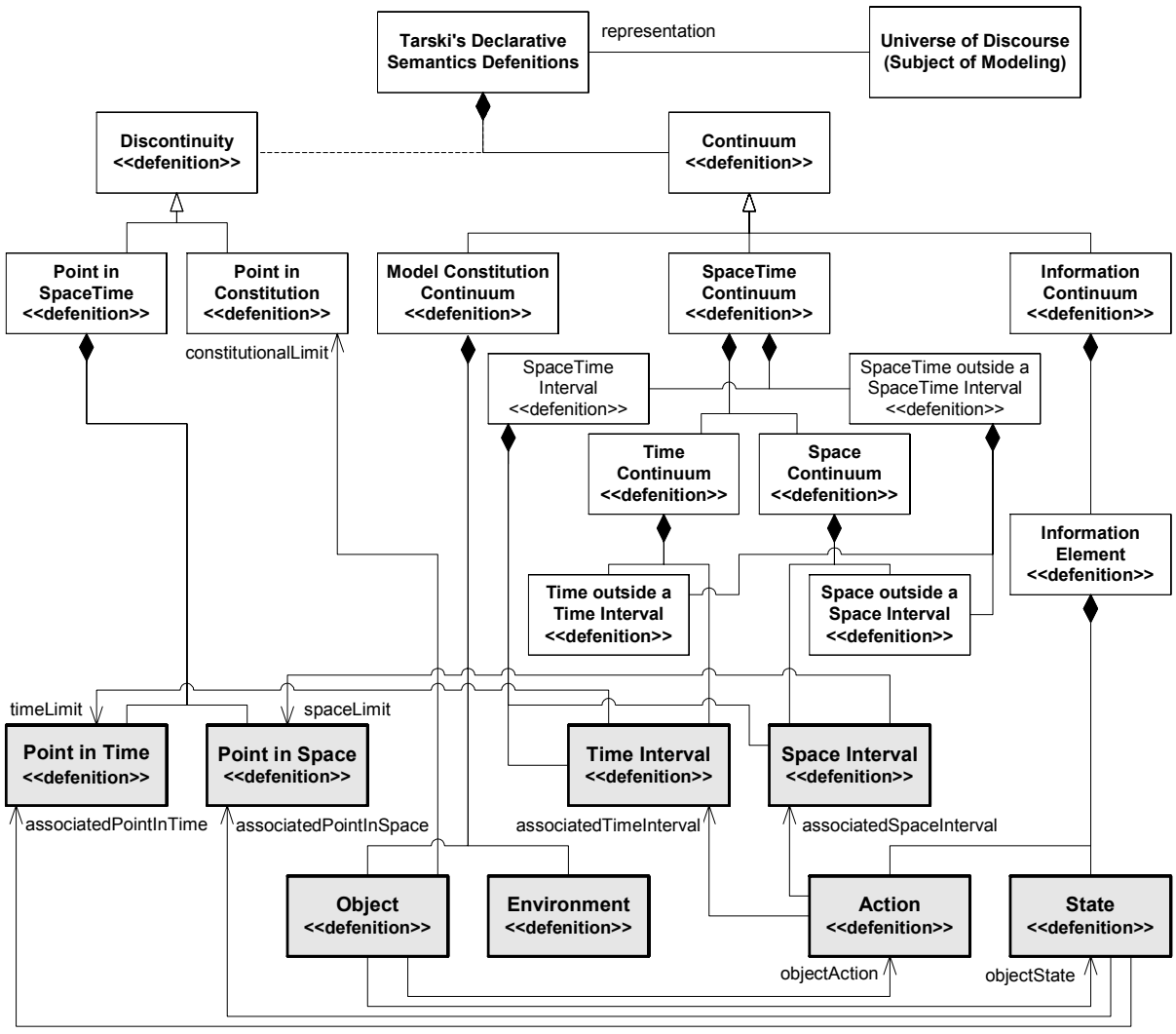

Figure 1.7: Basic Modeling Concepts: Complete Definitions Structure.

Indeed, traditionally, modeling tasks are concerned with the modeling of a concrete system, thus the consideration of overall continuums of space, of time and of information is not of primary importance. So we can consider the respective concepts of the continuums, and of parts outside intervals in the continuums, as relatively unimportant. The concepts of "Spatial State" and "Spatial Trace" are usually considered as constituent parts of "State" and "Action" respectively. This is not possible in the general case, but becomes possible with concrete space-time models where, for a given constitutional entity, a time interval is bound to a space interval and a point in time is bound to a point in space.

To conclude this section let us present the diagrams describing the introduced structures of basic modeling concepts. The diagram from Figure 1.7 presents the complete structure of definitions for basic modeling concepts under the assumption of concrete space-time models where, for a given constitutional entity, a time interval is bound to a space interval and a point in time is bound to a point in space (see the previous paragraph). Under the same assumption, the diagram from Figure 1.8 presents conceptual relations for the defined essential set of 8 basic modeling concepts.

The Figure 1.7 diagram extends the previously presented figures 1.2, 1.3, 1.4 and 1.5. It shows how the definitions were constructed in this section. Here we demonstrate how the defined semantic constraints (*def. 7, 12, 20, 27*) relate the rest of the definitions between each other as composites with their component parts. The decompositions present the complete partitionings of their composites, and because the composites were introduced to represent the complete scope of the universe of discourse (subject of modeling), - we see again that the components of compositions, which are the mentioned 8 basic modeling concepts (marked as grey-filled rectangles), are necessary and sufficient for the complete representation of an arbitrary universe of discourse in general system modeling.

The Figure 1.8 diagram shows the resulting conceptual specialization of the basic modeling concepts set. The 8 concepts from the essential set of basic modeling concepts are again marked as grey-filled rectangles. As we can see, these are the most specific basic modeling concepts, which means that in the model these are the most specific concepts among all, having their semantics defined as a kind of Tarski's declarative semantics (in the relation with the universe of discourse). Thus any further specialization of modeling constructs involving these 8 concepts can be done only with the aid of concepts from the specification concepts category.

Both of the diagrams present the introduced semantic relations for the essential set of 8 basic modeling concepts.

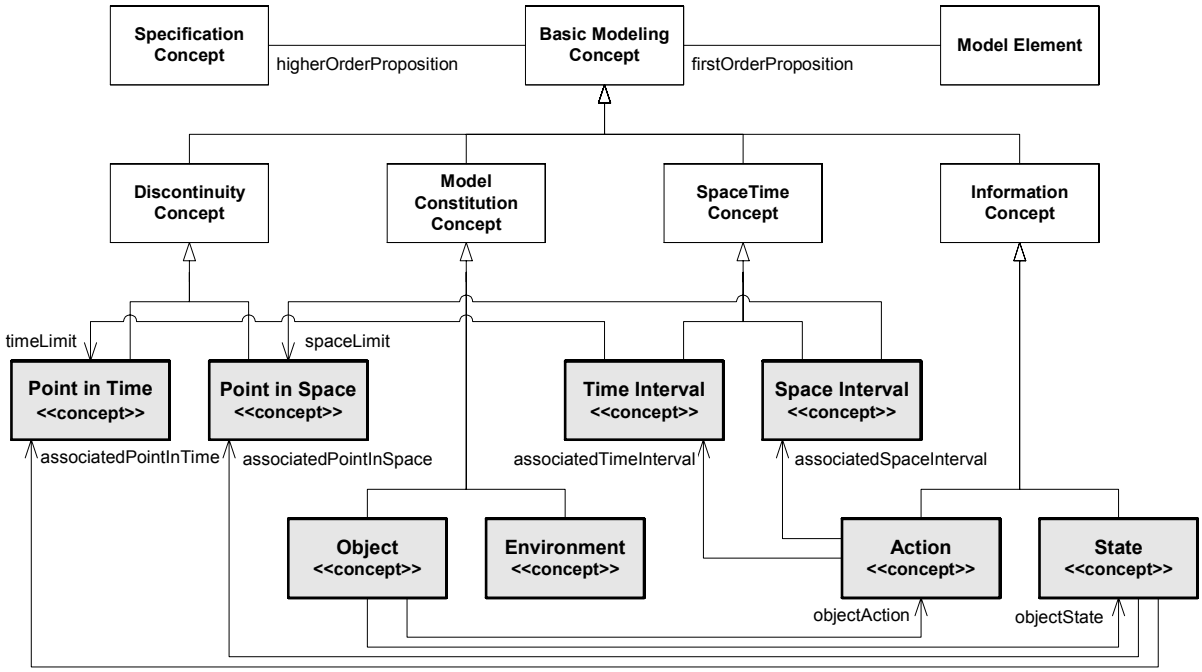

Figure 1.8: Basic Modeling Concepts: Conceptual Specialization.

1.3. Specification Concepts

In this section we will discuss the "Specification Concepts" conceptual category of our metamodel and present an example for the definition of the specification concepts that were taken from RM-ODP.

As we explained in Section 1.1 introducing our metamodel structure, the set of specification concepts contains all the concepts in the model that are neither basic modeling concepts nor model elements. This set of concepts in the model corresponds to the set of the higher-order propositions from the universe of discourse; thus in the general case the set of specification concepts is not limited (see Section 1.1).

We also explained that presenting the higher-order propositions from the universe of discourse, specification concepts do not represent 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.

The basic design purpose of the specification concepts is to represent the set of higher-order propositions from the universe of discourse. Because in the general case the set of higher-order propositions is unlimited and destitute of any predefined

structure, there are many different possible to define the specification concepts set. The only two requirements for the specification concepts definitions is that:

- they should present the structure of concepts that is internally consistent (in particular, destitute of self-contradictions);
- they should allow for a presentation of all the information that is included in the modeling scope of a concrete application of our metamodel.

This means that different existing object-oriented modeling frameworks can easily adapt our metamodel as soon as their conceptual frameworks are internally consistent and destitute of self-contradictions. Thus the different modeling frameworks of such kind have a possibility to use their concepts without redefining them as specification concepts in our metamodel.

Let us present an example of the specification concepts definition that is taken from RM-ODP standard. We were motivated to present the example of RM-ODP and not some other example particularly:

- by the internal consistency of the RM-ODP specification concepts structure and absence of self-contradictions introduced in the structure;
- by the fact that its terminology is conventional for the nowadays object-oriented modeling community (e.g. containing the terms like "type", "instance", "class", "template", etc.);
- by its possibility to represent successfully the information relevant to the modeling of IT systems by means of its defined specification concepts set accompanied by the conceptually rich extensions provided in the RM-ODP viewpoints definition.

1.3.1. Type and its Related Concepts

We will begin introducing the definitions with one of the most important specification concepts in the RM-ODP specification concepts structure, namely the concept called "Type". RM-ODP defines:

def. [21] 2-9.7 ***Type (of an <X>)****: A predicate characterizing a collection of <X>s. An <X> is of the type, or satisfies the type, if the predicate holds for that <X>.*

This definition shows that "Type" specification concept has a primary importance in the RM-ODP models. Indeed, any predicate that can be found in models characterizes a collection of things ("*<X>s*"). Sometimes it is a collection with an unlimited number of things, sometimes – a collection of a concrete number of things, particularly it can be a collection of just one thing. Thus under this definition any predicate that can be found in models is "Type".

This definition of "Type" if considered in the scope of our metamodel should be supplemented with the following restricting comment:

def. Supplement to [21] 2-9.7: Because of the special appointment of the modeling terms that are reserved for definitions on the level of the metamodel definition, none of these terms can be considered as the predicate that defines a "Type".

These reserved terms include, in particular, the terms expressing basic modeling concepts, the terms expressing specification concepts, the term of "model element" and the terms expressing the universe of discourse (e.g. "entity", "proposition"). Because these terms already have a concrete significance on the meta-modeling level, it would be unreasonable to appoint them in another context assuming a concrete significance on the modeling level.

Here we would like to present a brief example illustrating a couple of important features of "Type". We are motivated for this presentation by our practical experience which shows that "Type" with its related concepts defined by the presented RM-ODP definition is very often the source of confusions in nowadays modelers' practices.

In the relation with the "Type" concept RM-ODP defines the concepts of "Instance (of a type)" that allows, for example, for a differentiation of things characterized by the same "Type" predicate:

def. [21] 2-9.18 ***Instance (of a type):*** *An <X> that satisfies the type.*

Another RM-ODP definition introduces the terms of "Subtype" and "Supertype" for the types that have particular relations with other types:

def. [21] 2-9.9 ***Subtype/supertype:*** *A type A is a subtype of a type B, and B is a supertype of A, if every <X> which satisfies A also satisfies B.*

An instance of a type, as soon as it is named by some name within a model, necessarily expresses the predicate of this name. Thus, according to the definition [21] 2-9.7, in RM-ODP models an instance of a type necessarily expresses another type and, according to [21] 2-9.9, the latter type is a subtype of the former type, while the former type is a supertype of the latter one. In other words, a named instance of a type is the named subtype of this type. For example, instance A of type B is the subtype A of type B. The backward equivalence, that subtype A of type B is the instance A of type B, is also straightforward.

Thus in accordance with the three presented definitions, a named instance of a type and a named subtype of a type have essentially the same semantics. In addition, because an unnamed instance of a type in accordance with the three definitions makes the same sense as an unnamed subtype of a type, we can conclude that for RM-ODP models "instance of a type" and "subtype of a type" are two semantically identical concepts.

Also, in accordance with the three presented definitions, instance of a type (or, interchangeably, subtype of a type) is a specialization of the type in some context.

The amount of different modeling contexts for the specialization is unlimited in general case (for example see Figure 1.9). So, since a definition of concrete modeling contexts is possible only in a limited modeling scope, the question of existence of the most concrete instance of a type (or, interchangeably, the most concrete subtype of a type) is irrelevant in the general case of an unlimited modeling scope.

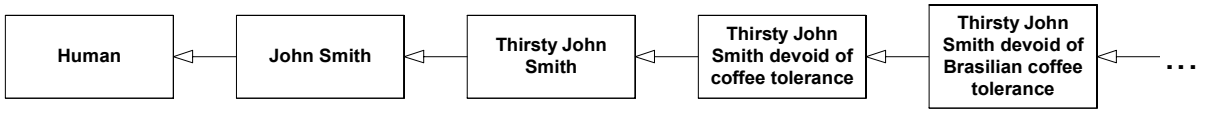

Figure 1.9: Illustration of the unlimited richness of modeling contexts leading to the irrelevance of question about existence of the most concrete instances of types in the general case.

Let us conclude the discussion about concrete specification concepts at this point. A detailed analysis of different specification concepts defined by RM-ODP, as well as their computer-interpretable formalization, will be presented in Chapter 5. In this section on example of RM-ODP specification concepts we would like to present some interesting particularities of conceptual organization of the specification concepts category that can play an important role on the level of metamodel definition.

1.3.2. Generic and Specific Specification Concepts

The group of RM-ODP specification concepts that are defined in the relations with "Type" includes concepts like "Instance", "Class", "Template", "Instantiation", etc. This group together with the concepts of "Composition" and "Decomposition" presents *generic specification concepts*.

Generic specification concepts are those specification concepts that didn't need a reference to concrete basic modeling concepts for the definitions of their semantics. Instead for their semantics definitions we can refer to an <X> as did RM-ODP, where <X> is a variable that can be substituted by absolutely any of the basic modeling concepts. As a result, these concepts can specify any of the different basic modeling concepts.

For example, in the models "Type" can be applied to different basic modeling concepts, which would result in: "Type of Object", "Type of Action", "Type of Time Interval", etc. Analogously for the concept of "Composition" we would have "Composition of Objects", "Composition of Actions", "Composition of Time Intervals", etc.

Another group of concepts for the semantics definitions needed a reference to concrete basic modeling concepts. As a result, these concepts can specify only those

concrete basic modeling concepts which were assumed in the definitions. Because of this specificity this group of concepts is called *specific specification concepts*.

For example "Internal Action", "Interaction", "Interface", "Role", etc. are specific specification concepts that should be related to the "Action" basic modeling concept; and "Precondition", "Postcondition" are specific specification concepts that are related with the "State" basic modeling concept.

As we defined them, generic specification concepts and specific specification concepts present a complete partitioning of the specification concepts category. Indeed, a specification concept can be either defined without using a reference to concrete basic modeling concepts or with a reference to concrete basic modeling concepts. Thus any specification concept is either generic or specific.

Figure 1.10 presents the diagram with the overall conceptual specialization of our metamodel, this conceptual structure includes the most important specification concepts of RM-ODP.

As we see from Figure 1.10, specific specification concepts are in fact concrete specializations of "Type" for a particular basic modeling concept. For example, "Precondition" is a special "Type of State"; "Internal Action" is a special "Type of Action" etc.

We should also note that generic specification concepts are suitable not only for an application on the basic modeling concepts, but also for building more complex specification concepts by describing any of specification concepts themselves. Indeed, instead of <X> in the generic specification concepts definitions we can put any of specification concepts (generic or specific) to construct a more complex specification concept.

When we apply a generic specification concept on a generic specification concept, the possible applications of the resulting complex specification concept are defined by the latter generic specification concept, thus this complex specification concept will also be generic. For example, "type of type" and "composition of templates" are complex specification concepts that are generic.

When we apply a generic specification concept on a specific specification concept, obviously the resulting complex specification concept will be specific. For example "class of interaction" is a complex specification concept that is specific.

Of course, these complex specification concepts may have as many levels of complexity as it will be necessary; for instance: "type of composition of templates" or "type of composition of templates for types". This corresponds to the unlimited in the general case complexity of higher-order propositions from the universe of discourse that should be modeled by means of the specification concepts.

But as we already explained in Section 1.1, if a specification concept of any level of complexity should be used in a model, then it will be applied to a basic modeling concept that, in its turn, will characterize a model element. And the statement

constructed with the specification concept and the basic modeling concept, irrespectively of its complexity, will be a single assertion about the model element.

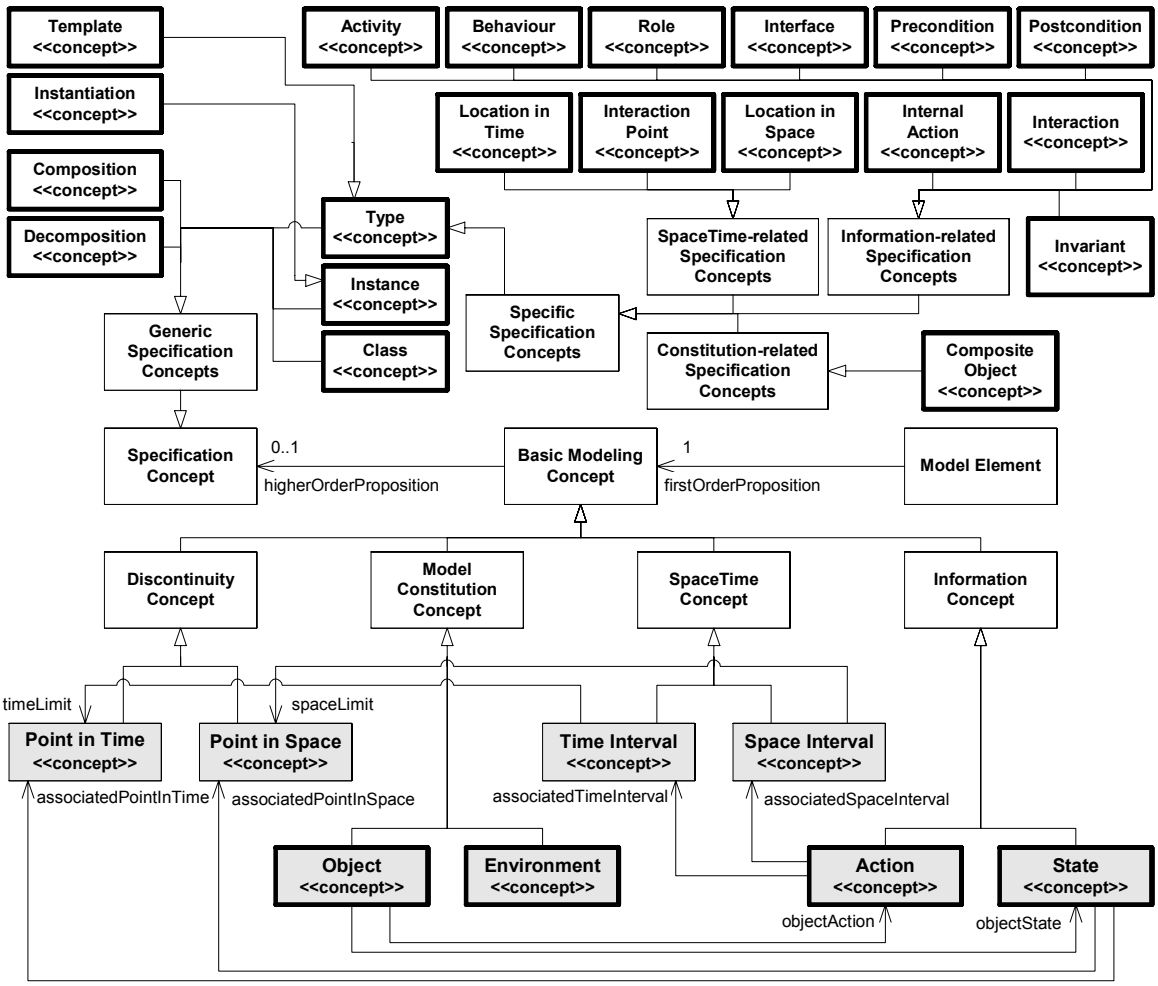

Figure 1.10: Example of RM-ODP conceptual framework presented using our defined metamodeling structure.

Thus, using RM-ODP concepts we have explained an example of the structure of specification concepts that can be used in our metamodel. The resulting conceptual specialization is presented in Figure 1.10.

We would like to note again that the structure of specification concepts used in our metamodel is not reserved by any particular example and that the structure allows many different ways for its definition. Thus our metamodel is flexible for adaptation by different object-oriented conceptual frameworks (assuming that such a framework obeys the requirements defined in the beginning of Section 1.3).

In the diagram from Figure 1.10, the original RM-ODP concepts are represented with the aid of rectangles with the thick-lined borders; our reserved basic modeling concepts are represented by gray rectangles. In Figure 1.10 we see how the internally consistent structure of our metamodel allowed us to formalize the overall presented conceptual framework of RM-ODP. Originally, the RM-ODP standard does not define an appropriate metamodel that would allow for such a formalization. And with the aid of our metamodel it is now possible to verify RM-ODP models with the aid of computer tools. This shows again the usefulness of the work that we did on the metamodel definition.

This concludes the discussion on definition of our metamodel.

2. PHILOSOPHICAL AND NATURAL SCIENCE FOUNDATIONS OF CONCEPTS SEMANTICS IN THE METAMODEL

Chapter 2 is one of the two essential chapters (as well as Chapter 1) defining Triune Continuum Paradigm. This chapter presents the theoretical foundations of the paradigm supporting the metamodel that was defined in Chapter 1. In particular, in this chapter the reader will:

- become acquainted with the fundamental origins of definitions for "modeling", "conceptual modeling" and "general system modeling" domains, and understand these definitions;
- see how the defined paradigm proves its sufficiency for the problems of general system modeling.

Chapter 2 is a particularly important chapter of this thesis; it presents the ideas that form the intrinsic essence of Triune Continuum Paradigm. The chapter is potentially useful to the readers who are interested in theoretical foundations of modeling.

A definition of formal ontology for system modeling is a general philosophical problem; its different solutions provided by humanity can be traced back to many centuries in the past. The book [41] gives a good review of those different solutions. Let us present here the philosophical foundations of our solution, which will allow to understand its particularities and to realize the differences that our ontology brings in comparison with other existing ontologies.

2.1. Modeling

In the general case we can define "Modeling" as a relation between a subject of interest to a modeler and a model of this subject. Sometimes we refer to the subject of interest to a modeler as the universe of discourse, or as the subject of modeling, or as the subject being modeled, etc.

A model is under the responsibility and control of its modeler. The subject of a modeler's interest in the general case is not under the responsibility and control of the modeler. The specific case if the subject of a modeler's interest is also a model of some other universe of discourse, the model belonging to the same modeler, - is the

only specific case when the subject of a modeler's interest is under the responsibility and control of the modeler.

Thus a model is impossible without its modeler. Obviously, the subject of a modeler's interest makes sense only in the scope of the modeler's consideration, thus it makes no sense without the modeler who would express this interest. The question whether the subject of a modeler's interest is possible or not without the modeler who would express the interest, is irrelevant for modeling because in this case:

- a model of this subject is impossible;
- the subject cannot make any sense to the modeler.

Clearly, in the general case, when there is no any concrete essence (in particular, spatiotemporal essence) implied neither at the modeling level (when establishing the modeling relation), nor on the metamodeling level (when defining the term of "Modeling"), we cannot refer to "Modeling" as anything more concrete then just a relation (e.g. as a process, or as a result of a process). Thus let us define:

def. 38: (metamodeling level). **Modeling** is a relation belonging to a modeler, the relation between a subject of the modeler's interest and a model of this subject.

2.2. To Categorize or not to Categorize? Continuum/Discontinuity Foundations

As it is defined in [47]: "The subject of *ontology* is the study of the *categories* of things that exist or may exist in some domain." As we saw it in Section 1.2, defining basic modeling concepts, in our solution prior to providing a particular study of general system modeling categories we have touched the very nature of categorization. We said that we see this nature in duality of two essences: continuum and discontinuity. Continuum as soon as it is introduced, automatically allows for discontinuity to appear. Discontinuity allows to define limiting points within a continuum, which consequently allows defining the interval between limiting points and the space outside the interval within the continuum. Thus we were able to define a concept as a discrete interval within a continuous conceptual dimension [*def. 1.*].

def. 39: (metamodeling level). **Concept** is a discrete interval within a **Continuum**.

However, here we would like to note that the continuum/discontinuity vision allowing a differentiation of things in a given domain (e.g. in time, in space, in some non-spatiotemporal conceptual domain) is by itself a modeling approach. And even if this modeling approach provides foundations not only for ontological studies, but for all the conventional science, nevertheless it was successfully challenged by Zeno, a pre-socratic Greek philosopher (490-425 BC), who formulated a set of paradoxes

[1] proving that the continuum/discontinuity model is contradictory by its internal nature and hence cannot be an adequate model for a subject of modeling.

Moreover, the definitions for discontinuous terminology in the continuum/discontinuity modeling approach (terms like "point", "boundaries", "contact", etc) still remain an undetermined [51] topic of mereology, the branch of philosophy that investigates whole-part relationships. And it is exactly for this reason that our definition of point (*def. 2* in Section 1.2) does not really introduce a Tarski's declarative semantics for the concept, instead it refers to an imaginary entity that can be positioned inside a model continuum. For the declarative semantics definition to exist, there are supposed to be some reasons originated in the subject of modeling that trigger this imagination, but while modern science is not able to conceptualize them, we are not able to define a kind of the declarative semantics in this case. This is also the reason why we have put in figures 1.2, 1.3, 1.4, 1.5 and 1.7 a dashed line for the discontinuity definition seen as a component in the overall declarative semantics definitions.

So, taking into account Zeno's paradoxes and problems with the definition of declarative semantics for the terminology of discontinuity, we see the continuum/discontinuity approach that allows differentiating things in a given domain, as a fair approximation, whose fairness is justified by practical reasons, namely by the fact that all the conventional modeling results are achieved neglecting Zeno's paradoxes. Indeed, if we choose the case of conventional reasoning where we would be able to differentiate things (which is obviously the case of this work), then we have no choice but to neglect the paradoxes. The alternative approach is formulated in [1] by reconstruction of Zeno's argument referring to the paradoxes as following: "*if you allow that reality can be successively divided into parts, you find yourself with the insupportable paradoxes; so you must think of reality as a single indivisible One*". This is a different approach, and it will not be investigated in this work. Thus in this work we are determined to stick to the continuum/discontinuity approach that gives us a possibility to define the term of concept and to have concepts grouped in the categories of concepts for a representation of a subject of modeling.

2.3. Conceptual Modeling and General System Modeling

The next step was to define the most fundamental conceptual categories for a representation of a subject of modeling. To accomplish this goal we needed first to understand the notion of conceptual modeling itself.

Clearly, conceptual modeling is a specific case of modeling, namely the specialization that is characterized by the modeling nature being conceptual. Thus we see that after we have defined the term of "modeling", the only step in defining the term of "conceptual modeling" was a realization that conceptual modeling is a

modeling done with the aid of concepts. Here the importance of the previous subsection explaining the notion of "concept" becomes clear. We see that an explanation and a definition of the "concept" term was really necessary for us to advance further, because without the understanding of what "concept" is and why it is, we wouldn't be able to understand what would an alternative to the conceptual modeling (that is the modeling done without an aid of concepts) mean. And without understanding the alternative, the phrase "*conceptual modeling is a modeling done with the aid of concepts*" would be a trivial tautology destitute of any practical sense.

def. 40: (metamodeling level). **Conceptual Modeling** is a **Modeling** done with the aid of **Concepts**.

Having understood this very basic principle we are determined to construct a concrete framework of reasoning for conceptual modeling. Particularly, our interest was to define the "general system modeling" framework within conceptual modeling.

In fact, "general system modeling" (as well as "conceptual modeling") is one of those terms that are often used in practice without a concrete definition, assuming that their definitions are implicitly understood by the community. Thus, analogously to the case of "conceptual modeling", we thought it could be useful to provide an explicit definition for the term.

When defining the term of "conceptual modeling" we referred to the key word "concept" and thus to its definition that was previously constructed with the aid of continuum/discontinuity foundations. The same approach could have been taken for the "general system modeling", by referring to the general modeling of systems and trying to define the key word "system". This approach would lead us to the findings of general system theory introduced by Ludwig von Bertalanffy [3], where "system" is defined as a "*set of elements standing in interrelations*". We see that this definition can be positioned on the same level of specificity as the definition that we had for the term of "concept". Thus this approach will not help us to define the general system modeling framework as a particular specialization of conceptual modeling.

So we decided to take a different approach that was to analyze the numerous practical experiences referred nowadays as experiences of general system modeling. We found that these experiences assume (in most of the cases implicitly) for general system modeling a relevance of the basic natural science foundations (such as spatiotemporal consideration of different subjects). Thus, we considered it relevant to define general system modeling by positioning conceptual modeling in the relation with natural science.

Classical (Newtonian) mechanics presents a framework where space is relational depending on a particular frame of reference and time is absolute (invariant). A

frame of reference is defined in the relation to a particular observer[7]. With such a model classical mechanics then studies material objects in space. In this mechanics Galilean transformations allow to pass from one frame of reference to another to study the events that happen with material objects from different observers' perspectives. Thus here material objects considered to be as invariant as time with regard to different observers' viewpoints (or different frames of reference).

Relativistic mechanics (the mechanics that is based on Einstein's principle of relativity) presents a framework where both space and time are relational depending on a particular frame of reference. Thus Minkowski's space-time is introduced, where, based on the invariance of the interval between events that happen with material objects in different frames of reference, Lorentz transformations are defined. As it was with Galilean transformations in classical mechanics, Lorentz transformations in relativistic mechanics allow to pass from one frame of reference to another to study the events that happen with material objects from different observers' perspectives. Thus in relativistic mechanics again it is assumed that material objects are considered to be invariant with regard to different observers' viewpoints (or different frames of reference).

In the case of conceptual modeling we don't pretend to study material objects as it is done in mechanics. Instead we study perceptions of different observers about a subject of modeling. A subject of modeling can be not only a material object but also a purely imaginary entity. Any entity, whether it is real or imaginary, can be modeled as soon as it is of interest to a modeler. Different perceptions about a subject of modeling result in different models. In conceptual modeling the perceptions are always subjective, that is: every modeler (observer) necessarily has an own perception about a subject of modeling, the perception which can be different from the perceptions of other modelers about the same subject. The existence of subjective viewpoints for each observer about any subject of modeling (including the laws of nature that are objective for natural science) and the extension of scope of modeling subjects with an imaginary area are two essential features that make difference between natural science and conceptual modeling.

Let us allocate a separate line for this quite important conclusion about the identified differences between conceptual modeling and natural science. So, the two differences between conceptual modeling and natural science are:

- the existence of subjective viewpoints for each observer about any subject of modeling in conceptual modeling from one side related with the invariance of laws of nature and subjects of study (e.g. material objects) for the different observers in natural science from the other side;

[7] When comparing modeling and natural science we consider the terms of «modeler» and of «observer» as interchangeable.

- the scope of the modeled subjects in conceptual modeling is different from the scope of the investigated subjects in natural science: the former can be seen as an extension of the later with the imaginary subjects of modeling.

Let us illustrate this difference with an extreme example: in conceptual modeling a modeler can build a model representing purely imaginary entities that are described by some laws having nothing in common with the laws of nature from natural science; and as soon as the model is internally consistent, it has its justified reason for existence presenting the subjective vision of the modeler with regard to the defined universe of discourse.

However, any of such subjective models is usually constructed to be used in a concrete practical context that has some kind of relation (e.g. motivating the modeling interest) with the identified universe of discourse. Thus, the modeler using a model in its assumed practical context depending on the outcome of the model use can consider this practical experience either as successful or as unsuccessful with regard to some set of criteria. Consequently, depending on the degree of success of this practical experience, the model is proved either to represent the identified universe of discourse adequately (in the case of successful practical experience), or inadequately (in the case of unsuccessful practical experience). The degree of adequateness may be compared for different models of the same universe of discourse, belonging to the same modeler (or on a bigger scale – belonging to the same modeling community) and being evaluated by the practical experiences judged with the same set of criteria. And if a model does not represent the universe of discourse adequately enough, then due to the model inefficiency in comparison with some other possible models, the model has a high chance to be discarded from practical applications of the modeler (or on a bigger scale - of the modeling community) in favor of more efficient models.

The judgment on the degree of success of practical experience of a model use including the selection of the set of criteria, in relation to which the experience should be judged, is under the responsibility and control of the modeler (or of the modeling community). Thus in modeling every modeler (every community) is naturally encouraged for the construction of adequate models for his/her (its) own benefit realized by the successful practical experiences.

Having concluded the analysis of this example of conceptual modeling let us return to the presented relations between natural science and conceptual modeling. With the performed analysis of these relations it becomes clear that all the results of natural science (and particularly of the classical mechanics and of the relativistic mechanics) are a straightforward example of a concrete modeling framework with the particular subject of modeling. Thus conceptual modeling can be seen as an extension of natural science with the two presented essential features.

This means that in a conceptual modeling framework we can use the framework definitions of natural science under the condition that they are adapted in flexibility to allow for a support of the two presented essential features of conceptual modeling. Of course, this vision of conceptual modeling having adopted the framework definitions of natural science that are adapted with regard to the two mentioned criteria is a specialization of the general vision of conceptual modeling that we defined previously as "*modeling done with the aid of concepts*".

Indeed, modeling done with the aid of concepts in the general case has no reason to be related with natural science; and it was our choice to establish this particular relation with natural science, the choice that we made because general system modeling practices imply this specialization. Thus now we can define "general system modeling" as a term expressing this particular specialization of conceptual modeling.

def. 41.1: (metamodeling level). **General System Modeling** is a kind of **Conceptual Modeling** that adopts for modeling the framework of natural science under the condition when the framework of natural science is adapted to support the following two essential features of Conceptual Modeling:

- the existence of subjective viewpoints about any subject of modeling for each observer (contrary to the invariance of laws of nature and subjects of study (e.g. material objects) for the different observers in natural science);

- the possibility to have any subject of potential interest to a modeler within the modeling scope (thus the scope of the investigated subjects of natural science should be extended with the imaginary subjects of modeling).

2.4. Adapting Natural Science Foundations: Spatiotemporal and Non-spatiotemporal continuums presentation by means of a relational reference frame

As we defined in the previous section (2.3), to construct a conceptual framework for general system modeling we needed to adapt the basic foundations of natural science to support the two identified constraints of conceptual modeling:

- the existence of subjective viewpoints about any subject of modeling for each observer (contrary to the invariance of laws of nature and subjects of study (e.g. material objects) for the different observers in natural science);

- the possibility to have any subject of potential interest to a modeler within the modeling scope (thus the scope of the investigated subjects of natural science should be extended with the imaginary subjects of modeling).

The adaptation to the second of the two mentioned constraints was relatively straightforward. For this adaptation the basic scientific frameworks of natural science remain unchanged, only the subject of investigation of natural science, that can be referred in general case as material objects with their properties, is replaced by concepts which, as we know, are not necessarily material.

The adaptation to the first of the two mentioned constraints was a bit more creative and produced an original solution; but this solution also appears in a relatively straightforward way, as soon as we perform the following analysis of the problem.

As we explained in the previous section, in natural science the frame of reference is defined in the relation to an observer, and then, according to the laws of nature (such as Einstein's principle of relativity for the relativistic mechanics) some transformations are introduced for a transition from one frame of reference to another.

We mentioned that in classical (Newtonian) mechanics the observer-relational reference frames exhibit the relational nature only in space, while time and material objects remain invariant for different observers.

Also we mentioned that in relativistic mechanics the observer-relational reference frames exhibit the relational nature in space and in time, while material objects remain invariant for different observers.

From this experience the next possible step emerges naturally, this step will give us the necessary solution. To adapt the natural science foundations to the existence of subjective viewpoints about any subject of modeling for each observer, we need to consider the reference frames that include relational space, relational time and relational representations of subjects of modeling. This can be presented as making material objects from relativistic mechanics relational instead of invariant and substituting them with concepts.

In practice this solution is realized by the definition of an additional dimension for the observer-relational reference frames (the dimension is additional in comparison with the relativistic mechanics that has only spatiotemporal dimensions). This dimension should present concepts that constitute the content of models. The way as these concepts constitute the content of models is analogous to the way as material objects constitute the content of natural science models. But the concepts are observer-relational while the material objects are observer-invariant. We call this additional to mechanics observer-relational dimension as "Model Constitution Continuum".

In agreement with these ideas we defined this continuum (*def. 8* from Section 1.2) as presenting the non-spatiotemporal conceptual essence of the universe of discourse. Indeed, as we showed, the definition of "Object" (*def.10, 12* from Section 1.2), which is the principal concept within the model constitution continuum, is done with the aid of defined non-spatiotemporal conceptual limits and in the relation with "Environment" that complements the object within the continuum. That is "Object" and "Environment of an Object" in their definitions do not have any reference to the spatiotemporal essence of the universe of discourse.

In such a way we defined for the model a reference frame representing two essences of the universe of discourse: the spatiotemporal essence and the non-spatiotemporal essence. The former is presented by "SpaceTime Continuum" in the reference frame while the latter is presented by "Model Constitution Continuum". For example, we can compare the conventional four-dimensional observer-relational model of reality (three spatial dimensions and one temporal) with our solution that if having the analogous outset will present the five-dimensional observer-relational model of reality (three spatial dimensions one temporal and one constitutional).

In accordance with its explained design goal, and in correspondence with the analogous roles of reference frames in natural science, our reference frame is defined for a given modeler. That is every modeler (and on a bigger scale – every modeling community) has his/her own reference frame where he/she constructs the models.

The possibility to introduce transformations that would allow transitions from one frame of reference to another in the case of natural science assumed the existence of universal laws of nature (such as Einstein's principle of relativity in the case of relativistic mechanics).

In our case the vision of laws of nature is subjectively controlled by every concrete modeler as well as all the information within the modeler's frame of reference. Because of this it may seem that in general system modeling, as we defined it, there cannot be a universal transformation rule for the transition from one frame of reference to another. However, the absence of universal transformation rule can also be considered as a rule, which may seem to introduce a logical paradox. But in fact it doesn't introduce a paradox, because due to the defined intrinsic design features of conceptual modeling, in conceptual modeling any rule, be it a "universal transformation rule" or "the rule of absence of universal transformation" is valid or is not valid only in the *limited* scope of models belonging to a concrete modeler (or on a bigger scale – belonging to a concrete modeling community). Thus *the question of existence or non-existence of any rules or laws in the unlimited scope of models is irrelevant in conceptual modeling*. That is, in these conditions the question doesn't have logically justified reasons for existence and has concrete logically justified reasons for nonexistence.

However, for a limited scope of models this question indeed has logically justified reasons for existence, and in this case of the limited scope the question either can have or cannot have a concrete answer, depending on the concrete conditions of models from the limited scope. If the conditions are such that the question cannot have a concrete answer, then there is indeed a local paradox. But we should note again that this paradox is relevant only within this concrete limited scope of models under the concrete conditions, and on the unlimited scope of models that is considered by conceptual modeling in general case such paradox is impossible.

Let us conclude this section by summarizing its results. In this section based on the foundations of conceptual modeling and of natural science we have defined a modeler-relational frame of reference to be used for general system modeling. The frame of reference presents in models two essences of subjects of modeling: spatiotemporal essence is presented by means of SpaceTime Continuum, non-spatiotemporal essence is presented by means of Model Constitution Continuum.

2.5. Three is a Charm: Information Continuum

By the definition of the modeler-relational frame of reference we introduced the first two basic conceptual categories that should exist in models done in general system modeling: the first is SpaceTime category which is defined by the SpaceTime Continuum dimension in the frame of reference; the second is Model Constitution category, which is defined by the Model Constitution Continuum dimension in the frame of reference.

As it we already demonstrated in the section that introduced basic modeling concepts, SpaceTime and Model Constitution necessarily give birth to the third conceptual category as soon as we positioned them in the same scope of consideration within our frame of reference. The fact of their positioning in the same reference frame by itself defines the possibility for existence of information about model constitution with regard to space-time in the models, as well as about space-time in the models with regard to model constitution. Thus our defined reference frame automatically introduces the third conceptual category emerging out of the first two. Because of its described informational nature, we call this conceptual category as "Information"; in the models it is presented by the corresponding "Information Continuum".

It is impossible to overestimate the importance of this emerging information continuum for the general system modeling. Hermann Minkowski (1864-1909) when talking about his defined observer-relational space-time frame of reference that is used in relativistic mechanics, wrote: "*Henceforth space by itself, and time by itself, are doomed to fade away into mere shadows, and only a kind of union of the two will preserve an independent reality.*" [17]. Analogously, for the models of general system modeling, with

the support of our defined observer-relational space-time-constitution frame of reference we can say that in the models space-time by itself and model constitution by itself are introduced only for the convenience of presentation, but the only thing that really needs to be presented is the emergent information about model constitution within space-time and about space-time within model constitution.

Modelers nowadays traditionally concerned with representation of information about model constitution within space-time and practically neglect a representation of information about space-time within model constitution. That is the reason why in this work we defined the terminology representing the former part of information. However, we would like to note that a priori space-time and model constitution are equally balanced within our defined observer-relational frame of reference, thus the possibilities to investigate model constitution within space-time and space-time within model constitution are equally important.

Indeed, a priory both cases are equally interesting for the modeling consideration. The first case when model constitution, that is objects and their environments, is positioned within space-time is interesting for example if a modeler would like to present how information about a particular object is projected on different time intervals. The second case when space-time, including particularly time intervals and space intervals, is positioned within model constitution (objects and their environments) is interesting for example if a modeler would like to present how information about a particular time interval is projected on different objects.

Thus, as we don't have any reason to favor space-time over model constitution or model constitution over space-time, we can consider them as two independent mutually assumed and mutually contained dimensions. That is model constitution assumed by and contained in space-time as well as space-time assumed by and contained in model constitution. This shows again that the only modeling interest is to present information about their mutual containment and not them individually.

It may seem that our reasoning for uniting space-time and constitution to produce information is the same as the reasoning that led Minkowski to unite space and time to produce a single space-time. In fact this is not true because in the latter case Einstein's principle of relativity was necessary to create the union. And our case is different: we have the third conceptual dimension (information) naturally emerging out from the first two (from constitution and space-time) positioned in the same context.

In our case information is a real emergent concept in the sense of general systems theory and system sciences. That is, it is not equivalent and cannot be equivalent to its components with a composition rule. In natural science a composition rule is impossible to formulate when there is no means to reduce components to a common dimension (or to derive components from a common dimension, which is the inverse operation). This is exactly our case, when our two components

(spatiotemporal conceptual essence and non-spatiotemporal conceptual essence) are introduced as complementary to each other. So in our case we have an emergence union of complements (space-time and constitution) producing the emergent (information).

In the case of Minkowski's space-time, the space-time is not an emergent concept, but just a simple union of space and time. That is, here the space-time is equivalent to the two components (space and time) with their composition rule. The composition rule in Minkowski's space-time is allowed by Einstein's principle of relativity, particularly by the speed of light constant, which makes it possible to reduce the spatial essence and the temporal essence to the common conceptual scope of spatiotemporal interval.

So, we see that the adaptation of the natural science foundations that was performed in the previous section led us to the realization of a new feature that from one side is essential for conceptual modeling and from the other side is not supported by natural science. This feature is emergence union of concepts leading to a new emergent concept. As we explained, in natural science there is no possibility to define a composition rule for the component concepts to be composed producing an emergent concept. Thus we can supplement our definition of "general system modeling" term with this third feature that is essential for conceptual modeling and that is not supported by natural science:

def. 41.2: (metamodeling level). **General System Modeling** is a kind of **Conceptual Modeling** that adopts for modeling the framework of natural science under the condition when the framework of natural science is adapted to support the following three essential features of Conceptual Modeling:

- the existence of subjective viewpoints about any subject of modeling for each observer (contrary to the invariance of laws of nature and subjects of study (e.g. material objects) for the different observers in natural science);

- the possibility to have any subject of potential interest to a modeler within the modeling scope (thus the scope of the investigated subjects of natural science should be extended with the imaginary subjects of modeling);

- the possibility to have emergence unions of concepts leading to new emergent concepts.

It is easy to note from our discussion that this definition by itself is an emergent definition. Indeed, before performing the adaptation of natural science foundations to the conceptual modeling needs we didn't have a reason defined in the scope of conceptual modeling consideration that would make it necessary for us to introduce the third supplement to the definition of general system modeling. But immediately

after the adaptation such a reason has emerged which led us to the final form of general system modeling definition. This demonstrates that our means of argumentation in this discussion in their level of complexity correspond to the level of complexity of the discussed problems. A correspondence of the level of complexity of the methods and techniques used for modeling to the level of complexity of the modeling problems is an important requirement for the methods and techniques which is defined in the scope of system sciences [32].

Let us now return to the Information Continuum in the models. In the section introducing basic modeling concepts we defined principal concepts belonging to this continuum. These concepts were "Action" and "State", expressing respectively dynamic (behavioral) and static (structural) information about model constitution (e.g. objects) in space-time. As we explained when introducing the structure of our metamodel, in the models these two basic modeling concepts can be specified by means of specification concepts. We also explained that the set of specification concepts is an unlimited set. The unlimited amount of specification concepts that can be potentially applied to specify these two basic modeling concepts corresponds to the unlimited richness of information.

Thus taking our defined approach a modeler starting from an intention to create a *model* would be then confronted with the two complementary essences that are intrinsically assumed by the *model*, namely with *SpaceTime* and *Model Constitution*; and the two would necessarily produce the third emergent essence of *Information*, that can be *unlimitedly rich*.

We found an interesting philosophical supporting evidence that corresponds surprisingly well to the presented in the previous paragraph realization of our approach. The evidence was found in Taoist philosophy: in "Tao Te Ching" [28] an ancient book that is believed to have been written by Lao Tsu (604-531 BC) in the paragraph 42 we find:

> *The Tao begot one.*
> *One begot two.*
> *Two begot three.*
> *And three begot the ten thousand things.*

The analogy that our approach presents with regard to this quoted piece of "Tao Te Ching" is straightforward:

- "*The Tao begot one.*" – a *universe* allows for *modeling* (or a *modeler* intends to *model*), a choice between the passive or the active perspective doesn't influence the meaning;
- "*One begot two.*" – a *model* intrinsically assumes two essences: *space-time* and *model constitution*;

- "*Two begot three.*" – from *space-time* and *model constitution* necessarily emerges *information* about their mutual relation;
- "*And three begot the ten thousand things.*" – *information* about mutually related *space-time* and *model constitution* is *unlimitedly rich*.

Because of the presented triune essence of SpaceTime, Model Constitution and Information continuums, which covers completely the defined scope of general system modeling, we decided to call our paradigm as **Triune Continuum Paradigm**.

At this point we would like to conclude the presentation of philosophical, logical (deductive) and natural science foundations of our metamodel.

PART I SUMMARY

In chapters 1 and 2 we presented an approach to the definition of an object-oriented modeling paradigm made in the scope of general system modeling. The paradigm includes a formally defined metamodel and its supporting philosophical and natural science foundations. The metamodel exhibits the following important features:

- it is internally consistent (supported by Russell's theory of types [43]);
- it is coherent and unambiguous in the interpretations of subjects of modeling (supported by the defined kind of Tarski's declarative semantics [50] for the basic modeling concepts).
- it is applied on a concrete example of the RM-ODP conceptual framework that is formalized in a computer-interpretable form (the formalization will be presented in Chapter 5).

When presenting the philosophical and natural science foundations of our metamodel, we defined the domain of general system modeling where the metamodel can be used. These theoretical foundations provided justifications for the introduced conceptualization of the universe of discourse. The conceptualization was necessary for a definition of Tarski's declarative semantics for the basic modeling concepts. By justifying the conceptualization and by defining the semantics we showed that the set of basic modeling concepts is necessary and sufficient to represent the defined scope of the general system modeling interest.

A particular originality of our paradigm is in the organization of its observer-relational frame of reference. For this frame of reference we propose to use two principal independent dimensions: the first presenting the spatiotemporal continuum (e.g. time and space intervals within models) and the second presenting the model constitution conceptual continuum (e.g. objects and their respective environments within models).

By introducing the observer-relational model constitution continuum, we extend the traditional 4-dimensional Minkowski's space-time framework into the 5-dimensional space-time-constitution framework.

By positioning space-time and model constitution as independent dimensions in the same frame of reference, we automatically introduce the third conceptual category emerging out of the first two: that is information about the mutual relation of model constitution and space-time. These defined SpaceTime Continuum, Model Constitution Continuum and Information Continuum are three intrinsic features of

our paradigm, three foundations that ensure its efficiency for general system modeling. That is the reason why we call our paradigm as "Triune Continuum Paradigm".

Thus in Part I we defined an object-oriented paradigm that is based on the fundamentals of philosophy and natural science and that introduces formal definitions for the traditional object-oriented terminology. The fundamental foundations and formal presentation of our paradigm make a positive difference in favor of the paradigm being compared with the current state of the art in the object-oriented terminology. For example, traditionally, in the software modeling and development community, the object-oriented terminology semantics are considered to originate from modeling languages and rarely presented in a formal way. Thus our metamodel can have a constructive influence on the current programming practices by providing the software systems designers and developers with its logical rigor, its formal presentation and its solid theoretical foundations.

PART II

Triune Continuum Paradigm

Application to UML

3. SOFTWARE AND BUSINESS SYSTEM MODELING: SOLUTIONS FOR THE UML METAMODEL

Chapter 3 positions Triune Continuum Paradigm that was defined in Part I in the relation with the Unified Modeling Language (UML).

In particular, in this chapter the reader will become acquainted with the analysis of the following three important problems of the UML metamodel:

- absence of an explicit structural organization defined for the UML metamodel;
- absence of a formal declarative semantics in the UML metamodel;
- absence of theoretical justifications for the UML metamodel to represent the modeling scope that is targeted by UML.

In Section 3.4 of this chapter the reader will see the demonstrations of fundamental inconsistencies of the UML terminology definitions.

Throughout the chapter the reader will see how the metamodel of our defined Triune Continuum Paradigm successfully resolves these problems of UML.

This chapter can be particularly useful for the readers who are interested in semantics for the UML terminology.

In Part I we defined Triune Continuum Paradigm for general system modeling. One of the particularly interesting applications of this paradigm is in the scope of software and business system modeling.

Unified Modeling Language (UML) proposed by the Object Management Group (OMG) is currently the leading modeling framework that is available for the software and business system modeling. Thus, it is important to analyze the current state of the UML metamodel and to demonstrate the constructive potential that our paradigm brings to the existing situation.

3.1. UML Metamodel: Motivations and Problems Identification

Modern practices of software development should successfully manage the progressing complexity of modeling problems. 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 imperfection of a modeling language metamodel may cause major problems in the language applications. Thus with model-driven systems development, the solidity

of foundations of modeling languages 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 subject of modeling that is represented by the model 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).

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

It is important to notice that here we are talking about UML metamodel, thus this discussion will be concerned with definitions of semantics of UML modeling concepts. Section 2 of [40] called "UML Semantics" is the corresponding part of UML specifications that will be in the focus of our consideration in this part of the thesis. Other sections of UML specifications [40] will not be discussed here.

In this chapter, we will analyze several important problems of the existing UML metamodel and show how our metamodel, defined in Chapter 1 in the scope of Triune Continuum Paradigm, successfully resolves these problems. As we demonstrated in Chapter 1 (in Section 1.3), a concrete example of our metamodel application implements a formalization of RM-ODP conceptual framework. UML specifications mention RM-ODP as a framework that has already influenced UML metamodel architectures ([40] in *Preface: Relationships to Other Models*). This increases the probability for the constructive potential of our metamodel to influence future evolution of UML specifications.

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 in modelers' community practices, the modeling concepts need to have clear, logically structured and consistent semantics. In other words, the better structured the semantics are, and the less internal inconsistencies they have – the more useful the language for the modelers that are interested in representing the identified modeling scope.

Unified Modeling Language (UML) was designed by the 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*" ([40] section 1.1). This identifies the scope of UML applications.

The experience of modeling practices in modern industries shows that UML is found useful by modelers. The amount of modeling projects that use UML, the amount of books written about UML and the number of software tools that support UML are large 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, although it doesn't mean that there are no problems with the current state of UML.

Consistently with the explained scenario of a language definition, the UML specification [40] 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** we can identify is that these UML semantics are considered to be complex and difficult to understand by many modelers. OMG itself in its article "*Introduction to OMG's Unified Modeling Language (UML™)*" [37] confirms this, saying that the UML specification [40] 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 cause its complexity and by proposing a better organization of semantics for modeling concepts. In particular, 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 [43], makes a positive difference in relation with the absence of such a structure in the current UML semantics. The explicit presence of such a structure helps to understand how the modeling concepts should be used in practice, whereas its absence creates numerous possibilities for confusions in practical applications of modeling concepts.

While performing the analysis of the existing UML semantics we can localize the **second problem**. Specifically that existing UML semantics are very ambiguous in presenting relations between models constructed using the language on the one hand and the subject that is being modeled on the other hand. This is an important problem, because even an internally consistent model will not have much of practical sense when its relations with the subject that it is supposed to represent are undefined. This situation with UML is improved in our solution by the introduction of a coherent and unambiguous set of modeling concepts definitions expressing a kind of Tarski's declarative semantics [50] for the mentioned relations between the model and the subject of modeling.

The **third problem** of the UML semantics, which we will consider in this chapter, 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, the 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.

3.2. Problems Analysis based on the Foundations of UML Semantics

As we can see from the previous section, all three identified problems are related to the non-optimal 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 [40] 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.*" [40], 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.*" [40], 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.*" [40], section 2.7.1.

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

3.2.1. Problem 1: Structural Chaos of UML Semantics

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

As it is presented in Figure 2-5 from the Core package specification [40], the most general concept in the UML metamodel is called "Element". It is defined ([40],

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 it is presented in the diagrams 2-5,6,7,8,9 of the UML specifications [40], 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. And even if the elements obviously belong to different semantic categories (for example, "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 the UML specifications. For example, "Association" is defined ([40], section 2.5.2.3) relative 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 help 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 pseudo-categories 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 pseudo-categories is particularly difficult to understand.

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 the introduction of particular concepts remains obscure even on an implicit level. Surprisingly some concepts seem to appear without a significant justification whereas 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 ([40], 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 the addition of the descriptions of possible ways of presentation inside the definition of "ModelElement" would simplify the metamodel without diminishing its value.

3.2.2. 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 ([40], 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 ([40], 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 relations 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 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 than 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 conclusion 'b.' is particularly important, because it means that for the UML concepts the specification does not define any kind of formal declarative semantics that were introduced by Alfred Tarski [50]. Indeed, Tarski's declarative semantics for concepts used inside models are 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 the relations between concepts in the UML metamodel is greater 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, whereas 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 not 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[8]. In other words, in this case several mutually contradicting models can represent the same subject of modeling and all of them may be confirmed as adequate; or,

[8] 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 subject. The internal consistency of a model can be insured by the consistency of the UML metamodel, and to ensure the logical consistency of the interpretation, a kind of consistent Tarski's declarative semantics is necessary.

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 [50] 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 [40], 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 [40], 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 a differentiation of modeling concepts, thus the choice of only these two concepts to be defined using the declarative semantics is not practical.

3.2.3. 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 the integration of the best existing industrial modeling practices, but these practices were never really linked with the existing scientific theories. Although the "best practices" strategy can be considered as an attempt at 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 the UML metamodel is a theoretical problem, compared to the first two identified problems that 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.

3.3. Solutions for the Identified Problems of the UML Metamodel

3.3.1. Solution to Problem 1: Categorization of Concepts Based on Russell's Theory of Types

As we explained in Section 3.2.1, UML doesn't define any explicit logically consistent strategy for the introduction of modeling concepts. This problem was solved in the scope of our Triune Continuum Paradigm when we defined the structure of our metamodel. Section 1.1 from Chapter 1 presented this solution.

In particular, when defining the metamodel structure we followed the strategy defined in Russell's theory of types [43] to distinguish "*individual as something destitute of complexity*", "*first-order propositions*" and "*higher-order propositions*". This was done for both of the two domains participating in modeling: for the universe of discourse (that is the subject being modeled) and for the model of the universe of discourse.

As the result of this application of the Russell's theory of types, we were able to define *entities* in the universe of discourse and *model elements* in the model as Russell's individuals that are "*destitute of complexity*". Then we defined the following correspondence:

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

So the first conceptual category that was defined for the model, contained these "Model Elements". Analogously we introduced in the model "Basic Modeling Concepts" and "Specification Concepts" as two conceptual categories that correspond to Russell's first-order propositions and higher-order propositions respectively. Thus we defined the following two correspondences:

- *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.

We showed that these three conceptual categories from within the model ("Model Elements", "Basic Modeling Concepts" and "Specification Concepts") have concrete semantic differences:

- *Model Elements* in the Model are destitute of complexity and thus their only purpose is to be characterized by some of the Basic Modeling Concepts.
- *Basic Modeling Concepts* model intrinsic qualities of the Universe of Discourse entities, thus Tarski's declarative semantics [50] are defined for the concepts from this category. In the Model these concepts characterize Model Elements and can be specified by Specification Concepts.
- *Specification Concepts* contain all the other concepts in the Model (the concepts which are neither Model Elements nor Basic Modeling Concepts). Specification Concepts do not model intrinsic qualities of the Universe of Discourse entities, thus these concepts are defined only in the relations between themselves and in the relations with Basic Modeling Concepts, but not in the relation with the Universe of Discourse. In the Model these concepts may specify different Basic Modeling Concepts.

For the detailed explanation of our solution please read Section 1.1 in Chapter 1. To summarize, for the structure of our metamodel we defined a rigorous categorization that is supported by the strong theoretical foundations. The categorization clearly defines concrete application contexts for all of the different categories of modeling concepts. As we showed in Section 3.2.1, UML metamodel does not have any structural organization, which makes it very difficult for modelers to understand the application contexts for different UML concepts. Thus our categorization presents a significant positive difference in the favor of our metamodel if it is compared with the structural chaos of the currently available UML semantics.

3.3.2. 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 Section 1.2 of Chapter 1. Here we will just briefly remind the overall structure of basic modeling concepts.

The original idea of general organization for the basic modeling concepts category can be seen in Figure 1.6 from Chapter 1. Essentially the set of concepts is determined by the consideration of the spatiotemporal conceptual continuum and the non-spatiotemporal conceptual continuum. The former represents in the model a space-time from the universe of discourse, and the latter represents in the model the non-spatiotemporal conceptual entities that constitute the universe of discourse. In correspondence with their ancestors from the universe of discourse, the former is presented by the Space and the Time dimensions on Figure 1.6, 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.

By defining limiting points within Space-Time and Model Constitution dimensions we obtain 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 the 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 obtained the structure of basic modeling concepts that was presented in the diagram from Figure 1.8 in Chapter 1.

As it was noted in the previous section and in correspondence with the explanations from Section 1.1 all the basic modeling concepts have formal definitions in the form of Tarski's declarative semantics. The reader can check Section 1.2 for all these concrete definitions.

The definitions of basic modeling concepts, as well as the definitions for all the other concepts proposed in our metamodel, have much in common (and are even identical in many cases) with the definitions of corresponding concepts given by RM-ODP. As it will be presented in Chapter 5, we formalized the overall RM-ODP foundations framework, including the basic modeling concepts part, using Alloy [24] formal description technique. 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.

Detailed theoretical analysis of our approach to the definitions of basic modeling concepts can be found in Chapter 2.

3.3.3. Solution to Problem 3: Philosophical and Natural Science Foundations of our Proposed Metamodel

As we explained in the analysis of Problem 2 (Section 3.2.2), even for the choice of the 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 the very 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, the very 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.

As the previous paragraph shows, the approach to the 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 theoretical 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 Chapter 2. 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 a conceptual continuum that is independent with regard to the spatiotemporal continuum is an original idea. This idea generalizes fundamental foundations of both classical and relativistic mechanics that study spatiotemporal characteristics of material objects. In our case the scope is generalized to include 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 considered in the same context 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 Chapter 2) 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 the limited modeling scope introduced by the conceptualization provided a straightforward logical proof that the resulting limited set of basic modeling concepts is necessary and sufficient for the modeling scope representation.

3.4. Proposed Metamodel and Existing UML Concepts

In Part I in the scope of our Triune Continuum Paradigm we defined the metamodel that, as we showed in this chapter, solves the analyzed problems of the UML metamodel. Our metamodel 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 Figure 1.8 from Chapter 1: 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.

As an example, in Section 1.3 of Chapter 1 we demonstrated how the RM-ODP conceptual framework that is defined in [21] fits successfully the structure of our metamodel (see Figure 1.10 from Chapter 1). Original RM-ODP concepts are presented in Figure 1.10 with the aid of rectangles with 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 (the formalization will be presented in Chapter 5), 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 1.10. Unfortunately these expectations were proved to be naive by the research that we have done on UML specifications [40].

After having done a detailed study of UML specifications [40] 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" ([40], section 2.5.2.9), "Flow" ([40], section 2.5.2.22), "Node" ([40], section 2.5.2.29), "Operation" ([40], 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" ([40], 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 negative examples that can be found in the UML specifications.

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 ([40], 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 ([40], 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 ([40], 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 ([40], section 2.5.2.9) and "run-time physical object" from ([40], 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 their 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, and as we showed here its adaptation for the metamodeling structure is already completed.

PART II SUMMARY

In Part II of this thesis we demonstrated the constructive potential that Triune Continuum Paradigm provides to Unified Modeling Language. In particular, we identified and analysed three important problems of the UML metamodel. These were the following problems:

- Absence of an explicit structural organization defined for the UML metamodel;
- Absence of a formal declarative semantics in the UML metamodel;
- Absence of theoretical justifications for the UML metamodel to represent the modeling scope that is targeted by UML.

We showed that the metamodel that was defined in Part I in the scope of Triune Continuum Paradigm solved these problems. In particular:

- it has an internally consistent structure supported by Russell's theory of types [43];
- it defines a kind of Tarski's declarative semantics [50] for the basic modeling concepts, thus it is coherent and unambiguous in the interpretations of subjects of modeling;
- it is applied on a concrete example of the RM-ODP conceptual framework that is formalized in a computer-interpretable form;
- it 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 the UML metamodel, 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.

In Part II we also showed that UML terminology in its current state contains self-contradictions and baseless references (references to undefined concepts). We demonstrated that existing UML terminology doesn't allow for its logically consistent interpretation. This problem with terminology is probably the most fundamental problem of UML.

The problem of UML terminology and the three problems of UML metamodel clearly demonstrate that at the present time from the theoretical standpoint, UML assimilates itself to the naked Emperor from the famous story: "The Emperor's New Clothes" by Hans Christian Andersen (1805-1875). In the story those who were not able to see the most magnificent cloth of the Emperor were considered as either

stupid or incompetent. So everybody, including the Emperor himself, pretended to be able to see the cloth regardless of its nonexistence. This continued until people heard the voice of an innocent child: "But the Emperor has nothing at all on!". Then the people began to whisper to one another what the child had said, till everyone was saying: "But he has nothing on!". The Emperor himself understood that the people were right, but he was not courageous enough to publicly acknowledge the fault. "*So he drew himself up and walked boldly on holding his head higher than before, and the courtiers held on to the train that wasn't there at all.*" – the story ends.

The analogy with UML semantics is straightforward. In Part II of the thesis we showed:

- that the UML metamodel does not exhibit some of the essential metamodeling features;
- that the UML metamodel does not have any theoretical support to justify its claims to an adequate representation of the identified (by UML) modeling scope;
- that the UML terminology definitions are not even a house of cards that can be ruined by a light wind, but rather the ruins from which it is impossible to reconstruct a single coherent framework.

We acknowledge the comparative practical domination of UML in the current industrial modeling practices. We do not call in question the relatively successful definition of constraints for practical applications of UML formulated in section 3 ("UML Notation Guide") of UML specifications [40]. But as we showed, from the theoretical standpoint UML "has nothing at all on!". And obviously, even successfully promoting UML will not eliminate this very important problem. "The Emperor" may keep "walking boldly on holding his head higher than before", pretending that "the most magnificent cloth" is "*very difficult for beginners to understand*" [37] but the theoretical "cloth" for UML will not appear by itself.

In Part I of the thesis we provided concrete solutions for the problems of UML that were analyzed in Part II.

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 the OMG's attention to the results of our research.

PART III

Triune Continuum Paradigm

Application to RM-ODP

4. EXAMPLE OF THE PARADIGM IMPLEMENTATION: INTRODUCTION TO THE FORMALIZATION OF RM-ODP CONCEPTUAL FRAMEWORK

With the aid of Triune Continuum Paradigm we achieved an important result: a consistent formalization of the RM-ODP conceptual framework. Chapter 4 introduces all the necessary background information for the computer-interpretable form of the formalization that will be presented in Chapter 5. In particular, in this chapter the reader will:

- become acquainted with the structure of the RM-ODP standard;
- become acquainted with a review of the previous research that was done on the RM-ODP formalization;
- see the definition of a concrete scope for the formalization;
- become acquainted with Alloy, a concrete formal description technique that we used to present our formalization in a computer-interpretable form.

This chapter:

- is potentially interesting for the readers who want to see a concrete implementation of Triune Continuum Paradigm;
- is particularly important to the readers who are interested in RM-ODP and its formal foundations.

As we explained in Chapter 1 (Section 1.3) and in Chapter 3 (Section 3.4), different object-oriented frameworks can benefit from advantages of the metamodel of our defined Triune Continuum Paradigm. The advantages include in particular: the metamodel's internal consistency, its logical coherency of interpretation, its formalized semantics and theoretically justified foundations.

As an example, in Section 1.3 of Chapter 1 we demonstrated how RM-ODP conceptual framework that is defined in [21] fits successfully the structure of our metamodel (see Figure 1.10 from Chapter 1). Thus, with the aid of our defined Triune Continuum Paradigm we were able to formally interpret RM-ODP conceptual framework as a single consistent construction.

In fact such formalization was officially defined as one of the goals of the RM-ODP standard (in particular, of its part 4), but was never successfully accomplished. So, with the aid of our paradigm we provided a result, whose importance is not only justified by the ISO/ITU standard's goal, but also emphasized by the current absence of any solution that would achieve the goal.

To promote the value of this result we decided to represent the formalization of the RM-ODP framework in a computer-interpretable form. This chapter will introduce all the necessary background information that is related to our RM-ODP formalization, and the next chapter will present the formalization itself.

4.1. RM-ODP International Standard: an Introduction for the Formalization

4.1.1. Analysis of the RM-ODP Standard

Let us first position ourselves with regard to the different parts of RM-ODP[9]. There are four parts in the standard. Part 1 will not be considered for formalization since it contains a motivation overview of ODP and is not normative. Part 2 of the standard introduces ODP concepts and is the core of our formalization work. The work may be continued in future with part 3 that presents the constraints to which ODP standards must conform. Part 4 of RM-ODP describes the recommendations for approaching the standard formalization with LOTOS ([20], [30]), ACT ONE [16], SDL-92 [22], Z ([13], [48]) and ESTELLE [19] languages. Unfortunately, these are just recommendations or, in the best cases, small unrelated pieces of formalization for the part 2 concepts presented with different formal techniques. As a matter of fact, while the goal of the part 4 is to present "*a formalisation of the ODP modeling concepts defined in ITU-T Rec. X.902 | ISO/IEC 10746-2, clauses 8 and 9, and a formalisation of the viewpoint languages of ITU-T Rec. X.903 | ISO/IEC 10746-3*" (see [21]), no single consistent formalization representing the clauses 8 and 9 of part 2 can be found there. Also some of the ODP practitioners tend to consider the formalization of the RM-ODP part 2 is achieved simply because the ITU-T Rec. X.904 | ISO/IEC 10746-4 is standardized. This is unfortunate because the standard fails to provide a true consistent formalization of the concepts of part 2. This lack is also noticed in [42], which says: "*It is definitely the case that the degree of formalism in architecture specifications today varies, and none, including RM-ODP, have achieved a fully mathematical formal specification using an appropriate formal description technique*". Unfortunately the book [42] leaves out the part 4 in its consideration. All of this provided us with excellent motivation for this research on the RM-ODP formalization and emphasizes the value of its results.

Analyzing the causes that prevented the standardization of a single consistent formalization of the clauses 8 and 9 of part 2, we would argue that the approach that

[9] Each time when we mention RM-ODP in the thesis, we refer to [21].

was taken in part 4 hardly favors this kind of formalization. Detailed explanations of the approach may be found in [44]. The approach never considers explicitly the relations that exist between different concept categories (such as between basic modeling concepts (RM-ODP 2-8) and specification concepts (RM-ODP 2-9)). In addition, it considers the concepts from RM-ODP 2-8 and 2-9 without considering the basic interpretation concepts (RM-ODP 2-6), that is, it considers the concepts used within models without a relation to the concepts representing the universe of discourse being modeled. In summary, the approach abstracts from the categorization of concepts (RM-ODP 2-5), which makes the single holistic framework informally presented in part 2 impossible to represent formally in part 4.

However, we never doubted that the currently existing version of part 4 makes sense. Each of the formal languages presented there was chosen for a particular limited scope of the standard applications. Thus, in these reduced scopes the suggested approach for formalization has successfully proved its value.

For example, chapter 4.1 of part 4 presents the standard architectural semantics in LOTOS ([20], [30]). As a result, they are oriented towards the simulation of the execution of ODP models.

Our result is different; we present a formal metamodel of the standard that allows ODP models to be verified and checked for consistency. Our metamodel of RM-ODP part 2 stays on the same conceptual level as the UML metamodel [40] thus providing potential for UML to be influenced by RM-ODP. Hence, we preserve the generic essence of the ODP framework with regard to the potential applications.

Positioning itself as the standard metamodel, our approach presents a significant advantage in comparison with those described in chapters 4-1, 4-2, 4-3, 4-4 and 4-5 of RM-ODP. Specifically, we aim to formalize definitions and mutual relations not only for the modeling concepts (the RM-ODP's basic modeling concepts and specification concepts), as it is in the part 4, but for all the other relevant ODP concept categories. Hence, we benefit from the completeness of the scope definition within the RM-ODP standard and are able to show clear relations between the universe of discourse being modeled and the model of it (including the basic modeling part and the specification part). We heavily emphasize the importance of RM-ODP 2-5 (categorization of concepts) that is often ignored by ODP practitioners - perhaps due to its relatively implicit definition in the standard.

4.1.2. Analysis of Previous Research on the RM-ODP Formalization

A formal view on RM-ODP specifications insures their consistency within a frame of a particular project, and the ODP research community has produced several interesting results that are important for understanding the consistency and for implementing consistent specifications. Particularly, [10] presented a nice discussion

on the requirements that the RM-ODP framework imposes on different Formal Description Techniques (FDTs) for its formal interpretation. [10] considered a set of general ODP requirements and elaborated on the requirements for specific ODP viewpoints. Further research ([8], [9], [29]) defined a general meaning of the specification consistency in the context of RM-ODP viewpoints. Based on these papers, relating formalisms used for different viewpoints became a standard approach for ODP formalization work. This kind of the problem positioning was previously considered independently from the context of the RM-ODP standard; for example, the analogous question of "multiperspective specifications" consistency was discussed in [18].

In the case of RM-ODP, the viewpoints are well defined in the standard, - this allowed publications of some successful case studies. For instance, studying the interrelations of the viewpoints, [2] presents mappings between the information viewpoint and the computational viewpoint languages; [7] relates the computational viewpoint with the engineering viewpoint.

At the same time another research thread concentrates only on formalizations for specific viewpoints. [15], [49] propose approaches for the enterprise specifications and [35], [36], [45] formalize the ODP computational model. Examples of approaches for computational, engineering and technology viewpoints can be found in [6]. Thus, historically the emphasis in ODP formalization research was put on the viewpoints.

The idea behind our work differs from those mentioned in the previous paragraph. Its originality is to consider formalization of the RM-ODP foundations presented in the part 2 of the standard, rather then formalization of ODP viewpoints introduced in the part 3. This choice is justified by the standard, since the ODP conceptual framework from the part 2 is defined to support ODP viewpoints. It presents a general vision on modeling, the vision that should further be applied in the context of a particular ODP viewpoint.

In particular, [44], reporting on experiences out of the standard development, clarifies: "*The relationship between Part 2 and Part 3 of the RM-ODP may be seen as specialization. That is Part 2 gives a basic interpretation of a given concept and Part 3 gives a more specialized version.*"

The importance of a formal view on the part 2 of RM-ODP was noted not only by the standard itself but also in ([26], [27]). Specifically, [26] is saying: "*RM-ODP emphasizes common fundamental concepts encountered in any open distributed system, including distribution-independent concepts! It is used for descriptions of any system, not just software; and it includes both viewpoint-specific concepts, as well as – perhaps, more importantly – concepts common to all viewpoints – enterprise, information, computational, and so on.*" Thus by formalizing part 2 of RM-ODP, we provide a general, consistent and complete framework for

modeling, suitable for further applications in the less general but more precise contexts of the viewpoints.

4.1.3. RM-ODP Part 2: Scope for Formalization

As we explained the motivation for this work on the RM-ODP formalization, let us now define the sections of RM-ODP part 2 that should be formalized. In part 2 there are 15 clauses. Clauses 1-4 are auxiliary to the rest of part 2 and will not be considered for our formalization. These clauses serve to introduce "RM-ODP part 2: Foundations" in the overall context of the standard. Specifically, they:

- define the scope of "RM-ODP part 2: Foundations" (RM-ODP 2-1);
- refer to the general rules of ISO and ITU standardization (RM-ODP 2-2);
- list "*Background definitions*" for the general terms used in the standard references (RM-ODP 2-3);
- define the abbreviations used in the standard (RM-ODP 2-4).

Further, in the scope definition (RM-ODP 2-1) it is mentioned that: "*This ITU-T Recommendation | Part of ISO/IEC 10746 covers the concepts which are needed to perform the modelling of ODP systems (clauses 5 to 14), and the principles of conformance to ODP systems (clause 15).*" Thus, clause 15 will not be considered for the formalization.

Clauses 10-14 represent the so-called "structuring concepts". As they are defined in RM-ODP 1-6.2.1: "*structuring concepts - building on the basic modelling concepts and the specification concepts to address recurrent structures in distributed systems, and cover such concerns as policy, naming, behaviour, dependability and communication.*" Essentially, these are the concepts defined by means of the RM-ODP's basic modelling concepts and the specification concepts (clauses 8 and 9). Consequently, they could be formalized as soon as RM-OPD 2-8 and 2-9 concepts are formalized. Since the structuring concepts from 2-10 – 2-14 are specific for the particular aspects of distributed system modeling, their formalization will also not be considered in this work.

Clause 7, "Basic linguistic concepts", introduces two concepts: "Term" and "Sentence". These are linguistic constructs that should be used for expressions in any language that could be employed for the description of ODP modeling. So, these are concepts that are defined on the meta-level for the RM-ODP metamodel (meta-meta-level concepts) and they will not be considered for our formalization.

We will formalize all the remaining clauses of the part 2, namely the clauses 5, 6, 8 and 9. They represent the kernel of the RM-ODP framework, including:

- RM-ODP 2-5: "Categorization of concepts";
- RM-ODP 2-6: "Basic interpretation concepts";
- RM-ODP 2-8: "Basic modelling concepts";
- RM-ODP 2-9: "Specification concepts".

4.2. Formalization Language: Alloy

As we explained in Section 1.3 (Chapter 1), using Triune Continuum Paradigm that was defined in Part I of the thesis, we achieved a complete formalization of the RM-ODP conceptual framework. For example, to interpret the categorization of RM-ODP concepts (that was not explicitly defined in the standard) we used Russell's theory of types [43], - the technique that ensures logical consistency of conceptual categorization in the paradigm. Thus all the necessary techniques for the formal interpretation of the RM-ODP conceptual framework were already defined in the paradigm, and the only remaining step was to present the formalization in a computer-interpretable form. To accomplish this step we needed some concrete formal description technique. We chose Alloy [24], - the language for description of structural properties of a model.

Alloy was chosen as one of the simplest modeling notations with semantics that are "*expressive enough to capture complex properties while remaining amenable to efficient analysis*" [24]. It is interesting that Alloy was developed having Z as a starting point and adopting Z for the object modeling (for the comparison of Alloy and Z see [23]). Z, the origin of Alloy, was used in the standard part 4, - this gives an additional argument to support our language choice. Another advantage of this choice is the public availability of the associated tool for checking specifications written in Alloy. With the tool, all the Alloy models presented in this thesis can be tested and used in future research. We should mention that Alloy, as well as any other possible language that we could have chosen, does not impose any restriction on our argumentation presented in the thesis. We just picked the language that was best for our presentation needs and we do not intend to discuss the general advantages or disadvantages of Alloy in comparison with other languages. Based on the paradigm that we defined in Part I of this thesis, our framework can also be expressed using another formal description technique.

Here we will briefly describe the basic structure of the Alloy language that will help to read our formal models.

An Alloy model starts from the declaration of the model name and contains two main paragraphs: domain and state. In the domain declaration we find the Alloy domain names: "*These are basic sets that are disjoint from one another*" [25]. For example:

```
model RM-ODP {
domain {ODP_Concepts}
state {
        partition FirstSubset, SecondSubset : static ODP_Concepts
        firstRelation (~secondRelation) : FirstSubset -> SecondSubset
 }
}
```

Here we have an Alloy model that is called "RM-ODP" and has a single top-level domain that is called "ODP_Concepts". Having declared the domains, Alloy allows for the declaration of their subsets and of the relations between the elements of these subsets, thus introducing the model structure. In Alloy models this is done within state schema. In the example above we see the declaration of static partitioning of the ODP_Concepts domain into two subsets: FirstSubset and SecondSubset. Also we see the declaration of two relations: firstRelation and secondRelation, where firstRelation relates elements from FirstSubset to elements of SecondSubset, and secondRelation – elements from SecondSubset to elements of FirstSubset.

Another important Alloy schema is called def. It is used for the definitions of additional constraints to which either the elements from a declared set or some of the declared relations should obey. Thus in most cases, to formalize an RM-ODP part 2 concept we will use two steps in Alloy: the concept declaration within state schema, and the concept definition within def schema that is done by means of additional constraints that are applicable to the concept.

Two other Alloy schemas that will appear in our models will be inv and op. The former is the invariant schema; it "*introduces a state invariant*" [25]. While the latter is the operation schema; it "*introduces an operation that describes a set of state transitions. Its body is usually a formula that includes primed and unprimed components*" [25]. Primed components within an operation formula correspond to the state after the operation and those unprimed – before the operation.

In Alloy "*markings at the ends of relation arrows denote multiplicity constraints:* ***!*** *for exactly one,* ***?*** *for zero or one,* ******* *for zero or more and* ***+*** *for one or more. Omission of a marking is equivalent to* *******." [23]. Logical operators and words reserved for Alloy quantifiers and relations are [24]:

logic-op ::= **&&** / **||** / **–>** / **<–>**
negate ::= **not** | **!**
comp-op ::= **in** / **=** / *negate* **in** / *negate* **=** / **/=** / **/in**
quantifier ::= **all** | **some** | **no** | **sole** | **one**
expr-op ::= **+** / **-** / **&**

This concludes our brief description of Alloy language, for more details the reader may check [25], [23], [24].

5. FORMALIZATION OF RM-ODP CONCEPTUAL FRAMEWORK

The previous chapter introduced background information for the formalization of RM-ODP conceptual framework. Chapter 5 will present the formalization itself. In particular, in this chapter the reader will:

- see how the concepts defined in the RM-ODP standard are represented with the aid of Triune Continuum Paradigm and Alloy formal description technique.

This chapter as well as the previous one:

- is potentially interesting for the readers who want to see a concrete implementation of Triune Continuum Paradigm;
- is particularly important to the readers who are interested in RM-ODP and its formal foundations.

5.1. Categorization of the RM-ODP Conceptual Categories

"RM-ODP part 2: Foundations" introduces ODP concepts that are necessary to perform the modeling of ODP systems. Part 2 clause 5 introduces different categories of ODP concepts. Here we will formalize the relations between the concept categories defined as relevant for formalization in Section 4.1.3 (Chapter 4) of the thesis.

First of all, we stress the importance of a formal view on the categorization of ODP concepts. The motivation to such a categorization was already considered in Chapter 3, when we analyzed structural problems of the UML metamodel. In comparison with UML that did not define any conceptual categorization for its metamodel, RM-ODP presents a considerable advantage defining different conceptual categories that were discussed in Section 4.1.3 (Chapter 4).

The RM-ODP standard clause 2-5, "Categorization of concepts", should define relations between the concept categories introduced in the other clauses (6, 8 and 9). Unfortunately, in the current version of RM-ODP the clause 5 statements are vague and do not express explicitly these relations.

However, as we already mentioned, these relations have an enormous importance not only for the standard formalization, but also for its use in practice. Without them it's impossible to define a particular conceptual category (its sense and functionality) within the RM-ODP framework. Thus, if these relations are not explicitly defined, it

is impossible to understand the context in which the concepts of a particular category should be used and the one in which they should not be used. This often leads to a misunderstanding of the concepts thus resulting in multiple confusions within practical applications of RM-ODP.

As a good example we refer to [42], which in the very paragraph that emphasizes the importance of formalism for specifications ([42]: 18.1.3), also says that the use of a formal description technique "*allows propositions about entities of the model to be well-founded*". So [42], in the context of RM-ODP, tries to put entities and propositions about them within the model. There is a fundamental confusion here because entities and propositions are defined in RM-ODP as basic interpretation concepts contributing to the universe of discourse: the universe of discourse is the subject being modeled and it is not at all a possible resulting model representing this subject. The corresponding relation was already described in the thesis (for example, in Figure 1.1 or in Section 2.1 in Part I) and it will be formally presented on the example of RM-ODP later in this chapter.

Thus on the one hand, as the consequence of the vagueness of clause 5, RM-ODP leaves the question about a particular interpretation of its existing conceptual categorization without an explicit answer. And on the other hand, a formalization of the relations between existing RM-OPD conceptual categories is a very important issue, because without it we cannot perform a consistent formalization of the different RM-ODP conceptual categories in a single context of the standard formalization.

So, for the RM-ODP conceptual categorization it was necessary to define a framework of concrete interpretation rules that would be compatible with the existing RM-ODP definitions from one side, and that would provide all the necessary formal reasoning for the RM-ODP concepts applications from the other side. Our Triune Continuum Paradigm that was defined in Part I perfectly satisfies these criteria. Thus with the aid of the paradigm we were able to present a single consistent formalization of the RM-ODP conceptual framework.

Let us start the formalization by constructing the Alloy model introducing interrelated conceptual categories. Later on they will be elaborated upon and will have their own parts in the overall Alloy model, the parts that will correspond to their respective clause definitions from part 2 of RM-ODP.

5.1.1. Introduction of Basic Interpretation Concepts

The basic interpretation concepts described in part 2 clause 6 define the universe of discourse being modeled (6.1 "Entity", 6.2 "Proposition", 6.5 "System") and introduce (for modelers) the possibilities of interpretation of the universe of discourse (6.3 "Abstraction", 6.4 "Atomicity", 6.6 "Architecture").

To position different categories of ODP concepts, let us introduce RM-ODP model in Alloy:

```
model RM-ODP {
domain {ODP_Concepts}
state {
        BasicInterpretationConcepts : ODP_Concepts
        partition UniverseOfDiscourse, InterpretationPossibilities : static
BasicInterpretationConcepts

// … to be completed with the other concept categories
}}
```

The model says that in ODP_Concepts there is a category BasicInterpretationConcepts, which is partitioned in UniverseOfDiscourse and InterpretationPossibilities. Now, having introduced BasicInterpretationConcepts, we may explore them in a separate model that would correspond to RM-ODP part 2 clause 6.

The InterpretationPossibilities part of BasicInterpretationConcepts contains concepts of:

- Abstraction (2-6.3), that allows for different levels of details to exist when modeling the universe of discourse;
- Atomicity (2-6.4), that allows the definition of granularity for a given level of abstraction;
- Architecture (2-6.6), that introduces a set of rules that define the structure of a system in the universe of discourse.

As we see, these are the concepts defined on the meta-level for the RM-ODP metamodel framework (meta-meta-level concepts). On the same meta-meta-level we find RM-ODP 2-7: "*Basic linguistic concepts*" that was considered irrelevant for formalization in Section 4.1.3 (Chapter 4) of the thesis. Since here our goal is to formalize RM-ODP metamodel and not the meta-meta-view of RM-ODP modeling framework, we will also discard the InterpretationPossibilities concepts from our formalization.

Let us now consider the concepts related to the UniverseOfDiscourse part of the basic interpretation concepts.

The universe of discourse consists of entities (defined in 6.1 as "*any concrete or abstract thing of interest*") and propositions that can be asserted or denied to be hold for entities (defined 6.2). The concept of system is defined as a kind of entity and the concept of subsystem as a kind of system (definition 6.5). This allows us to present the domain that corresponds to the UniverseOfDiscourse in Alloy model for the BasicInterpretationConcepts:

```
model ODP2-6 {
```

```
domain {UniverseOfDiscourse}
state {
        partition Entity, Proposition: UniverseOfDiscourse
        holds : Proposition –> Entity+
        System : Entity
        Sybsystem : System
       }
inv AssertOrDeny {
        all a: Entity, b: Proposition | (a in b.holds) || (a not in b.holds)
       }
}
```

As we can see, the universe of discourse proposed by clause 6 (consisting essentially of entities and propositions over entities) is defined in correspondence with Triune Continuum Paradigm that was defined in Part I. In particular, it is organized in correspondence with Russell's theory of types [43] (that has individuals and propositions over individuals). Let us repeat here the already quoted (in Section 1.1 of Chapter 1) part of [43] that 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 the case of RM-ODP we have "entity" (defined in 2-6.1) corresponding to Russell's "*something destitute of complexity*", because the only intrinsic meaning of an entity is to be "something" that can be qualified by means of propositions. An entity has no other meaning without the associated to it propositions. Thus, by mapping Russell's "individual" and "proposition" to RM-ODP's "entity" and "proposition" correspondingly, we can use Russell's suggestion in the context of the universe of discourse from the RM-ODP clause 6. This allows us to differentiate the propositions with regard to their subject of application in a direct correspondence with Triune Continuum Paradigm:

- 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, as it was already described in Chapter 1, in an application of any of the higher-order propositions we can always descend trough all the lower-order propositions down to the entities, on which this structure of multiple propositions is applied. So, the higher-order propositions, as well as the first-order propositions, satisfy the RM-ODP 2-6.2 definition.

Let's define this structure of propositions in the Alloy model (here and further for the clarity of reading we put the text added to the previous version of the model in the boldfaced type):

```
model ODP2-6 {
domain {UniverseOfDiscourse}
state {
        partition Entity, Proposition: UniverseOfDiscourse
        partition FirstOrderProposition, HigherOrderProposition: Proposition
        holds : Proposition –> UniverseOfDiscourse+
        System : Entity
        Subsystem : System
      }
inv AssertOrDeny {
        all a: UniverseOfDiscourse, b: Proposition | (a in b.holds) || (a not in b.holds)
      }
def FirstOrderProposition {
        all p: FirstOrderProposition | (p.holds: Entity)
      }
def HigherOrderProposition {
        all p: HigherOrderProposition | (p.holds: Proposition)
      }
}
```

This is the complete (within RM-ODP modeling framework) Alloy model for the "Basic interpretation concepts" category. It introduces "universe of discourse" as a subject that is of interest for a modeler, who will create a subjective representation of it within his/her models. To construct models for a given universe of discourse, RM-ODP proposes two conceptual categories: "Basic modelling concepts" and "Specification concepts".

5.1.2. Introduction of Basic Modelling Concepts

"*Basic modelling concepts*" defined in part 2 clause 8 is the first conceptual category that allows to construct models for a given universe of discourse. Hence, now we

will need to understand, first of all, the general positioning of "*basic modelling concepts*" within a model of a given universe of discourse.

In correspondence with Triune Continuum Paradigm (see Part I of the thesis), let's assume that Russell's theory of types [43] may be applied to the model as well as it was applied to the universe of discourse in the previous section. Then within the model we will be able to identify the 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*", and some concepts – analogs of the "*higher-order propositions*".

Now, let's look on the "*basic modelling concepts*" (RM-ODP 2-8), and let's try to understand how they can be used within models. We may particularly note, that as soon as there will be an entity and a proposition applied to the entity in the universe of discourse, a modeler should be able to represent this in the model by means of a model element that will be characterized by a RM-ODP basic modelling concept. So the RM-ODP basic modelling concepts can be used within the model as the Russell's "*first-order propositions*". Apart from that, it's easy to see that because of the intrinsic complexity introduced in their definitions, the basic modelling concepts cannot be used as the Russell's "*individuals*". And also, using the definitions from 2-8, it's easy to check that the basic modelling concepts cannot be used as the Russell's "*higher-order propositions*".

Consequently, we may conclude that the application of Russell's theory of types on the model is compatible with all the possible applications of the RM-ODP basic modelling concepts within the model. And, analogously to what was explained in Chapter 1, the basic modelling concepts are essentially the first-order propositions about model elements.

Now, as we understood the positioning of the basic modelling concepts within the model, we may define a correspondence between them and the universe of discourse that is modeled. Applying Triune Continuum Paradigm, we are determined to model the first-order propositions from the universe of discourse by means of the first-order propositions in the model (that is by means of the basic modelling concepts). The justification of this application will be discussed further in Section 5.1.4.

Now, concluding this discussion, we may introduce the basic modelling concepts category and its relation with the universe of discourse in our Alloy model of RM-ODP:

```
model RM-ODP {
domain {ODP_Concepts}
state {
    partition BasicInterpretationConcepts, BasicModellingConcepts : static
ODP_Concepts
```

```
        partition UniverseOfDiscourse, InterpretationPossibilities : static
BasicInterpretationConcepts
        partition Entity, Proposition: UniverseOfDiscourse
        partition FirstOrderProposition, HigherOrderProposition: Proposition
        holds : Proposition –> UniverseOfDiscourse+
        modeledByBMC : FirstOrderProposition –> BasicModellingConcepts

// … to be completed with the other concept categories
}}
```

Which is to say that now we have another category in ODP_Concepts, called BasicModellingConcepts, and that we can define relation named modeledByBMC between FirstOrderProposition and BasicModellingConcepts. The last relation represents that the first-order propositions from the universe of discourse (propositions about entities) should be described by means of basic modeling concepts in the model.

5.1.3. Introduction of Specification Concepts

"Specification concepts" defined in part 2 clause 9 of RM-ODP is the second and the last conceptual category that allows to construct models for a given universe of discourse.

The same reasoning that was used in the previous chapter for the basic modelling concepts and the first-order propositions may be successfully employed again, but now for the RM-ODP specification concepts and the higher-order propositions. Thus, we may conclude that the application of Russell's theory of types on the model is compatible with all the possible applications of both the basic modelling concepts and the specification concepts within the model. And, in correspondence with Triune Continuum Paradigm, in the model the RM-ODP specification concepts are essentially the higher-order propositions applied to the first-order propositions about the model elements.

That is the specification concepts may apply either to the basic modelling concepts or to the specification concepts themselves. But in general, as it is with any of the higher-order propositions, we can always descend trough all the lower-order propositions down to the model elements, on which this structure of multiple propositions is applied. Thus, no matter how complex a structure of higher-order propositions is, it will always apply to a first-order proposition that, in its turn, will apply to a model element. In the model an application of a first-order proposition (a basic modelling concept) specialized with its higher-order propositions (specification concepts) on a model element will result in a single assertion that will qualify the model element.

Now, as we understood the positioning of specification concepts within the model, we may define a correspondence between them and the universe of discourse that is modeled. Applying Triune Continuum Paradigm, we are determined to model the higher-order propositions from the universe of discourse by means of the higher-order propositions in the model (that is by means of the specification concepts). The justification of this application will be discussed further in Section 5.1.4.

Hence, we may introduce the specification concepts category and its relation with the universe of discourse in our Alloy model of RM-ODP:

```
model RM-ODP {
domain {ODP_Concepts}
state {
        partition BasicInterpretationConcepts, BasicModellingConcepts, SpecificationConcepts
: static ODP_Concepts
        partition UniverseOfDiscourse, InterpretationPossibilities : static
BasicInterpretationConcepts
        partition Entity, Proposition: UniverseOfDiscourse
        partition FirstOrderProposition, HigherOrderProposition: Proposition
        holds : Proposition –> UniverseOfDiscourse+
        modeledByBMC : FirstOrderProposition –> BasicModellingConcepts
        modeledBySC : HigherOrderProposition –> SpecificationConcepts
      }
}
```

Which is to say that now we have yet another category in ODP_Concepts, called SpecificationConcepts, and that we can define a relation named modeledBySC between HigherOrderProposition and SpecificationConcepts. The last relation represents that the higher-order propositions from the universe of discourse (propositions on propositions about entities) should be described by means of the specification concepts in the model.

5.1.4. Relation between the Universe of Discourse and its Models

Three previous sections of this chapter introduced the correspondence of the RM-ODP modeling framework to one of the fundamental ideas of Triune Continuum Paradigm: existence of the universe of discourse (as a subject of interest for modeling) and existence of models of the universe of discourse (constructed by means of the basic modelling and the specification concepts). The fundamental difference of these two domains was discussed in Section 2.1 of Chapter 2. Here, having the goal to understand the consequences of this idea for our Alloy formalization, let us review the most important principles that apply to the relations between the universe of discourse and its models.

First of all, the universe of discourse and any of its models are two different things, - in any case they have no common parts. Consequently, in our Alloy formalization we have disjoint sets for the universe of discourse (UniverseOfDiscourse) and for the model (union of BasicModellingConcepts and SpecificationConcepts). The model is always under the responsibility of its modeler. Starting from the initiation (that is the decision to consider a particular universe of discourse as relevant for modeling), the content of the model is completely controlled by the modeler's decisions. This allows for an argumentation about the content of the model. While, in general case, the universe of discourse content is controlled by nobody, thus nobody has enough reasons to argue about it.

Our formal vision of relations between the universe of discourse and its models, defined in the scope of Triune Continuum Paradigm, is in agreement with basic principles of philosophy of science (see [1] on "philosophy of science").

In sciences we find a subject of investigation observed from the perspective of a given science, and posited truths proposed by the science in relation with the subject. Respectively, in the paradigm (and in the RM-ODP formalization as in a concrete example of the paradigm application) we find the universe of discourse as the subject of investigation from one side, and the model built by means of the basic modeling and specification concepts from the other side. So the model here is analogous to the "*posited truths concerning the world*" ([1] on "philosophy of science").

A science proposes the framework to model the subject of scientific investigation; analogously Triune Continuum Paradigm (and RM-ODP as a concrete example of its application) proposes the framework to model the universe of discourse. Any science is based on an application of the axiomatic method (see [1] on "axiomatic method"), analogously in Triune Continuum Paradigm (and consequently in RM-ODP) we also have all the necessary parts of the method application.

The correspondence of Triune Continuum Paradigm to the axiomatic method is shown in Appendix A; there it is shown that the decisions:

i. from Section 5.1.2: to model the perceived entities from the universe of discourse by means of the model elements in the model
ii. from Section 5.1.2: to model the perceived first-order propositions from the universe of discourse by means of the first-order propositions in the model (that is by means of the basic modelling concepts)
iii. from Section 5.1.3: to model the perceived higher-order propositions from the universe of discourse by means of the higher-order propositions in the model (that is by means of the specification concepts)

correspond to "axioms" in terminology of the axiomatic method, the axioms "*whose truth is knowable immediately without the use of deduction*" [1].

Statements (ii) and (iii) led to the introduction of modeledByBMC and modeledBySC formal relations in our Alloy model. These relations as well as the relation between

the entities and the model elements (i) are presented in the Figure 5.1 by means of dashed arrows.

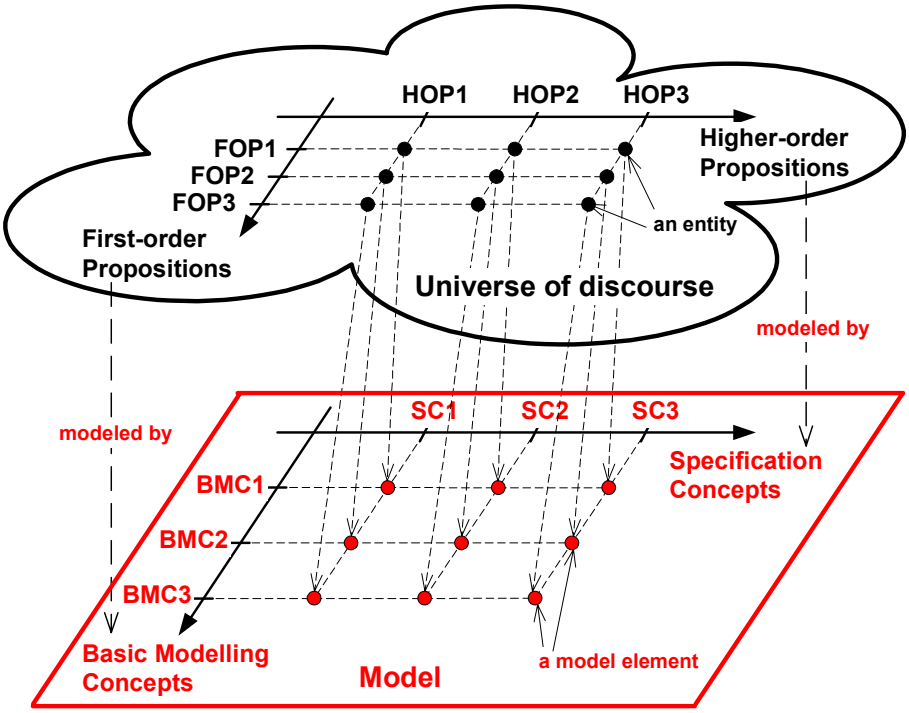

Figure 5.1: Relation between the *entities*, *first-order propositions* (FOP) and *higher-order propositions* (HOP) from the universe of discourse and their respective *model elements*, *basic modelling concepts* (BMC) and *specification concepts* (SC) in the model.

At this point we can return to the sections 5.1.2 and 5.1.3 where we promised to provide justifications for the decisions (i), (ii) and (iii). Now it's clear that these are axioms that we decided to postulate for Triune Continuum Paradigm, and as the consequence, for our formalization of RM-ODP.

As soon as a modeler decides to model some universe of discourse, he/she has no choice but to decide how the universe of discourse elements should be represented in the model. A concrete choice for this representation has no a priori restrictions. So, different choices of axioms could have been formulated for the representation, but some choice of the axioms on the universe of discourse representation is inevitable.

We find the three axioms to be reasonable with regard to our own modeling experience. In particular, our chosen axioms allow successful practical applications of the RM-ODP part 2 clauses 6, 8 and 9.

As we explained in this chapter (and also in parts I and II of the thesis), the relation between the universe of discourse and its model is a very important relation.

In particular, it explains the limits of the model and thus the limits of the modeler's responsibilities within a modeling project. Thus we think that this relation should be mentioned explicitly in the categorization of concepts part of the RM-ODP foundations.

5.1.5. Relation between Basic Modelling and Specification Concepts

In this section we would like to emphasize the independence of the RM-ODP basic modelling concepts category and the RM-ODP specification concepts category. Indeed, in part 2 clause 5 the RM-ODP standard defines them independently. That is, the basic modelling concepts can very well exist without any notion of the specification concepts and vice versa. Each of the categories gives the possibility for the universe of discourse to be projected into the corresponding conceptual dimension. And because of the categories' independence, the resulting projections are orthogonal views on the universe of discourse (see Figure 5.1).

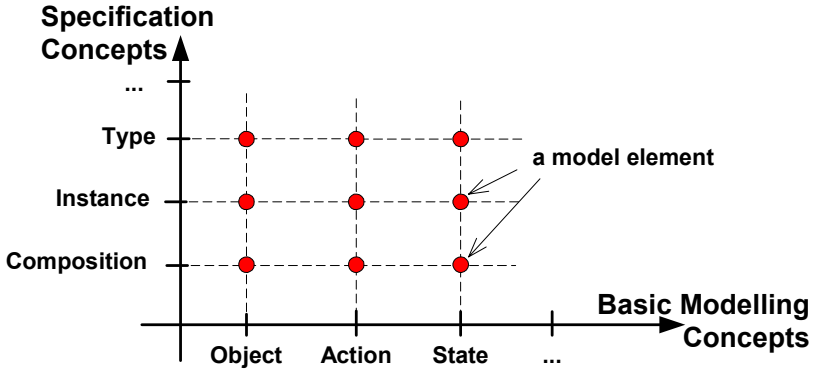

Figure 5.2: Set of model elements seen as Cartesian product of the basic modelling concepts set and the specification concepts set.

The generic higher-order propositions for the universe of discourse do not depend on the first-order propositions to which they can be potentially applied. And the first-order propositions from the universe of discourse do not depend on the higher-order propositions that they can potentially have. Consequently, the generic specification concepts[10] do not depend on the basic modelling concepts that they can characterize, and the basic modelling concepts do not depend on the specification concepts that can characterize them. For example (see Figure 5.2), "Type" specification concept (RM-ODP 2-9.7) can be applied to any of the basic modelling

[10] Alloy sub-categorization of the RM-ODP specification concepts into generic and specific specification concepts will be introduced in Section 5.3 in correspondence with Triune Continuum Paradigm (see Section 1.3.2 of Chapter 1).

concepts (such as "Object", "Action", "State", etc); or "Action" basic modelling concept (RM-ODP 2-8.3) can be characterized by any of the generic specification concepts (such as "Composition", "Instance", "Type", etc).

Thus, the link between a basic modelling concept and a specification concept can be established only by means of a model element corresponding to an entity from the universe of discourse being modeled. For the mapping to exist, concepts of both kinds should characterize the same model element. In other words, the mapping between a basic modelling concept and a specification concept is equivalent to the concrete assertion (about a model element) that uses this basic modelling concept specialized by this specification concept. We can explain the mapping formally introducing the corresponding modifications to the RM-ODP model in Alloy:

```
model RM-ODP { // corresponds to RM-ODP 2-5
domain {ODP_Concepts}
state {
        partition BasicInterpretationConcepts, BasicModellingConcepts, SpecificationConcepts :
static ODP_Concepts
        partition UniverseOfDiscourse, InterpretationPossibilities : static
BasicInterpretationConcepts
        partition Entity, Proposition: UniverseOfDiscourse
        partition FirstOrderProposition, HigherOrderProposition: Proposition
        holds : Proposition –> UniverseOfDiscourse+
        modeledByBMC : FirstOrderProposition –> BasicModellingConcepts
        modeledBySC : HigherOrderProposition –> SpecificationConcepts
        mappedToBMC : SpecificationConcepts –> BasicModellingConcepts
        mappedToSC : BasicModellingConcepts –> SpecificationConcepts

    }
def mappedToBMC {
        all bmc: BasicModellingConcepts, sc: SpecificationConcepts, fop:
FirstOrderProposition, hop: HigherOrderProposition | bmc in sc.mappedToBMC <–>
(fop.holds=hop.holds.holds) && (fop.modeledByBMC = bmc) && (hop.modeledBySC =
sc)
    }
def mappedToSC {
        all bmc: BasicModellingConcepts, sc: SpecificationConcepts, fop:
FirstOrderProposition, hop: HigherOrderProposition | sc in bmc.mappedToSC <–>
(fop.holds=hop.holds.holds) && (fop.modeledByBMC = bmc) && (hop.modeledBySC =
sc)
    }
}
```

Here we have declared and defined two relations: mappedToBMC and mappedToSC that allow to map SpecificationConcepts to BasicModellingConcepts and vice versa within the context of the same model element corresponding to some entity from UniverseOfDiscourse.

5.1.6. RM-ODP Standard and Constraints for its Formalization

As attentive readers may have noticed, in the previous section, while formally defining the context of a particular assertion that can be done about a model element with the aid of a basic modelling concept and of a specification concept, we avoided any reference to the model element itself.

It is not by chance that we followed this strategy. In fact, as it was explained in Section 4.1.1 of Chapter 4, a formalization of the RM-ODP standard by itself is a very important research result, because the standard defined such a formalization as one of its goals that, however, was never achieved. As it is clear from Section 5.1.4, the RM-ODP standard does not provide a self-sufficient metamodel of RM-ODP (the metamodel that would have enough information for its unambiguous interpretation). For example, as we noted in Section 5.1.4, the part of "Categorization of concepts" (RM-ODP 2-5) is vague and does not explain how a modeler should interpret the existing categorization of RM-ODP concepts. So we showed that a formalization of the RM-ODP conceptual framework is impossible without additional assumptions on the standard interpretation. Triune Continuum Paradigm, which was defined in Part I of the thesis, provides a sufficient set of interpretation constraints that makes possible the formalization of RM-ODP.

Chapter 1 (in particular, Section 1.3.2) shows that Triune Continuum Paradigm can provide theoretical foundations not only for the RM-ODP conceptual framework, but also for other object-oriented conceptual frameworks. And the RM-ODP case is presented as one of the possible examples in Figure 1.10. But, as it is clear from this figure, to perfectly conform to Triune Continuum Paradigm, RM-ODP conceptual structure needs to be slightly modified.

Appendix B presents all the necessary modifications that make RM-ODP completely conformant with the paradigm. From the appendix it becomes clear that the modifications practically do not change the semantics of existing RM-ODP concepts, instead they clarify and complete the semantics and make explicit the constraints (that are proposed by the paradigm) for the RM-ODP metamodel interpretation. Thus, it was in fact possible to formalize the currently existing in the standard conceptual structure by assuming these proposed interpretation constraints and by introducing no modifications to the standard. Of course, this formalization is not fully conformant with Triune Continuum Paradigm, but it is very close to the paradigm and it is fully conformant with the currently standardized RM-ODP conceptual framework.

So we decided to present here the formalization of the RM-ODP conceptual framework in its currently standardized state, without introducing the explicit modifications that are presented in Appendix B. This decision allows us to present

the result that can be practical for the RM-ODP ISO/ITU standard evolution even without any change of the currently standardized state of the framework.

In practice this means, for example, that in our formalization we avoided an explicit introduction of model elements defined in the paradigm, since their RM-ODP analogs are not explicitly defined in the standard.

For the readers who are interested to see how the explicit introduction of the model elements would influence our Alloy formalization we present here a piece of code that is necessary to introduce in the Alloy model for this case, here it is:

```
partition BasicInterpretationConcepts, Model: static ODP_Concepts
partition BasicModellingConcepts, SpecificationConcepts, ModelElement: static Model

modeledByME : Entity –> ModelElement
assertOnME : BasicModellingConcepts –> ModelElement
assertOnBMC : SpecificationConcepts –> BasicModellingConcepts
assertOnSC : SpecificationConcepts –> SpecificationConcepts
```

The first two lines of the code should be put in the state declaration of our Alloy model instead of the following line:

```
partition BasicInterpretationConcepts, BasicModellingConcepts, SpecificationConcepts : static ODP_Concepts
```

The last four lines of the code should be added in the state declaration of our Alloy model after the declaration of the following partitioning:

```
partition UniverseOfDiscourse, InterpretationPossibilities : static BasicInterpretationConcepts
partition Entity, Proposition: UniverseOfDiscourse
```

This would adapt the formalization to the explicit introduction of model elements; but as we explained in this section, we would like to present here the formalization of the currently standardized version of RM-ODP, that is why we will not include this piece of code in the final version of our Alloy model (presented in Appendix C).

Because of the same reason, we will not change (as it is proposed in Appendix B) the currently existing assignments of RM-ODP concepts to the concrete RM-ODP conceptual categories. Thus the concepts like “Internal action”, “Interaction”, “Interface”, etc. will be formalized in the basic modelling concepts category (where they can currently be found in the standard), regardless them being essentially the specific specification concepts for Triune Continuum Paradigm.

5.1.7. Semantic Difference of Model Elements and Basic Modelling Concepts

As we showed on Figure 5.1, an entity from the universe of discourse is modeled by a model element in the model. And as we explained in sections 5.1.1-5.1.3, both entity and model element correspond to the Russell's definition for "*individual*" [43], hence they are "*destitute of complexity*". That is, their only intrinsic meaning is to be "something" that can be qualified by means of propositions (in the case of entities) or by means of basic modelling and specification concepts (in the case of model elements).

For example, let's assume that a basic modelling concept such as the RM-ODP "Action" characterizes a model element. This means that the model element represents (in the model) an entity from the universe of discourse that (according to the "Action" definition in RM-ODP 2-8.3) has "something which happens" as its first-order proposition. And the proposition is modeled by the basic modelling concept.

If an entity has no propositions applied to it, then we will only have a model element that models this entity and we will not have any of basic modelling concepts characterizing it. For example, in this case we will not be able to assert that the model element is the RM-ODP "Object". Because this assertion would mean that the corresponding entity in the universe of discourse should have had the first-order proposition that is (according to the "Object" definition in RM-ODP 2-8.1) "something is a model of an entity". This contradicts to our assumption that the entity had no propositions applied to it.

Particularly, in the last example it is important to understand that "something is a model of an entity" is a proposition, and like all the other propositions it may be asserted or denied about some entity. And both the proposition and the entity for which it is asserted exist in the universe of discourse. So both will have their representations in a model: entity ("something") - as a model element, proposition for the entity ("a model of an entity") – as the corresponding basic modelling concept for the model element (as "Object"). It would be wrong to think that "Object" basic modelling concept does not model its corresponding proposition but instead directly models any entity from the universe of discourse, because this would essentially replace all the model elements by objects thus predicating them in accordance with the object term definition. This is unacceptable for several reasons:

- First, an entity from the universe of discourse doesn't have any meaning (these are only its propositions that give a concrete meaning to it). So it is in general logically consistent to represent it in the model by a meaningless model element rather then by the meaningful (RM-ODP 2-8.1) object. This is in fact

an exhibit of the classical predicate logic [11] nature being an uninterpreted logic. The logic doesn't impose any constraint on the subject of its interest or on the way it should be predicated. The use of this logic in combination with ontology, such as the one that RM-ODP presents, is justified by different authors (see for example [47]).

- Second, to consider model elements being "*destitute of complexity*" is consistent with Russell's theory of types [43].
- Third, by considering all the model elements as having the object's properties we exclude the possibility of having in the model those basic modelling concepts that contradict with these properties, and hence exclude the possibility of modeling the corresponding propositions from the universe of discourse. And the standard introduces many concepts that in general should not (and often cannot) be objects, such as "environment", "action", "location in space", "location in time" and others.
- Fourth, if RM-ODP 2-8.1 definition "*Object: A model of an entity*" has indeed assumed the general relation between the universe of discourse entities and their models (instead of assuming that "Object" models "something is a model of an entity" proposition from the universe of discourse), then "Object" would not be a basic modelling concept like all the others. So there would be no reason to introduce it in RM-ODP 2-8, rather it would have been introduced among the Interpretation Possibilities concepts (2-6.3, 6.4, 6.6) in RM-ODP 2-6.

All this refutes the acceptability of use of objects in place of model elements. Thus, as we explained, "to be a model of an entity" in the context of the "Object" definition (RM-ODP 2-8.1) is an ordinary proposition from the universe of discourse. This proposition, as any other proposition from the universe of discourse, has nothing in common with "modeled by" relation that exists between the universe of discourse and its model (specifically, between pairs: entities and model elements, first-order propositions and basic modelling concepts, higher-order propositions and specification concepts).

5.2. Formalization of Basic Modelling Concepts

In this section we will formally describe in Alloy the terms introduced in the basic modelling concepts category of RM-ODP. One of the important goals of this description is to define explicitly all the relations that are mentioned in ODP part 2 clause 8 concept definitions. Without their explicit definitions these relations are often at risk of being overlooked in different practical interpretations of RM-ODP. And sometimes ignoring one seemingly insignificant relation between the concepts leads to implicit assumption of another relation between them. The difference

between the assumed relation and the one that was really mentioned in the standard may cause a conceptual confusion in applications of RM-ODP.

5.2.1. Concepts Partitioning

As defined by RM-ODP, basic modelling concepts "*are concerned with existence and activity: the expression of what exists, where it is and what it does*". In other words with the aid of the basic modelling concepts we model the situation when: "something is", "something is somewhere" and "something acts" in the universe of discourse. Considering the concepts introduced by ODP in clause 8 we can map them to three basic categories that correspond to the quoted design goals for the basic modelling concepts.

- First, ODP has concepts that belong to the category responsible for modeling of constitution of the universe of discourse. This category models the propositions that in the universe of discourse answer the question: "what is something?" Here we will have two concepts: "Object" (RM-ODP 2-8.1) and "Environment" (RM-ODP 2-8.2). Let's correspondingly call this category as Constitution in our Alloy model.
- Second, ODP has concepts that are related with space and with time, which define the corresponding SpaceTime category. In this category we find "Location in time", "Location in space" and "Interaction point" concepts (RM-ODP 2-8.9, 8.10, 8.11). These concepts model the propositions that in the universe of discourse answer the question: "where/when is something?"
- The third conceptual category emerges automatically as soon as the first (Constitution) and the second (SpaceTime) categories are put within the same scope of consideration. It presents all the information from their mutual relation, explaining "how something is there and then". We will refer to this category as to Information, and inside we find concepts like "State" (RM-ODP 2-8.7), "Action" (RM-ODP 2-8.3), "Behaviour" (RM-ODP 2-8.6), etc.

As we see, this grouping of the basic modelling concepts into the three categories (according to their design goals defined by the RM-ODP standard) perfectly corresponds to the categorization of the basic modeling concepts in Triune Continuum Paradigm (see Chapter 2).

Now, for our Alloy formalization, we can partition BasicModellingConcepts (that was introduced in Section 5.1.2) in these three categories (Constitution, SpaceTime and Information):

```
// part of Alloy state declaration
partition Constitution, SpaceTime, Information : BasicModellingConcepts
```

5.2.2. Introduction of Constitution–related Concepts

Let us consider Constitution concepts. They are defined for modeling the constitution of the universe of discourse. Namely they model the first-order propositions that answer the question: "what is something?". That is, these propositions are responsible for entity modeling in the universe of discourse. In clauses 8.1 and 8.2 of part 2, RM-ODP defines two particular kinds of concepts used to model these propositions: object and its environment. Corresponding to RM-ODP 2-8.1 definition, having a model element that models a particular entity seen as "*a model of an entity*" in the universe of discourse, we can assert that the model element is "*object*". Environment is defined (RM-ODP 2-8.2) as a complement to an object. This means that in relation with a given model element (that is considered as object), all the other model elements (that model entities to which the entity-modeling propositions are applied) will be considered as environment.

Formally put, the model of the universe of discourse contains the model elements predicated with regard to their entity-modeling nature, which defines a universal set that is partitioned in two nonintersecting subsets: the object and its environment. The union of the subsets gives the model and the intersection gives nil. That is, the environment is the complement of its corresponding object and vice versa the object is the complement of the environment in the universal set, that is the model seen from Constitution–related perspective. Object and environment are always defined in relation with each other. Now it is obvious that this vision of the RM-ODP object and environment is in a perfect correspondence with the analogous vision defined in Triune Continuum Paradigm (see Section 1.2 of Chapter 1). Thus, expressing this in Alloy, we will have:

```
// part of Alloy state declaration
partition Object, Environment : static Constitution
environment (~object) : Object! -> Environment!
```

As ODP suggests, the content of the environment is defined by the scope of a concrete model. For example if a model (universal set) includes only one object and nothing else, then environment of the object is the empty set (nil). If a model includes only a set of objects and nothing else, then the environment of an object from the set includes all the other objects. If a model contains a set of objects and some "other kind" (different from object) of model of an entity from the universe of discourse, then the environment of an object from the set includes all the other objects and this entity model of the "other kind". In fact RM-ODP does not explicitly define any entity models that would be different from the object and its

environment. But it leaves a potential possibility for their existence by defining object being "*a model*" of an entity.

In Triune Continuum Paradigm the situation is different. There the "other kind" of constitution-related entity models is impossible because in the paradigm the declarative semantics of "object" is different. In the paradigm the semantics of "object" assume all the possible kinds of concepts related to the model constitution (see Section 1.2 of Chapter 1).

5.2.3. Introduction of Information–related Concepts

Part 2 clause 8.1 of RM-ODP defines an object to be characterized dually, by its behavior and by its state. These are the characteristics that provide us with the information on two aspects: what object does and how it is. The first will include behavior, action and other related concepts. The second information aspect will include state. Let us emphasize the difference between these two information aspects.

In the first part of information we can have essentially "dynamic" concepts. Those are the concepts that cannot be realized in a single instant in time, any of them would assume a time evolution at least for two different moments. For example, for the action concept that is defined as "*something which happens*" (RM-ODP 2-8.3), we cannot tell anything about the happening without having two instants: before and after the happening. With just one time instant we would not be able to define a change associated with the happening.

Contrary to this, the second information aspect provides us with the "static" information, such as state. Indeed, a state can be precisely determined for each particular instant in time; thus to define a state we don't need a time evolution in the model.

As we see from the presented arguments, this vision is in agreement with Triune Continuum Paradigm (see the corresponding part of Section 1.2 in Chapter 1).

The one-to-one mapping from time to state allows for the differentiation of state per instant in time when no changes of state between two different instants of time were registered. Since the backward mapping (from state to time) is in general a many-to-one mapping, this differentiation is a matter of choice. The choice of the differentiation gives us the possibility to model any kind of action, including those that do not influence object state and those that influence object state through their execution but return to the original state after an execution. The other choice (not to distinguish identical states that correspond to different time instants) can be considered with the same degree of success, if the requirements of a particular standard application will suggest its relevance.

So, now we may introduce the corresponding changes to our Alloy model. For the Information we should have its partitioning into StructuralInfo (which is also sometimes may be called as "static" or "state" information) and BehavioralInfo (which is sometimes called as "dynamic" information). State_[11] and Behavior will be the subsets of StructuralInfo and BehavioralInfo categories correspondingly.

```
// part of Alloy state declaration
partition StructuralInfo, BehavioralInfo : static Information
Behavior : BehavioralInfo
State_ : StructuralInfo
```

According to the definition RM-ODP 2-8.1 we may now associate Object with its State_ and Behavior. In addition to this we may also associate Environment and Constitution, which are also used for entity modeling with their corresponding State_ and Behavior. This does not contradict the standard and will help us to refer to the environment part of a model.

```
// part of Alloy state declaration
constitution_state: Constitution! –> State_
object_state: Object! –> State_
environment_state: Environment! –> State_
object_behavior: Object! –> Behavior!
environment_behavior: Environment! –> Behavior!
```

The absence of "!" for State_ in the object_state declaration corresponds to the fact that an object may have many states. The possibility of existence of a particular object state will be declared later after the time introduction.

5.2.4. Introduction of SpaceTime–related Concepts

Now let us consider the SpaceTime category. RM-ODP defines "Location in time" and "Location in space" as concepts concerned with relational positioning of things within a model. According to the definitions (RM-ODP 2-8.9, 2-8.10) the intervals contain correspondingly some specifically defined time or some specifically defined space within themselves. So we will need to have disjoint Space and Time subsets within the SpaceTime category. "Interaction point" is another concept representing location. Since the standard says (RM-ODP 2-8.11): "*at any given location in time, an interaction point is associated with a location in space*", the corresponding InteractionPoint set in Alloy will complete the partition of SpaceTime category. InteractionPoint should

[11] State_ is written with the "_" symbol only to distinguish the RM-ODP state concept from the "state" word that is reserved as Alloy special term.

have relations (space_location and time_location) to its corresponding LocationInSpace and LocationInTime concepts. It is also linked by definition with the interface concept that will be introduced in a little while.

```
// part of Alloy state declaration
partition InteractionPoint, Space, Time : static SpaceTime
LocationInSpace : Space
space_within_interval : LocationInSpace –> Space+
LocationInTime : Time
time_within_interval : LocationInTime –> Time+
space_location: InteractionPoint -> LocationInSpace!
time_location: InteractionPoint -> LocationInTime!
interface_at_interaction_point: InteractionPoint –> Interface
```

5.2.5. Introduction of Relations between Basic Modeling Concepts Subcategories

The introduction of space-time allows us to define all the "dynamic" information concepts (those from BehavioralInfo category), as well as to complete the definition of state-related information (from StructuralInfo) relating it to a particular instant of time. For the latter we have:

```
// part of Alloy state declaration
instant: Time –> Time!
state_existence: Time! –> State_!
state_location(~corresponding_state) : State_! –> Space!
```

Here state_location(~corresponding_state) relates a particular State_ with a particular Space, this will be used further for definition of "Location in space" ODP concept. For the completion of the ODP state concept declaration we need an Alloy invariant to say that there always exists a correspondence between a moment in time and an object state:

```
inv TimeDependance{
        all o: Object, t: Time  |  one t.instant –> one o.object_state
}
```

And the last thing that we need according to the state definition (RM-ODP 2-8.7) is a link from the state of an object to its potential activity:

```
// part of Alloy state declaration
potential_activity: State_ –> Activity+
```

As for the BehavioralInfo concepts in RM-ODP, Behavior can be considered as the most general of them. As defined (RM-ODP 2-8.6), it includes a collection of actions with a set of constraints on when they may occur, having action, activity and interface as degenerate cases of itself. So, Behavior is partitioned into Action and BehavioralConstraint. Producing an Action is the responsibility of the acting Object and in the case of an interaction, constraining is the responsibility of its Environment. In this case the BehavioralConstraint can be considered as a reaction of the environment of an object to the object action. As already mentioned, RM-ODP Action (RM-ODP 2-8.3) requires for its definition the introduction of two time instants, that are before and after the action. The Action can be of two different types: internal action and interaction, for their further definition we need to relate Action with its participants. Summarizing this paragraph we have:

```
// part of Alloy state declaration
partition Action, BehavioralConstraint: static Behavior
Interface: Behavior
Activity: Behavior
corresponding_constraint (~constrained_action) : Action -> BehavioralConstraint
constraining : Environment! –> BehavioralConstraint
partition InternalAction, Interaction : static Action
participant : Action –> Constitution
participating_object : Action –> Object!
instant_begin : Action –> Time!
instant_end : Action –> Time!
```

Now, after having declared all the necessary concepts we may proceed with their definitions in Alloy as it was explained in Section 4.2 of Chapter 4.

5.2.6. Definition of Constitution–related Concepts

There are no logical constraints that apply on the Constitution-related concepts in addition to those expressed for the concepts declaration. Thus no additional definition is required for this group of concepts.

5.2.7. Definition of Information–related Concepts

There are no logical constraints that apply on the concepts related with StructuralInfo (structural information) in addition to the constraints expressed for the concepts declaration. Thus no additional definition is required for this group of

concepts. We only need to define the concepts related with BehavioralInfo (behavioral information, which is also sometimes called as "dynamic" information or literally as Action-related concepts).

For the definition of the Action-related concepts we will need first to define an auxiliary concept of participant that was introduced together with the Action concept introduction:

```
def participant {
        all a: Action, b: Constitution | b in a.participant <-> (a.instant_begin.state_existence in
b.constitution_state) && (a.instant_end.state_existence in b.constitution_state)
}
```

That is to say, something from Constitution is defined as a participant of an Action if and only if the pre- and post- states of the Action are in the allowed states of the element from Constitution under consideration.

Now we can define Action:

```
def Action{
        all a: Action |  (a.instant_begin != a.instant_end) && (a.instant_begin.state_existence !=
a.instant_end.state_existence) && (a.participating_object in a.participant)
}
```

We notice two parts in this Action definition. The first is the state difference in the beginning of the action (a.instant_begin) and in the end of it (a.instant_end), which reflects RM-ODP 2-8.3 definition statement that something should happen to be an action. And the second part of the definition is the fact that there should be an object among action participants, this reflects the definition associating action with at least one object. To define InternalAction and Interaction we need to say that in the first case the environment of the participating object does not participate in the action, and in the second case it does participate:

```
def InternalAction {
        all a: InternalAction | a.participating_object in a.participant –>
a.participating_object.environment not in a.participant
}
def Interaction {
        all a: Interaction | a.participating_object in a.participant –>
a.participating_object.environment in a.participant
}
```

The definition of Behavior should link a set of actions with the corresponding constraints:

```
def Behavior {
	all b: Behavior |  ((b in Action) && ( some b.corresponding_constraint) && ( b.corresponding_constraint in Behavior)) || ((b in BehavioralConstraint) && ( some b.constrained_action) && ( b.constrained_action in Behavior))
}
```

As defined (RM-ODP 2-8.4), interface is "*an abstraction of the behavior of an object that consists of a subset of the interactions of that object together with a set of constraints on when they may occur*". So for the Alloy definition of the Interface set we need only to add to the Behavior definition a condition saying that all the Actions within an Interface are Interactions:

```
def Interface {
	all i: Interface | ((i in Interaction) && ( some i.corresponding_constraint) && ( i.corresponding_constraint in Interface)) || ((i in BehavioralConstraint) && ( some i.constrained_action) && ( i.constrained_action in Interface))
}
```

5.2.8. Definition of SpaceTime–related Concepts

According to the standard (RM-ODP 2-8.9, 2-8.10), definitions for LocationInSpace and LocationInTime should include the condition on possibility for an action to occur within the corresponding intervals. So we have:

```
def LocationInSpace {
	all ls: LocationInSpace | some a: Action | (a.instant_begin.state_existence.state_location in ls.space_within_interval) && (a.instant_end.state_existence.state_location in ls.space_within_interval)
}
def LocationInTime {
	all lt: LocationInTime | some a: Action | (a.instant_begin in lt.time_within_interval) && (a.instant_end in lt.time_within_interval)
}
```

And for the InteractionPoint we will first need to define its relation with a set of Interfaces as it was mentioned in RM-ODP 2-8.11:

```
def interface_at_interaction_point {
	all ip: InteractionPoint, i: Interface | (i in ip.interface_at_interaction_point) <-> ((i.instant_begin.state_existence.state_location in ip.space_location.space_within_interval) && (i.instant_end.state_existence.state_location in ip.space_location.space_within_interval) && (i.instant_begin in ip.time_location.time_within_interval) && (i.instant_end in ip.time_location.time_within_interval))
}
```

Which is to say that an interface_at_interaction_point relation for an InteractionPoint points to a set of Interfaces that exists within the LocationInSpace and LocationInTime intervals that are associated with the InteractionPoint. Hence we can define the concept of InteractionPoint itself. For its definition we only need to condition it with the existence of the corresponding LocationInSpace and LocationInTime elements, and associate it with a set of Interfaces:

```
def InteractionPoint {
      all ip: InteractionPoint | one ls: LocationInSpace | one lt: LocationInTime |
ip.space_location = ls && ip.time_location = lt && some ip.interface_at_interaction_point
}
```

5.3. Formalization of Specification Concepts

In Section 5.2 we have formally defined the basic modelling concepts of RM-ODP. Analogously, here we will formalize the specification concepts, which are defined in RM-ODP 2-9.

As already mentioned in Section 5.1.3, specification concepts introduce the means to be used by a modeler for modeling the higher-order propositions of the universe of discourse being modeled. As we explained in Section 5.1.1, the higher-order propositions apply to the first-order propositions about entities in the universe of discourse. Analogously, specification concepts are the statements about basic modelling concepts that, in their turn, describe the model elements within a model. In general within a model RM-ODP specification concepts contain information that answers the question: "What kind of <BMC> is the model element?", where instead of BMC we would put any of the basic modelling concepts characterizing the model element.

Having examined all the different specification concepts found in the standard, we can sort them in three different groups.

Concepts from the first group can be defined in their interrelations. Thus, these are the statements that describe the basic modelling concepts and that make sense being related to each other. Here we find the concepts like "type", "class", "instance", "template", etc.

The second group concepts are "composition" and "decomposition". They describe relations between the basic modelling concepts.

These two groups consist of the generic specification concepts; that is the specification concepts from these groups can be applied as descriptions for any of the basic modelling concepts. The third group is different. Here we find so-called specific specification concepts. These are the specification concepts that are limited in their applications to a concrete basic modelling concept. "Composite object",

"interface signature", "precondition", "postcondition" are examples of the concepts from the third group.

The specification concepts from the first two groups are suitable not only for describing the basic modelling concepts characterizing model elements but also for building more complex specification concepts by describing any of specification concepts themselves.

The specification concepts from the third group, because of their specific orientation to a particular basic modelling concept, cannot be used as concepts from the first two for the construction of the complex specification concepts.

For example, "type of type", "composition of templates", or "class of composite object" are valid complex specification concepts: the first two being generic and the third being specific. Of course, these specification concepts may have as many levels of complexity as it will be necessary; for instance: "type of composition of templates" or "type of composition of templates for types". But as we already explained in Section 5.1.3, if a specification concept of any level of complexity should be used in a model, then it will be applied to a basic modelling concept that, in its turn, will characterize a model element. And the statement constructed with the specification concept and the basic modelling concept, irrespectively of its complexity, will be a single assertion about the model element.

Thus we successfully demonstrated the correspondence of RM-ODP to the vision of generic and specific specification concepts that was defined in Triune Continuum Paradigm (see Section 1.3.2 from Chapter 1).

5.3.1. Type, Class, Instance and Related Concepts

Now, having explained the basic features of specification concepts, we can start building the corresponding part of our Alloy model. Let us begin with the first group, with the specification concepts that can be defined in mutual relations. In the Alloy model we can partition the SpecificationConcepts category (that was introduced in Section 5.1.3) in subcategories for different specification concepts:

```
// part of Alloy state declaration
partition Type, Class, Instance: SpecificationConcepts
```

Here we start by introducing three concepts of the SpecificationConcepts category, they are: Type, Class and Instance. In the standard the concept of Type defined as "*a predicate characterizing a collection of* <*X*>*s*" (RM-ODP 2-9.7). We need first to define the concept of <X> that is also mentioned in several of other definitions of RM-ODP 2-9. In the model <X> is a variable to which the higher-order proposition expressed by a specification concept should apply in order to get a concrete

assertion. Thus <X> is used to refer to a particular basic modelling concept. In other words, in a concrete model, any basic modelling concept that is characterized by some specification concept will be the <X> concept. We can define <X> within the basic modelling concepts part of the model, using the structure introduced in Section 5.1.5 as following:

```
state {…// part of Alloy state declaration
        X : BasicModellingConcepts
}
def X {
        all a: X | a.mappedToSC.mappedToBMC = a
}
```

Now we can return to the Type concept. Referring to a basic modelling concept that contributes to the collection characterized by the Type's predicate we are able to define Type in Alloy:

```
def Type {
        all t: Type | some x:X | x.mappedToSC = t
}
```

Class and Instance, as well as Template and template-related concepts, will be defined only in their relations with Type, without making an additional reference to BasicModellingConcepts. For example, Instance is defined as "*an <X> that satisfies the type*" (RM-ODP 2-9.18). So in Alloy model we need the corresponding relation with the Type concept and can introduce the definition:

```
state {…// part of Alloy state declaration
        satisfies_type(~valid_for): Instance+ -> Type!
}
def Instance {
        all a: Instance | some t: Type | a.satisfies_type = t
}
```

In fact, this definition of Instance in Alloy doesn't impose additional constraints in comparison with relations declared in the state part; nevertheless we prefer to show it for the purpose of better concept's presentation.

For Class that is defined as "*the set of all <X>s satisfying a type; the elements of the set are referred to as members of the class*" (RM-ODP 2-9.8) we will have the corresponding relations with Type and with Instance, and definition:

```
state {…// part of Alloy state declaration
        associated_type: Class! -> Type!
        member_of_class(~set_of): Instance+ -> Class!
```

```
}
def Class {
        all c: Class | some i: Instance | one t: Type | i.satisfies_type = t && i in c.set_of &&
i.member_of_class = c && c.associated_type = t
}
```

Now let us define the concepts of subtype/supertype (RM-ODP 2-9.9) and subclass/superclass (RM-ODP 2-9.10) as relations between types and classes correspondingly. As both of the standard definitions refer to type for making correspondence between the terms, we will also have references to Type in both of the Alloy definitions.

```
state {…// part of Alloy state declaration
        subtype(~supertype): Type -> Type
        subclass(~superclass): Class -> Class
}
def subtype {
        all t1: Type, t2: Type | t1 in t2.subtype <-> ( t1.valid_for.satisfies_type=t2)
}
def supertype{
        all t1: Type, t2: Type | t2 in t1.supertype <-> ( t1.valid_for.satisfies_type=t2)
}
def subclass {
        all c1: Class, c2: Class | c1 in c2.subclass <-> ( c1.associated_type in
c2.associated_type.subtype)
}
def superclass {
        all c1: Class, c2: Class | c2 in c1.superclass <-> ( c1.associated_type in
c2.associated_type.subtype)
}
```

5.3.2. Template and Related Concepts

The Template concept is defined (RM-ODP 2-9.11) by a reference to the Instantiation that is itself defined referring to Instance and Template (RM-ODP 2-9.13). So, introducing instantiation as a relation between Template and Instance, we will be able to define Template recursively:

```
state {…// part of Alloy state declaration
        specification (~instantiation): Instance -> Template!
}
def Template {
        all tpl: Template |  tpl.instantiation.specification = tpl
}
```

To continue with the definitions of Template and instantiation, we will need to relate them with the Type concept. As it follows from definitions RM-ODP 2-9.7, 2-9.11, template is just one of the possible (for a given type) specifications of the type. For example, "the sound of the fourth octave LA" and "the sound of 440 Hz" are two different descriptions that express the same type of sound and may have correspondingly two templates for the same type. So the instantiation of a Template gives as a result just a subset of all possible Instances of the Type associated with the Template. Other Instances of the Type are potential results of other instantiations of the Type Templates.

In RM-ODP instances produced as results of a template instantiation are called "instantiations of a template". That is, the standard uses the same word (instantiation) to refer to the process and to its result. In our Alloy model we will refer to any of the instances produced as results of a template instantiation as Instantiation (with the first "I" capitalized, to distinguish the term from the instantiation that expresses corresponding relation between a Template and an Instantiation). So the previous fragment of the Alloy model will change as follows:

```
state {…// part of Alloy state declaration
        Instantiation: Instance
        specification (~instantiation): Instantiation -> Template!
}
def Template {
        all tpl: Template |  tpl.instantiation.specification = tpl
}
```

To name explicitly the Type associated with a Template the standard introduces the concept of "Template type" that is defined as RM-ODP 2-9.19. So we will need to introduce the TemplateType concept that is a kind of Type and that, as it is described in RM-ODP 2-9.13, consists of the associated Template and of "*other necessary information*". Let us refer to this "*other necessary information*" that is needed for a template instantiation as the "instantiation rules", and let us introduce the corresponding InstantiationRules concept. So Template and InstantiationRules partition TemplateType; and TemplateType itself is a kind of Type. This means that we are able to finalize the Template concept definition relating it with the Type concept:

```
state {…// part of Alloy state declaration
        TemplateType: Type
        partition Template, InstantiationRules : TemplateType
        Instantiation: Instance
        specification (~instantiation): Instantiation -> Template!
}
def Template {
        all tpl: Template |  tpl.instantiation.specification = tpl
```

```
}
```

The concept of "Template class" (RM-ODP 2-9.20), which refers to the class associated with a template, is introduced in relation with "Template type". So we have:

```
state {…// part of Alloy state declaration
        TemplateClass: Class
        associated_template_type: TemplateClass! -> TemplateType!
        member_of_template_class(~set_of_instantiations): Instantiation+ -> TemplateClass!
}
```

Where associated_template_type is essentially just the associated_type relation between TemplateClass and TemplateType, as well as member_of_template_class(~set_of_instantiations) is just a member_of_class(~set_of) relation between Instantiation and TemplateClass:

```
def associated_template_type {
        all c: Class, t: Type, tc: TemplateClass, tt: TemplateType | ((c.associated_type = t) &&
(tc.associated_template_type = tt)) <-> ((tt = t) && (tc = c))
}
def member_of_template_class {
        all ii: Instantiation, tc: TemplateClass, i: Instance, c: Class | ((i.member_of_class = c) &&
(ii.member_of_template_class = tc)) <-> ((i = ii) && (tc = c))
}
```

The standard definition RM-ODP 2-9.21 introduces a "Derived class / base class" relationship between template classes. It is defined with the aid of "incremental modification" relationship between the corresponding templates:

```
state {…// part of Alloy state declaration
        derived_class(~base_class): TemplateClass -> TemplateClass
        incremental_modification: Template -> Template
}
def derived_class {
        all tc1: TemplateClass, tc2: TemplateClass | tc1.derived_class = tc2 <->
tc1.set_of_instantiations.specification.incremental_modification.instantiation.member_of_templ
ate_class = tc2
}
```

As it is mentioned in RM-ODP 2-9.21: "*The criteria for considering an arbitrary change to be an incremental modification would depend on metrics and conventions outside of this Recommendation | International Standard*". This means that the formal definition of the incremental_modification concept is beyond the scope of the thesis, but it can be done

as a complement to our Alloy model within the scope of a particular application of RM-ODP.

To conclude with the template-related concepts, the standard partitions the Instantiation into two different kinds: Creation (RM-ODP 2-9.15) and Introduction (RM-ODP 2-9.16). These are two kinds of Instantiation that differ from each other by the way the basic modelling concept that they characterize appears in the model. The first is the result of "*instantiating an <X>, when it is achieved by an action of objects in the model*", while the second of "*instantiating an <X> when it is not achieved by an action of objects in the model*".

The proposition: "*Every action of interest for modelling purposes is associated with at least one object*" is valid for every action within the model as it is defined in RM-ODP 2-8.3. This means that an Introduction should be achieved by means of something else than an action from within the model. Since within the model action is the only concept that is applicable for the expression of model changes, the Introduction should be introduced from outside of the model. So we cannot define the Introduction concept as it should be referred to something that is beyond of the model scope. Because of this, the definition of Creation is not necessary; since, if staying within the scope of the model, we cannot have any alternative to it. So the most that we shall do for the relational definition of Creation and Introduction is to declare the corresponding partition of the Instantiation:

```
state {…// part of Alloy state declaration
        partition Creation, Introduction : Instantiation
}
```

As for the functional role that the two concepts should play in the model, which is to make a new basic modelling concept to appear within the model, we may define the corresponding Alloy operation:

```
op Instantiate (i: Instantiation, bmc: BasicModellingConcepts, new_bmc:
BasicModellingConcepts', new_i: Instantiation', x:X'!, ix: Instantiation'!) {
        x not in bmc
        ix not in i
        new_bmc = bmc + x
        new_i = i + ix
        x.mappedToSC' = ix
        ix.mappedToBMC' = x
}
```

The operation Instantiate changes the set of BasicModellingConcepts and the set of Instantiations by adding an element to each of the sets and mapping the added two elements to each other. Analogously we will define the concept of Deletion that is

defined in RM-ODP 2-9.17 as "*the action of destroying an instantiated* <*X*>". Here we will need an operation that removes the pair of a basic modelling concept mapped with an instantiation from their corresponding BasicModellingConcepts and Instantiations sets:

```
op Deletion (i: Instantiation, bmc: BasicModellingConcepts, new_bmc: BasicModellingConcepts', new_i: Instantiation', x:X!, ix: Instantiation!) {
        x in bmc
        ix in i
        x.mappedToSC = ix
        ix.mappedToBMC = x
        new_bmc = bmc - x
        new_i = i - ix
        x not in new_bmc
        ix not in new_i
}
```

5.3.3. Refinement

Refinement is defined (RM-ODP 2-9.5) as a relation between two specifications, where the second specification is obtained by adding details to the first one. According to this definition, in Alloy we will have a refinement relation between two sets of SpecificationConcepts, and the second will contain the first one as a subset, as well as some other non-empty set of SpecificationConcepts, which will correspond to the details added in the process of refinement.

```
state {…// part of Alloy state declaration
        refinement: SpecificationConcepts -> SpecificationConcepts
}

def refinement {
        all spec1: SpecificationConcepts, spec2: SpecificationConcepts | some d: SpecificationConcepts | (spec1.refinement = spec2) -> (spec1+d=spec2)
}
```

5.3.4. Composition, Decomposition

The previous chapter concludes the formalization of the important group of specification concepts, the concepts that can be defined in mutual relations. Now let us introduce the concepts of "Composition" and "Decomposition". In the standard "Composition (of objects)" is defined (RM-ODP 2-9.1.a) as "*a combination of two or more objects yielding a new object at a different level of abstraction.*" Analogously in the definition RM-ODP 2-9.1.b the standard introduces "Composition (of behaviours)".

But in fact, "Composition" may be applied to any of the basic modelling concepts that have been discussed in Section 5.2. So, for the definition of "Composition" let us not refer just to the two of basic modelling concepts (object and behaviour); instead let us consider "Composition" as a generic specification concept, which can characterize any kind of the basic modelling concepts. For this we will use the <X> concept that was introduced earlier to designate the basic modelling concept referred by a specification concept. Then for the "Composition" concept definition accordingly to the RM-ODP 2-9.1 we will have:

```
state {…// part of Alloy state declaration
        partition Type, Class, Instance, Composition: SpecificationConcepts
}
def Composition {
        all c: Composition | one a1:X | some a2:X | one a3:X   | a1+a2=a3 <-> (c =
a1.mappedToSC) && (c = a2.mappedToSC)
}
```

Which is to say that if two or more basic modelling concepts of some kind form another basic modelling concept of the same kind, then each of these two or more basic modelling concepts contributes to the Composition (composes the other basic modelling concept). Analogously for "Decomposition" (RM-ODP 2-9.3) we will have:

```
state {…// part of Alloy state declaration
        partition Type, Class, Instance, Composition, Decomposition: SpecificationConcepts
}
def Decomposition {
        all c: Decomposition | one a1:X | some a2:X | one a3:X   | a1+a2=a3 <-> c =
a3.mappedToSC
}
```

5.3.5. Specific Specification Concepts

Specific specification concepts are characteristics representing particular basic modelling concepts. This is the last category of specification concepts that we need to formalize. Sometimes they are just the results of a generic specification concept application to a particular basic modelling concept (such as "composite object" for example); in other cases they characterize unique intrinsic features of a particular basic modelling concept (such as "precondition" and "postcondition"). For the former, we have already introduced all the necessary information in our Alloy models, these concepts will be presented by the combinations of already existing in the model logical constraints that correspond to the pair of appropriate basic

modelling and specification concepts. For the latter we need to introduce the SpecificSpecConcepts category in the SpecificationConcepts partition:

```
state {…// part of Alloy state declaration
partition Type, Class, Instance, Composition, Decomposition, SpecificSpecConcepts:
SpecificationConcepts
}
```

But in both of the cases, specific specification concepts definitions assume an explicit reference to a concrete kind of basic modelling concept, which distinguishes them from all the previously introduced specification concepts. The latter, which we also call generic specification concepts, can characterize any kind of basic modelling concepts and also can be used for a construction of complex specification concepts being applied as characteristics of any of specification concepts themselves.

Let us begin with the Alloy definitions of specific specification concepts.

"Composite object" is defined (RM-ODP 2-9.2) as "*an object expressed as a composition*". So for its formal definition we have to link the terms of Object from BasicModellingConcepts and of Decomposition from SpecificationConcepts that correspond to this definition:

```
state {…// part of Alloy state declaration
        composite_object: Decomposition -> Object!
}
def composite_object {
        all d: Decomposition | one o: Object | (d.composite_object = o) <-> ((d.mappedToBMC =
o) && (o.mappedToSC = d))
}
```

The concept of "Behavioural compatibility" (RM-ODP 2-9.2) introduces the corresponding relation that can apply either to objects or to other specification concepts that would apply to objects (such as to object templates or to object template types). The definition says that: "*An object is behaviourally compatible with a second object with respect to a set of criteria (see notes) if the first object can replace the second object without the environment being able to notice the difference in the objects' behaviour on the basis of the set of criteria*". As we see it, it essentially depends on the set of criteria that is, as mentioned in the notes associated with the definition, defined by the conditions of a particular application of the standard reference model. Thus "Behavioural compatibility" cannot be expressed within the frames of our model; nevertheless, it may be defined within the scope of a particular standard application.

The concept of "Trace" is defined as: "*A record of an object's interactions, from its initial state to some other state. A trace of an object is thus a finite sequence of interactions. The behaviour uniquely determines the set of all possible traces, but not vice versa. A trace contains no record of an*

object's internal actions" (RM-ODP 2-9.6). At the same time, as it is defined in RM-ODP 2-8.1: "*An object is encapsulated, i.e. any change in its state can only occur as a result of an internal action or as a result of an interaction*". So since a state of an object can also be changed with its internal actions (that should not be contained in object's traces), there is no way to build the sequencing of object's interactions by anchoring them to the object states. And because no other indications were given with regard to the sequencing, we may say that according to the definition RM-ODP 2-9.6, trace is some arbitrary set of object's interactions, which was already formally defined in the Section 5.2.7.

The "Interface signature" concept is defined as "*the set of action templates associated with the interactions of an interface*" (RM-ODP 2-9.12). Thus we define it as a mapping between an interface and a set of templates, such that there exists set of interactions belonging to this interface and having the templates from the set as their corresponding specification concepts, while the templates have the interactions as their basic modelling concepts:

```
state {…// part of Alloy state declaration
        interface_signature: Template -> Interface!
}
def interface_signature{
        all t: Template, i: Interface, a: Interaction | (t.interface_signature = i) <-> a in i && t in
a.mappedToSC && a in t.mappedToBMC
}
```

The standard defines Role as "*identifier for a behaviour, which may appear as a parameter in a template for a composite object, and which is associated with one of the component objects of the composite object*" (RM-ODP 2-9.14). Thus, referring to the definitions that we made in Section 5.3.2, we see that Role is a kind of Type applied to Behavior that may happen to be a part of a TemplateType, and even more concretely, a part of Template. The references to Template in the definition RM-ODP 2-9.14 help us to understand this categorization of Role concept within specification concepts, but do not impose any restrictions for the concept definition. Indeed, according to the definition, Role "*may*" be used within the Template context, which doesn't prevent it from being used in some other context and from not being used in the Template context either. So in correspondence with the standard definition, we can define Role as a Type applied on a Behavior:

```
state {…// part of Alloy state declaration
        Role : Type
}
def Role{
        all id: Role | one b:Behavior | b.mappedToSC = id
}
```

The concept of Invariant is defined by RM-ODP 2-9.22 as "*a predicate that a specification requires to be true for the entire life time of a set of objects*". This qualifies it as the result of Type specification concept application on a set of Objects as basic modelling concept:

```
state {…// part of Alloy state declaration
        Invariant : Type
}
def Invariant{
        all i: Invariant | some o: Object | o.mappedToSC = i
}
```

The two remaining concepts are Precondition and Postcondition (RM-ODP 2-9.23,24). These are the predicates that are linked with the Action basic modelling concept and are required to be true before an Action and immediately after an Action. Since the predicates are required to be true in the single instants in time, with the structure of Information introduced in Section 5.2.3, we may conclude that the predicates apply to the State_ information before and after an action. Thus we can define:

```
state {…// part of Alloy state declaration
        Precondition, Postcondition : Type
}
def Precondition{
        all prec: Precondition | some a : Action | one s : State_ | a.mappedToSC = prec &&
a.instant_begin.state_existence = s
}
def Postcondition{
        all postc: Postcondition |some a : Action | one s : State_ | a.mappedToSC = postc &&
a.instant_end.state_existence = s
}
```

This concludes the formal definition of RM-ODP specification concepts category, and at the same time concludes the Alloy formalization of the RM-ODP standard. The introduced and explained code of this formalization can be found in Appendix C of this thesis.

6. APPLICATION OF THE RM-ODP-BASED FORMAL ONTOLOGY

This chapter is dedicated to application of the RM-ODP-based formal ontology that was defined and explained in Chapter 5 as a concrete implementation of Triune Continuum Paradigm. The reader will:

- see the advantages that the formalization provides to modelers;
- see how the formalization can be used in practice;
- become acquainted with a UML-based Systemic Notation that allows for representation of the formalization by means of drawings that are familiar to readers who use modern object-oriented modeling style in their practice;
- see the ontological engineering analysis that will explain how the framework applications should be positioned with regard to applications of other different ontologies.

6.1. Framework Application Principles

In Chapter 5 we presented the RM-ODP example of a concrete implementation of Triune Continuum Paradigm. This implementation thus presented a concrete RM-ODP-based formal ontology that is applicable in the scope of general system modeling (defined in Chapter 2). The work on definition of the formal ontology was stimulated by several practical needs:

1. The need to have formal definitions for the terminology used for modeling.
   - This was needed to eliminate conflicts in modeling terms interpretations by different modelers, which would allow agreeing on a common vocabulary to be used within the modelers' community.
2. The need to have formal definitions for relations between universe of discourse (subject of interest for modeling) and its models, as well as for relations between different conceptual categories used to construct models.
   - This was necessary to provide a logically consistent method of application for the modeling terminology
3. The need to ensure internal consistency and completeness in specifications of solutions for practical modeling problems
   - The internal consistency and completeness were always essential: internally inconsistent specification potentially causes malfunctioning

implementation, and incomplete specification causes only partially functioning implementation.

Resolutions for these needs were successfully provided by our RM-ODP-based formal ontology. The resolutions of the first two needs were presented in Chapter 5 while defining the ontology. Now we will show how the defined ontology resolves the third need maintaining rigorous relations between concepts in system specifications.

Let us consider a model presented with the aid of a traditional modeling technique (UML [40]), analyze its potential problems and see how the same model can be represented with the aid of the RM-ODP-based formal ontology resolving these problems.

Figure 6.1 contains two UML diagrams[12]: the class diagram on the left side of the figure and its corresponding object diagram on the right side of the figure.

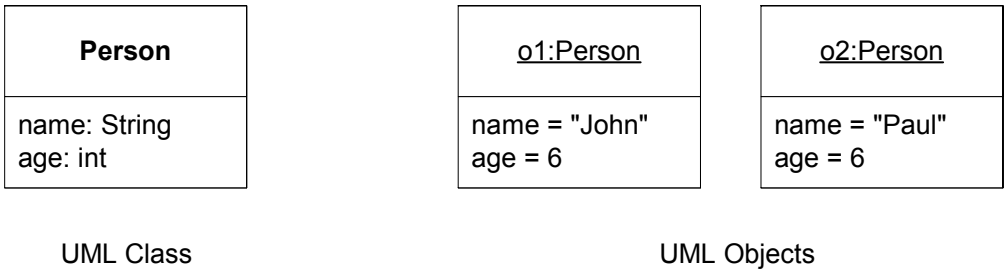

Figure 6.1: UML model example: UML Class and corresponding UML Objects.

The class diagram presents a UML class "Person" that has an attribute "name" of type "String" and an attribute "age" of type "int". The two corresponding UML objects are called "o1" and "o2"; they have "name" attributes as "John" and "Paul", and "age" attribute for both of them is "6". In RM-ODP this corresponds to a scenario where "Person" is an RM-ODP type and "o1", "o2" are two instances of this type.

We can notice that there are different possibilities for interpretation of the conceptual structure that is introduced on Figure 6.1. Particularly, we don't know, whether "Person" type is composed of "Name" and "Age" types, or "name" and "age" attributes are just references to types with lifecycles that are independent from the lifecycle of "Person". In Figure 6.2 the first of the two mentioned situations is realized when relation "a)" is valid and relation "b)" is not valid. And in the case when "b)" is valid and "a)" is not valid on the figure, we have the situation when

[12] This example has a direct conceptual correspondence with an example from [40] (see Figure 3-3 from Chapter 3.12 in [40]).

type "Person" under attribute "age" has an internal information referring to type "Age" that is external to and independent from "Person".

We see on the class diagram from Figure 6.2 that two cases treat relations between lifecycles of the types differently.

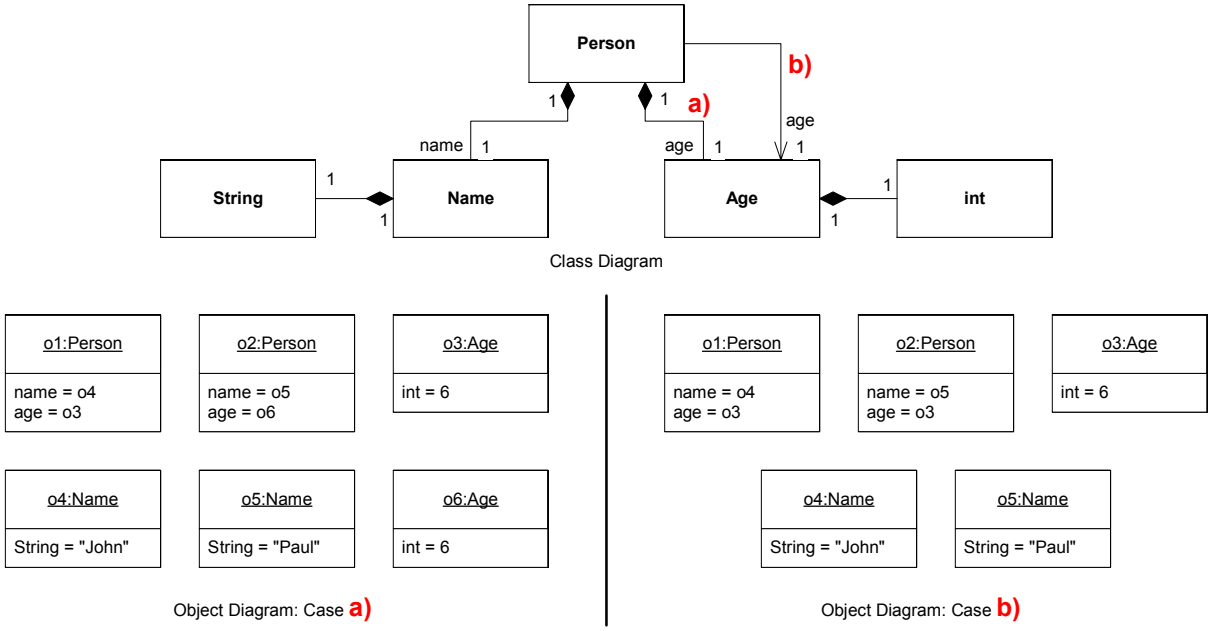

Figure 6.2: UML model example: UML Class and corresponding UML Objects.

Another difference becomes clear when we examine the object diagrams that correspond to the two cases (see the figure). In the case of "b)" we have only a one-way navigable association; it means that we don't have any information on the other way relation that is the relation from "Age" to "Person". Thus the situation where different instances of "Person" reference the same instance of "Age" may exist in the case of "b)". So, on the figure we can have "o1" and "o2" referencing the same "o3" under their corresponding attributes "age". This situation is certainly impossible in the case of "a)": here any instance of "Age" is reserved as a component object for only one instance of "Person".

So, we see that drawing diagrams on the level of details that is presented in Figure 6.1 is not sufficient. This level of details is incomplete and in practice inconsistency in the diagrams interpretations (which, as we showed, is quite probable here) will sooner or later lead to an inconsistency in system specifications and thus to potential implementation faults. And it will be quite difficult after that to find the cause of the inconsistency.

From the other side, we see that drawing diagrams on the level of details that is presented on Figure 6.2 requires a painstaking analysis of relations that sometimes

are not interesting for the modeling problem. To be more precise, it is always important that all relations between concepts constitute a consistent and complete solution within the scope of the modeling problem; but some relations can have a conceptual weight that is relatively small (or even negligible) within a solution in comparison with the weight of other relations from the same scope. Let us illustrate this situation on examples of modeling problems for Figure 6.2:

- if we want to model a health insurance business, we should chose "b)" case from Figure 6.2, because we would need to have many people of the same age and would need to define any of different insurance profiles corresponding to one of different ages. So, here neither it's reasonable to associate each "Age" instance to only one "Person", nor it is reasonable to make the "Age" type lifecycle subordinate to the "Person" type lifecycle (usually clients and their ages change more often then age-based insurance profiles);
- if we want to model family relations between people within a family, we should choose "a)" case from Figure 6.2, because here people cannot share the same instance of "Age". Even in the case of twins who born at the same date, one of them may die while the other will continue to live and thus to age. The last fact is the reason to manage lifecycle of "Age" as a subordinate to the lifecycle of "Person".
- now, if we want to model a family-based profile for the health insurance business (for example to have an encouraging policy for keeping the complete family as clients of the same insurance company), then there is no obvious preference in favor of any of the two cases ("a)" or "b)"). More then that, neither age profile nor personal profile is the primary concern in this problem. Personal perspective and age perspective although necessary, but not very important here; it is the family as a group that matters here the most.

If we consider these three cases as three hypercomplex systems [31], [33], [34] ("Family", "Personal Insurance" and "Family Insurance") in their relations, we see that the last one can be considered as a system that emerges from the first two if the two are considered in the same context of evolution. Thus the third system has a higher level of hierarchy (see Figure 6.3).

This example shows that modeling problems for a system that has several levels of conceptual hierarchy are concerned with conceptual relations corresponding to the highest level of hierarchy. And it is these relations that need to be defined to construct solutions for the identified problems. That's why all the relations for systems from the lower levels of hierarchy are not interesting for these problems. However even not interesting relations are important, because a solution is only possible under assumption that all the relations in the lower levels of hierarchy have been previously understood and correctly specified.

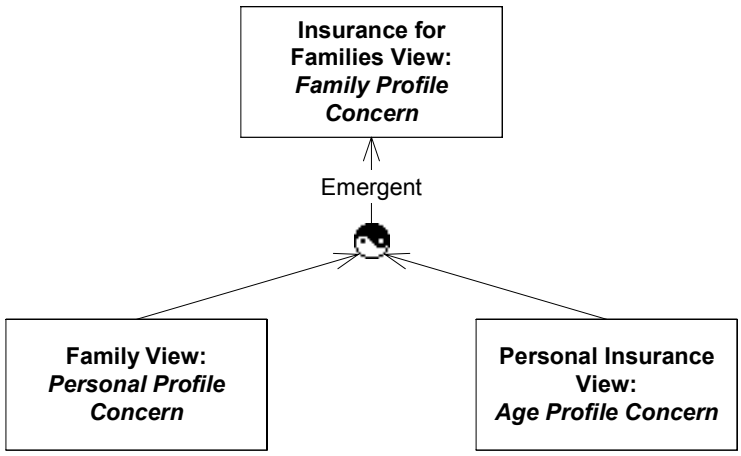

Figure 6.3: A new concern emerging from two different concerns considered in the same scope.

Summarizing, we see that modelers are confronted with a dilemma: from one side the Figure 6.1 level of specification details is not sufficient, from the other side the Figure 6.2 level of specification details is too profound containing details that are being important, however irrelevant for the direct problem concerns and number of which exponentially increases with the growth of problem complexity[13].

A way to solve this dilemma is to provide modelers with means that would automatically ensure internal consistency and completeness of specifications for all the different degrees of problem complexity allowing modelers to concentrate on essentials of their modeling problems. The RM-ODP-based formal ontology for general system modeling that was defined in Chapter 5 as an example of Triune Continuum Paradigm implementation provides such means. For a particular modeling problem it makes explicit the context of definition for any concept from within the problem scope. This anticipates a potential lack of information thus preventing interpretation conflicts for the concepts specification.

Let us illustrate how one of the discussed situations can be represented using our RM-ODP-based formal ontology. Figure 6.4 uses the ontology to present the model corresponding to the Figure 6.2 "a)" case.

[13] Problem complexity associated to a problem and measured as a number can be identified as directly proportional to the number of systems that need to be specified on the lowest level of hierarchy of conceptual organization of systems involved in the problem. Thus from one side problem complexity has exponential dependency on the total number of hierarchical levels considered in systems conceptual organization; and from the other side amount of specification details can be identified as directly proportional to the number of systems on the lowest level and consequently as directly proportional to the problem complexity.

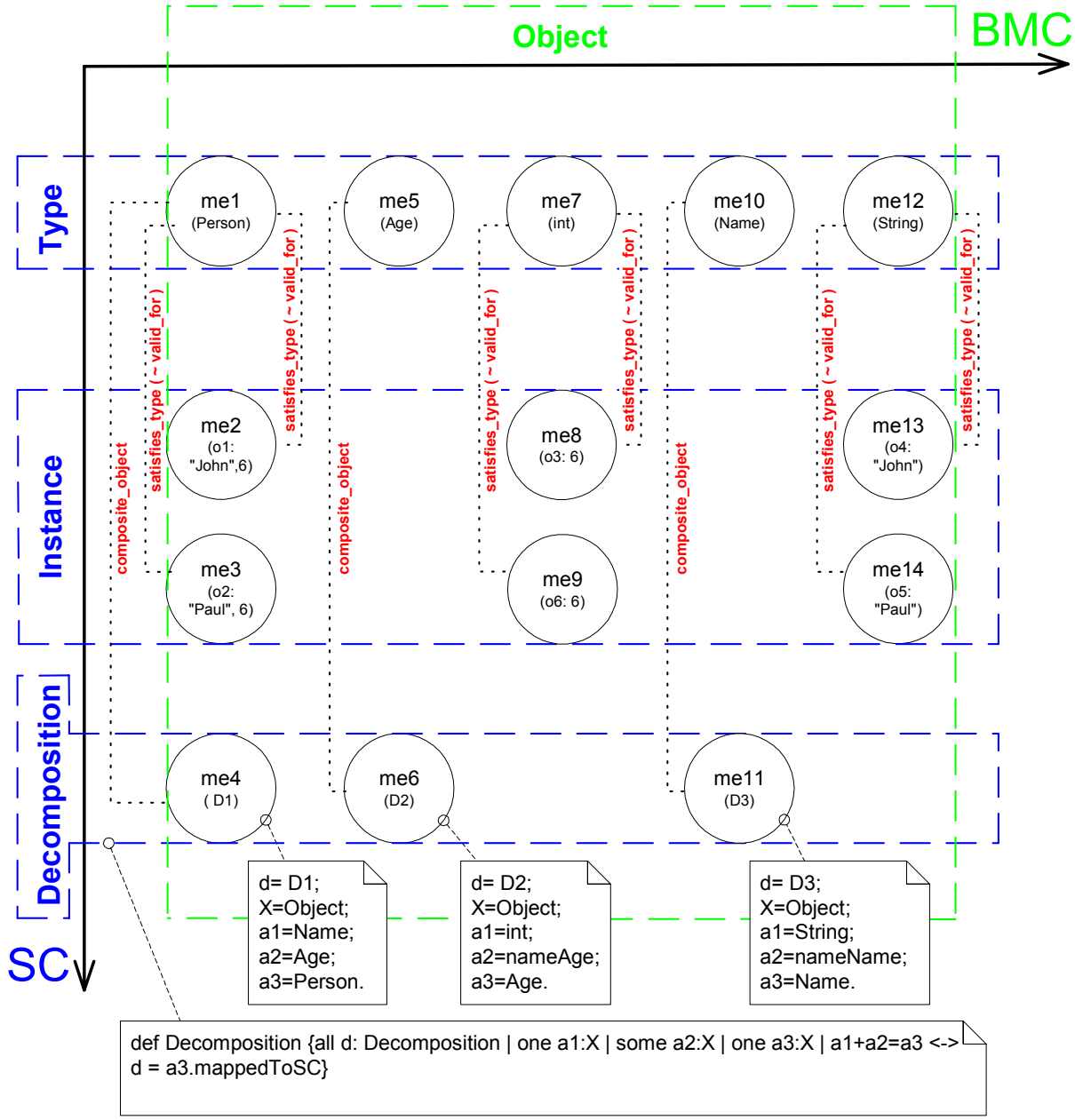

Figure 6.4: Example of a model done with the aid of RM-ODP-based formal ontology.

As it was explained when defining the ontology in Chapter 5, we build models using three principal means:

- Model Elements - anything in our models that we want to reason about. A Model Element is considered being destitute of complexity (that is destitute of propositions that can qualify its meaning).

- Basic Modelling Concepts – the corresponding collection of RM-ODP concepts which presents first-order propositions that can qualify Model Elements.
- Specification Concepts – the corresponding collection of RM-ODP concepts which presents higher-order propositions that can be applied on Basic Modelling Concepts.

Using this approach to model our scenario of interest we shall have 14 Model Elements (ME). Five of them should be qualified as "Type of object", other six of them – as "Instance of object" (relating the instances of objects to their corresponding types of objects) and the other three – as "Decomposition of object". These numbers are the same as the respective numbers from Figure 6.2 (case "a)"). Thus each of the 14 Model Elements should have "Object" Basic Modelling Concept (BMC) as its first-order proposition. The mentioned five of MEs should have "Type" Specification Concept (SC) as their higher-order proposition, the mentioned six MEs – "Instance" SC, and the mentioned three – "Decomposition" SC.

We present this on Figure 6.4 using BMCs and SCs as values for coordinates in the two-dimensional reference frame that is defined by the two conceptual categories. In our case, as it is shown on the horizontal axis, "Object" is the only relevant concept in BMC; it refers to all the 14 MEs shown as circles. "Type", "Instance" and "Decomposition" are the three relevant concepts in SC; they are shown on the vertical axis referring to their corresponding MEs qualified with "Object" as BMC.

As we see on Figure 6.4, all the relations between Model Elements are explicit. For example, {satisfies_type(~valid_for): Instance+ –> Type!} formal relations between instances and their type are shown for all the instances of objects and for the types of objects that have their corresponding instances of objects. Each of the decompositions is formally defined - see the formula at the bottom of Figure 6.4: the comments attached to the MEs qualified as decompositions of objects contain concrete values of the formula parameters for each of the decompositions. For example, for ME4 that is decomposition of object we define:

- an arbitrary name for the decomposition as "D1";
- that parameter "X" in the formal definition is equal to "Object" (which means that decomposition should be applied on the "Object" BMC);
- that "a1" and "a2" parameters are equal to "Name" and to "Age" (which means that the components of decomposed object are objects called "Age" and "Name");
- that "a3" parameter is equal to "Person" (which means that the composite object is called "Person"), this automatically creates the formal relation {composite_object: Decomposition –> Object!} from ME4 to ME1.

Now we are able to say that this decomposition is formally defined for ME1 (type of object called "Person") to be decomposed in ME5 (type of object called "Age") and ME10 (type of object called "Name"). So, this decomposition unambiguously defines relations between the mentioned Model Elements and is itself a Model Element.

This semantics are formal, which means that they can (and we assume that to facilitate the modeling process they should) be processed by a computer program that would automatically ensure internal consistency (absence of conflicts in relations between Model Elements) and completeness (existence of all the essential Model Elements and relations) in models.

In Chapter 5 we presented the formalization done in Alloy [24] that can be treated with the Alloy Constraint Analyzer utility. Currently models built with the aid of the RM-ODP-based formal ontology can be checked with this utility. A translation of the formal ontology from Alloy to other computer languages should be done to expand its practical application area.

6.2. Systemic Notation for the Formal Ontology Applications

Applying the formal ontology would probably be easier and thus potentially more acceptable by modelers if a modeler would not need to operate with the formal logical constructs presented in textual form, but rather would have a set of graphical modeling artifacts (such as the one of UML [40]) available for the modeling.

The development of such graphical notation requires a definition of the necessary graphical artifacts and a thorough mapping between the artifacts set and the set of formal logical constructs that was introduced in the Chapter 5. This development is beyond the scope of the thesis; however we think that it is relevant here to make a brief sketch of the notation that would be suitable for representation of the defined formal ontology.

6.2.1. General Overview of Requirements

We can distinguish *three main sets of requirements* for definition of a graphical notation that would be suitable for representing system models done with the aid of the RM-ODP-based formal ontology.

The *first set of requirements* originates from the General System Theory research. There it is shown (for example in [32]) that the level of any methodology should correspond to the level of problems within the discipline where the methodology is employed. Thus we need to define a notation that would be able to graphically represent systems as they are defined by L. v. Bertalanffy in his General System

Theory [3]. The theory defines system as a set of elements standing in interrelations that is considered as a whole having its emergent properties. Thus the notation should support the basic systemic principles that emerge out of this definition. Particularly it should:

- support multiple viewpoints on a represented subject:
  - → as on the system with emergent properties (both internal and external views),
  - → as on the element with its relations contributing to the system of a higher hierarchical level (subsystem seen from its own perspective and from the perspectives of other subsystems participating in relations with it)

The *second set of requirements* relevant for the notation originates from the RM-ODP-based formal ontology itself. The ontological conceptual structure needs to have a proper graphical representation. Thus the notation should be able to reflect the basic principles defined in Chapter 5 for concepts employed for modeling (Basic Modelling Concepts and Specification Concepts). Particularly for the Basic Modelling Concepts it should:

- properly present Constitution of models – objects and their environments;
- properly present Information about objects - static and dynamic information about objects and mutual dependency of the two informational aspects.

For the Specification Concepts it should explicitly present the nature of relations that the concepts introduce in models.

The *third set of requirements* comes from the modern industrial and research practices of the use of notations for object-oriented modeling. UML [40], proposed by OMG, has emerged as a unifying solution out of many different notations that had been employed through the history of development of object-oriented modeling. It is a very popular notation used for system modeling in our days, so currently it can be considered as a de facto standard. Thus our notation should preferably be as close as possible to UML, - this would facilitate its acceptance by the modern modelers community.

Keeping in mind the need to stay close to UML, we decided that the graphical artifacts in our notation will be the UML artifacts. However the way to use the artifacts in the notation will be different from the traditional UML way when it will be necessary due to other mentioned requirements. Let us further refer to our emerging notation as "Systemic Notation".

6.2.2. Invariant, Static, Dynamic Information Representation

As we explained while exploring requirements for Systemic Notation, we need to properly present Information about objects, particularly (1) static and (2) dynamic

information about them and (3) mutual dependency of the two informational aspects. RM-ODP in the clause 6.1 of part 3 [21] introduces three schemas to represent these three informational aspects[14]:

"*invariant schema: a set of predicates on one or more information objects that must always be true. The predicates constraint the possible states and state changes of the objects to which they apply.*" [clause 3-6.1.1]

"*static schema: a specification of the state of one or more information object at some point in time, subject to the constraints of any applicable invariant schemata.*" [clause 3-6.1.2]

"*dynamic schema: a specification of the allowable state changes of one or more information object, subject to the constraints of any applicable invariant schemata.*" [clause 3-6.1.3]

The invariant schema represents mutual dependency of the two informational aspects: mapping between the static (structural or state) information and the dynamic (behavioral) information. This property is explained by the object nature, exhibiting dually its state and its behavior. By representing both static and dynamic information in the invariant, the developer can make a more precise model. In particular, he/she can specify in what context things exist or are referenced. Note that concept of "being always true" (present in the definition of the invariant schema) has an implicit reference to a context. An invariant is always true in the context in which it is defined. Such context is typically the lifetime of an information object (as said in [21] clause 2-9.22). This invariant property will be shown on an example later in this chapter.

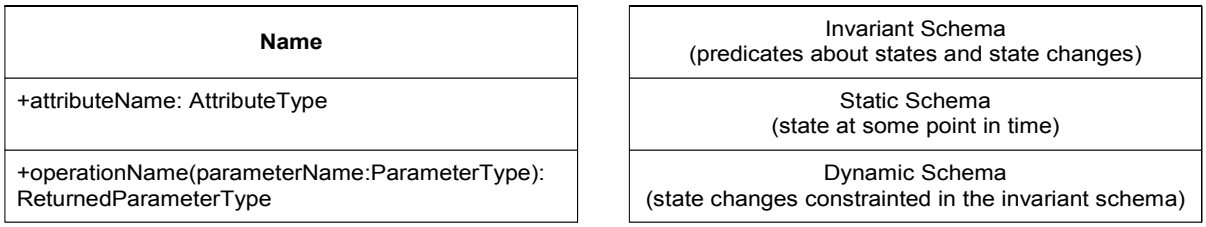

Figure 6.5: Systemic Notation for the RM-ODP objects as based on the UML notation for classes and objects.

In UML there are three fields (panes) inside a regular rectangle that is used to represent classes and objects. The upper pane is used to present the name of a class or an object, the middle pane - to show attributes and the lower – to show operations. As we see, there is a straightforward correspondence between the UML notational structure and the three informational schemas that wee need to represent

[14] Note that objects in the context of their informational characteristics are called "information objects" in part 3 of RM-ODP, section 6.

(as additional prove see Figure 3-21 in Chapter 3.26 in [40]). Thus we will use the UML upper pane to present the invariant schema, the middle pane – for the static schema and the lower pane – for the dynamic schema (see Figure 6.5).

Thus the proposed Systemic Notation will allow expressing the duality of static and dynamic informational aspects for the RM-ODP objects. This is an important added value to the UML notation. Because UML doesn't provide means for explicit presentation of the two aspects essentially constituting whole information about an object as it is done within the invariant schema.

6.2.3. Contextual Object Representation

The second important problem that needs to be solved in Systemic Notation is about representation of the RM-ODP objects in the context where they are defined. This includes two issues, namely:

- presentation of objects decompositions and compositions;
- presentation of an object with its environment.

The former is important to specify lifecycle dependencies for objects; the latter is of crucial importance for modeling interactions of an object (since by definition an RM-ODP object can interact only with its environment).

To model decompositions and compositions UML proposes three possible styles. Figure 6.6 (a copy of Example 3.47.4, Figure 3-36 from the UML specifications [40]) shows the three UML styles to present decompositions/compositions of classes.

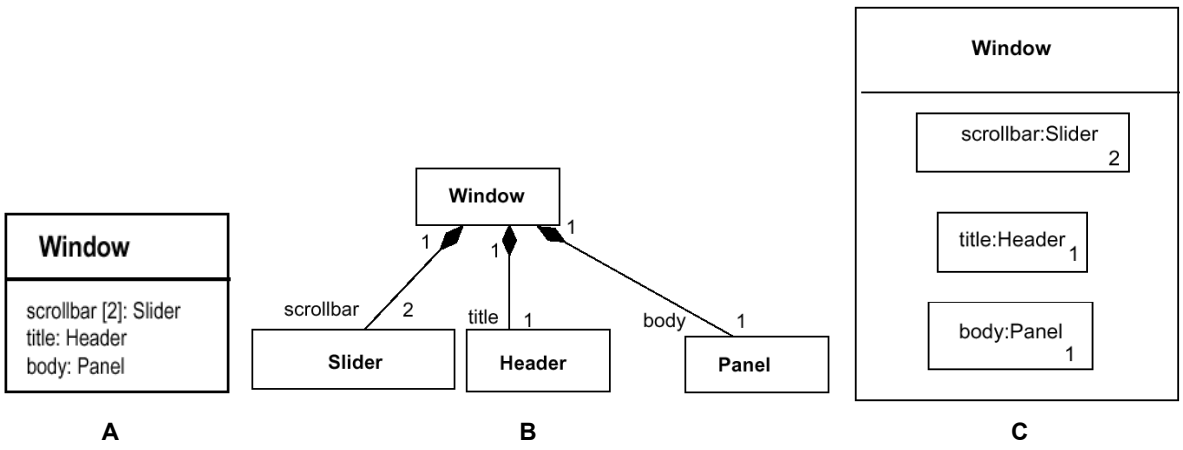

Figure 6.6: Three ways to present decomposition/composition in UML.

Among the three styles from Figure 6.6 we consider style "C" as the most appropriate for the base for Systemic Notation. Indeed, as we explained in Section 6.1, the style "A" presentation in UML can also be interpreted as a class containing

references to the set of independent objects as its attributes and not as a class composed by the subordinate objects (see Figure 6.1 and Figure 6.2). Style "B" eliminates this problem of style "A". However "C" is still better then "B" because "C" has potential of showing explicitly the environments for each of the components inside the composite class. This advantage of style "C" is particularly important because, as we mentioned, it is one of the targets for our Systemic Notation to have objects represented with their environments. This will allow for a representation of interactions between objects. To use this potential of style "C" we just need to refrain from indicating the multiplicity information for the RM-ODP component objects and instead we need to draw each component object explicitly (see Figure 6.7).

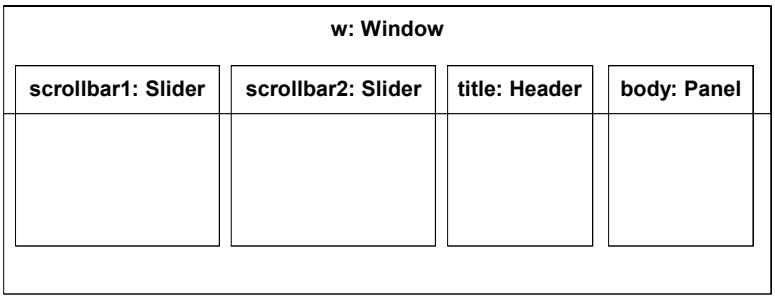

Figure 6.7: UML-style decomposition in Systemic Notation with explicit drawing of all the component objects.

The objects presented in Figure 6.7 are the same as those from Figure 6.6. As we see in Figure 6.7, for every of the component objects it is possible to show its corresponding environment that will be the rest of "w: Window" containing the other component objects. This is a very important feature for modeling of interactions of component objects. We will show this on an example later in this chapter.

Now we can combine the style for presenting composite objects with the style that presents invariant, static and dynamic schemas that we introduced in Figure 6.5 b). Figure 6.8 shows the resulting style.

As we see in Figure 6.8, the defined style allows the presentation of the three information schemas for all the component objects as well as for the composite object.

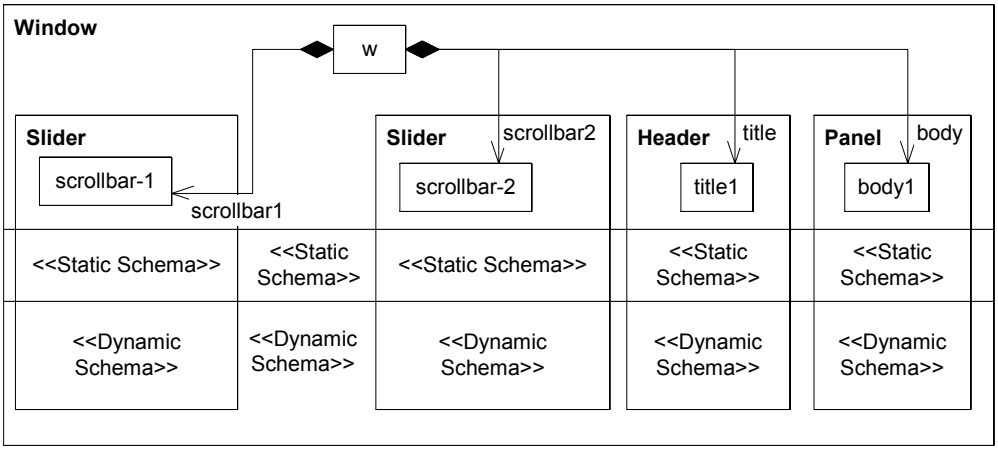

Figure 6.8: UML-style decomposition in Systemic Notation with explicit drawing of all the component objects.

6.2.4. Example of a Modeling Problem for Systemic Notation

Let us demonstrate on example the advantages that Systemic Notation is able to bring to modelers. As the example we will represent graphically a piece of Java code. Even if the example is a technical one, the presented reasoning is equally applicable on modeling problems belonging to any organizational level. The same notational artifacts can be used to model a business activity, an IT system architecture or software components.

Let us consider a Java application that consists of a window ("Frame1") with a button ("button1"). Here is the application code:

```
public class Frame1 extends Form
{       int i;
        X   x;
        Button button1 = new Button;

        public Frame1() // Constructor
        {       super();
                this.x = new X();       }

        private void button1_click()
        {       this.i = this.x.getA();   }
public class X
  {     int a;
        X() // Constructor
        {       this.a = 1;             }
        public int getA()
        {       return (this.a);        }
```

```
  } // X
} // Frame1
```

When a user clicks on the button, the method "button1_click()" is invoked. This method performs the assignment "this.i = this.x.getA()". Let us consider what is happening while the assignment is executed. As we see in the code, an object of type "Frame1" (let's assume that it is identified as "f") is composed of several parts. It includes an object[15] instance of type "int" that is referenced as "i" within "f". The instance is identified as "i1" and is automatically created in the "Frame1" constructor.

In addition, it includes an object instance of type "X" that is referenced as "x". The instance is identified as "x1" and is explicitly created by the statement "this.x = new X()". These parts are initialized during the construction of "f", which means that within the method "button1_click()" we are referencing already existing objects "i1" and "x1".

We present the component object specification followed by the composite object specification of "Frame1".

6.2.5. Example of a Component Object Specification

Figure 6.9 represents the component object "f" of the type "Frame1". Note that "f" is supposed to be a component of a larger system that is not represented here. It is interesting to describe the way the object "f" is specified.

The upper pane represents the invariant schema of an object. The invariant shows that, within the object "f", a button exists. Only the button is represented, as the other objects are not visible from outside the object "f".

The middle pane corresponds to the static schema of an object containing the structural part of the invariant information. It states that at time [@t1] (i.e. immediately before the "button1_click" action) "f" object refers to its "button1" object as "this.b". Note that "this" is a keyword representing the object "f". The static schema can of course only be represented for specific moments in time that must exist within the corresponding object lifecycle.

The lower pane represents the dynamic schema containing the behavioral part of the object information. The dynamic schema represents a certain part of the object behavior that it exhibits during its lifecycle. We use the UML activity diagram style to represent this information. The behavioral part presents that "f" accepts the button click from the environment (by executing a server interaction) and then executes the

[15] We use the words "object" and "type" correspondingly to the RM-ODP definitions. In java there is a slight difference, namely "an object is a class instance or an array" [12], which doesn't include an instance of int that is defined as a primitive type. The int should be wrapped either in the Integer or in an array to be instantiated as a real java object.

corresponding server processing. Note the comment outside the object box "f"; it represents the parameter value coming from the environment.

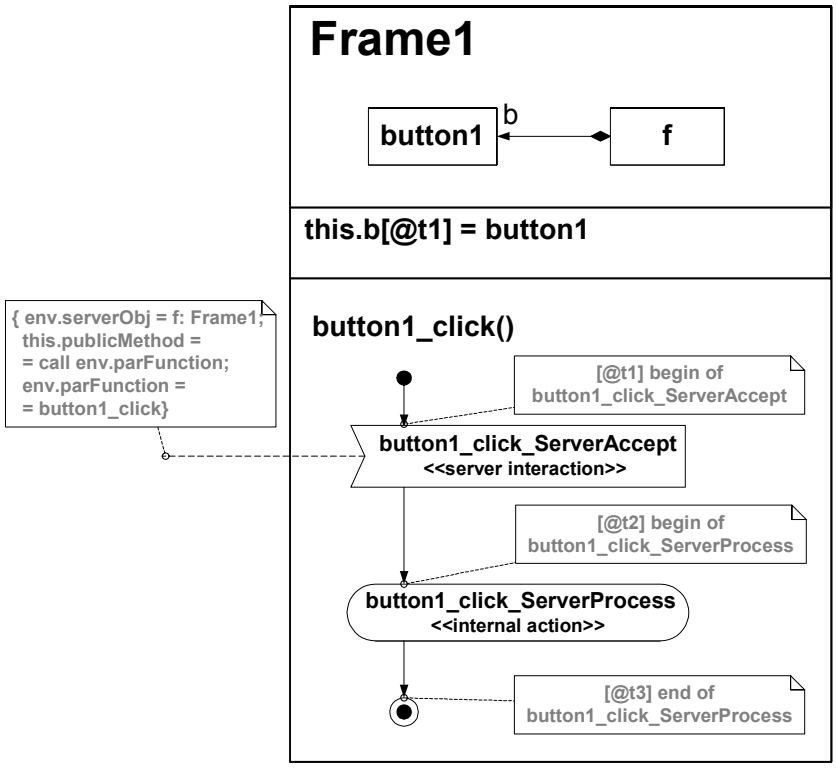

Figure 6.9: Example of Systemic Notation: Frame1 external representation

6.2.6. Example of a Composite Object Specification

Figure 6.10 presents the same object "f" but now as a composite object. The composite object specification presents an internal view on the system while the component object specification presented an external view. Presentations of both the external view (the view of an observer of the system) and the internal view (the view that presents the system external information) were mentioned in the review of requirements for Systemic Notation. It is important to have the two views in a system specification because this allows a modeler to separate and thus to manage efficiently external and internal responsibilities for a system.

It is interesting to compare the representation of the component object and the one of the corresponding composite object. We can see two object boxes corresponding to the "i1" and "x1" objects inside the object box representing the

composite object "f". We also should have presented the same kind of box for "button1", but to simplify the example we decided not to present "button1" in details.

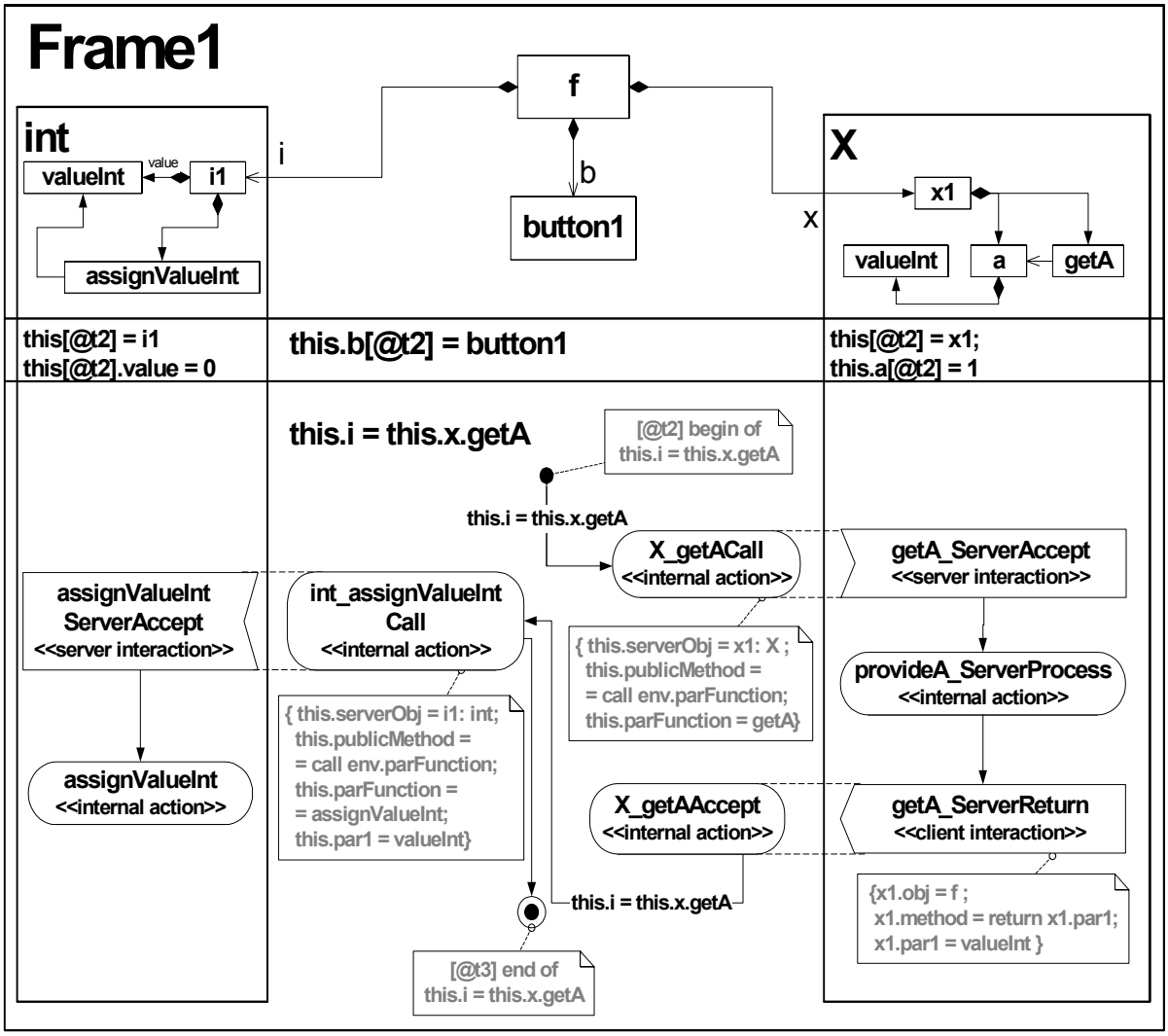

Figure 6.10: Example of Systemic Notation: Frame1 internal representation

If we are interested in the nature of any composition of objects then we should specify the mechanisms that would allow it to yield a composite object. We can define these mechanisms as "composition constraints". They are a set of structural and behavioral constraints that allow the resulting composite object to fulfill its mediation responsibilities with regard to the component objects participating in the composition. Note that all model elements shown in "f" and not in "x1" or "i1" present the composition constraints for the components "x1" and "i1" composed in the composite "f". Here we are able to show explicitly the validity of a famous expression: "*the whole is more than the sum of parts*". In summary, a composite object is

the result of the composition of two or more component objects with the corresponding composition constraints. A component object is defined by its structural and behavioral limits. These limits are necessary and sufficient for it to participate in a composition. The structural limits are defined by the object's external state specification. The behavioral limits are determined by the object's interfaces specification.

In Figure 6.10 all the objects (including "f") are defined with the three panes (invariant, static, dynamic). Note that the elements shown inside the "i1" (respectively "x1") object box represent the sub-model of "i1" (respectively "x1").

We see that inside the invariant schemas of object there both static and dynamic informational concepts (for example for "x1": "a" attribute and "valueInt" are structural informational concepts while "getA" transaction is a behavioral informational concept). Thanks to the invariant schema we are able to express this duality and to specify particular relations between the structural and the behavioral informational concepts existing in the context of each object.

Considering "i1" and "x1", we see that as they are declared independently: the object "i1" exists inside the context of "f" and doesn't have any relation with the object "x1". Analogously, "x1" doesn't have any relation with "i1". So, because of this independence, "i1" is not able to have a direct communication with "x1". Nevertheless both "i1" and "x1" exist within the same object "f"; so communicating with "f" they can transmit information to each other under the condition that "f" is fulfilling the above-mentioned composition constraints.

Within the method "button1_click()", which belongs to object "f" (and not to the "button1"), "f" performs assignment "this.i = this.x.getA()". Here it is intended to assign ("=") a value to its "i1" object. The value that it will assign to "i1" should be further found within the "x1" object by calling its "getA()" method. This equation expresses the composition constraint for the assignment. Knowing this constraint (as part of the code of "f"), "f" performs an internal action to execute the "getA()" method of its object "x1". From the point of view of "x1", this internal action is perceived as a server interaction coming from "x1" environment (i.e. from "f").

A possibility to have multiple subjective perspectives of one thing in the same specification is a powerful feature of Systemic Notation. This is an illustration of the constructivist approach principle: each object has its own perception of the same action occurrence. Here it also illustrates the concept of server interaction. The comment attached to the server interaction illustrates the passing of the structural parameters (which are essentially values such as "f.par1= valueInt"), and of the behavioral parameters (which are essentially actions to be made with values such as: "f.parFunction = assignValueInt").

The object "x1" executes the processing associated with the requested internal action and executes a client interaction returning the parameter "valueInt". This value

represents the integer value found in the "a" attribute of the "x1" object. The object "f" (i.e. the environment of "x1") perceives this sever interaction as being another internal action. Now, having received the resulting parameter from its "x1" and having the "this.i = this.x.getA()" composition constraint as an instruction for what needs to be done with "valueInt", "f" object executes yet another internal action to assign the value of the "i1" object to the received "valueInt". This internal action is perceived by "i1" as a server interaction with "valueInt" as parameter. And now it is "i1" who performs the internal action assigning its "valueInt" to the received parameter value.

This concludes the example that presents our Systemic Notation. As we showed, the notation provides a set of important advantages to modelers. Particularly, it is able:

- to present objects with their three informational aspects (invariant, static and dynamic information) expressing the static/dynamic duality of the informational nature;
- to present objects viewed and specified from an external perspective and from an internal perspective separating responsibilities as internal and external for each object;
- to present objects and their environments and thus to truly present interactions for an object;
- to present explicitly the context of objects compositions/decompositions (composition constraints) showing that "*the whole is more than the sum of parts*";
- to present different subjective viewpoints on the same informational aspect within a single specification.

6.3. Ontological Foundations for RM-ODP Framework

In our days there exist a multitude of technological solutions that propose different system architectures for information systems modeling. Constantly growing number of new system architectures and meta-data standards increases the difficulty of interoperability problems. Fortunately, the fundamental principles that are used by different system modeling frameworks are not so numerous. Understanding of ontological foundations for existing system modeling standards allows for the classification of the standards, and thus, provides system architects with the Ariadne's thread that helps to pass successfully through the labyrinth of heterogeneous system models.

The goal of this section is to present in details two key approaches existing in practical ontological engineering and to perform their comparative analysis. The analysis will familiarize the reader with strengths and weaknesses of the approaches, thus motivating preferences for their practical applications. In particular, we will

show that in system modeling a concreteness of viewpoints definitions allows for a gain in consistency of the represented system architecture but at the same time brings a lack of the architecture flexibility.

We will illustrate the first of the ontological approaches with the example of Model Driven Architecture, and the second – with the example of Reference Model of Open Distributed Processing.

Model Driven Architecture (MDA) ([14], [46]) proposed by the Object Management Group (OMG) is a recently emerging vision on system modeling that targets integration of different successful industrial solutions for the system architecture.

The scope of applications of MDA and RM-ODP and their goals are similar. Particularly, MDA deals with "*full lifecycle integration and interoperability of enterprise systems comprised of software, hardware, humans, and business practices*" [14]; and RM-ODP considers lifecycle of distributed systems from enterprise, information, computational, engineering and technology viewpoints. Both MDA and RM-ODP present ontologies for system modeling (for details see [5] and Chapter 5 correspondingly). However from the ontological engineering point of view, these frameworks employ two fundamentally different approaches.

6.4.1. Four-level Ontological Approach

As it is explained in [4], the MDA ontology uses an approach of four conceptual levels. This four-level approach is presented in Figure 6.11.

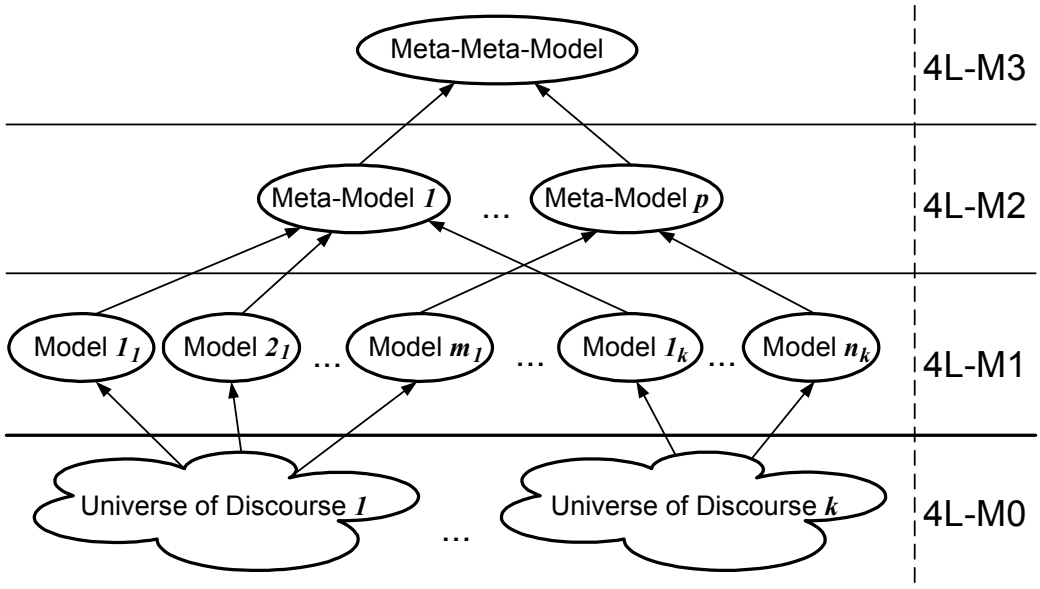

Figure 6.11: Four-level ontological approach (indexes k, m, n and p are natural numbers).

The lowest level (4L-M0) presents different subjects for modeling; each of them called as a universe of discourse. The only general requirement for a universe of discourse is that it should be interesting for a modeler as the subject to be modeled. Otherwise its modeling is not relevant. Thus in the general case we cannot say that a universe of discourse should be real, or imaginary, or having any other characteristic; whatever is interesting for a modeler qualifies automatically to be a universe of discourse.

Next level (4L-M1) contains different models of each of the universes of discourse. These models belong to diverse independent domains of interest with regard to the universe of discourse that they represent. It is possible that the same kind of interest is applicable to different universes of discourse, thus models of different universes of discourse may belong to the same domain of interest.

The next level (4L-M2) presents domain-specific meta-models: one meta-model for each of the domains of interest relevant for the 4L-M1 models. For a given domain of interest, its corresponding meta-model defines:

1) relations between different conceptual categories[16] that exist in the domain models;

2) internal conceptual structure for each of the conceptual categories existing in the domain models;

3) semantics for each of the modeling concepts within the conceptual categories.

The first and the second of these three definitions ensure coherency of the domain models. The third definition relates formally the domain models with the universes of discourse represented by the domain models. More concretely, corresponding to the notion of semantics introduced in [50], for every concept used inside the domain models there is its respective conceptualization of the universes of discourse that is defined in the meta-model.

And finally, 4L-M3 level presents a meta-meta-model. The meta-meta-model should be designed to allow for definition of all the existing in the scope of modeling interest meta-models and for their unification under a common framework. Thus a meta-meta-model is domain-independent and it contains the meta-characteristics for all the domain-specific meta-models.

An application of the four-level approach is presented with MDA. In the case of MDA a 4L-M1 model is used to describe an arbitrary universe of discourse. This model belongs to a particular "*platform-specific*" domain of interest in relation to the universe of discourse. The model should use a conceptual framework that is described in its corresponding "*platform-specific model*" (PSM) [39]. PSMs are the meta-

[16] A conceptual category can be defined as a group of concepts used for modeling and reserved for a particular modeling purpose. The number of conceptual categories that can be defined within a model corresponds to the number of such particular modeling purposes existing in the model.

models from 4L-M2. For example, CORBA, Java/EJB, .NET and other conceptual frameworks present possible PSMs within MDA. Then, in correspondence with the 4L-M3 meta-meta-model, MDA introduces a "*platform-independent model*" (PIM) [39] as the framework to integrate all the defined PSMs. The Meta-Object Facility (MOF) [38] is an example of PIM supported by Object Management Group within MDA. To summarize the MDA case of the four-level ontological approach application, let us quote [39]: "*A complete MDA application consists of a definitive PIM, plus one or more PSMs and complete implementations, one on each platform that the application developer decides to support.*"

6.4.2. Three-level Ontological Approach

Another ontological approach is based on three conceptual levels; it is applicable to the RM-ODP ontology that was introduced by the standard [21] and formally defined and explained here in the Chapter 5. The three-level approach is presented in Figure 6.12.

The lowest level (3L-M0) presents different subjects for modeling; each of them called as a universe of discourse. Analogously to the four-level approach, the only general requirement for a universe of discourse is that it should be interesting for a modeler as the subject to be modeled. Otherwise its modeling is not relevant. Thus in the general case we cannot say that a universe of discourse should be real, or imaginary, or having any other characteristic; whatever is interesting for a modeler qualifies automatically to be a universe of discourse.

Next level (3L-M1) contains models: one per each of the universes of discourse that are interesting for modeling. The models have a uniform structure; that is, all of them use the same modeling framework that is defined in a meta-model presented on the level 3L-M2. Such meta-model defines:

1) relations between different conceptual categories that exist in the 3L-M1 models;

2) internal conceptual structure for each of the conceptual categories existing in the 3L-M1 models;

3) semantics for each of the modeling concepts within the conceptual categories.

The first and the second of these three definitions ensure coherency of the 3L-M1 models. The third definition relates formally the 3L-M1 models with the universes of discourse represented by the 3L-M1 models. More concretely, corresponding to the notion of semantics introduced in [50], for every concept used inside the 3L-M1 models there is its respective conceptualization of the universes of discourse that is defined in the meta-model.

On the 3L-M1 level the models are disintegrated into their diverse domain-specific viewpoints. Since all the 3L-M1 models have a uniform structure, the

structure of viewpoints is also the same for all of the models. That is, if a specific viewpoint can be defined as relevant for one of the 3L-M1 models, then it will be automatically relevant for all the other 3L-M1 models, because all the 3L-M1 models use the same modeling framework defined in their common 3L-M2 meta-model.

The scope of the 3L-M1 viewpoints is limited by the scope of the 3L-M1 models. The scope for a particular viewpoint from 3L-M1 is less general then the scope of a 3L-M1 model, since it is related only to a specific domain within the model. But at the same time concepts within the scope of a viewpoint are more precise than their ancestors from models, since in a specific domain it is relevant to define the corresponding specific features that are not applicable in the general context of the original models. Thus the context of a 3L-M1 viewpoint is less broad but more profound than the context of the originating model of the viewpoint. We can call the 3L-M1 models as domain-independent in relation to the domain-specific 3L-M1 viewpoints for those models.

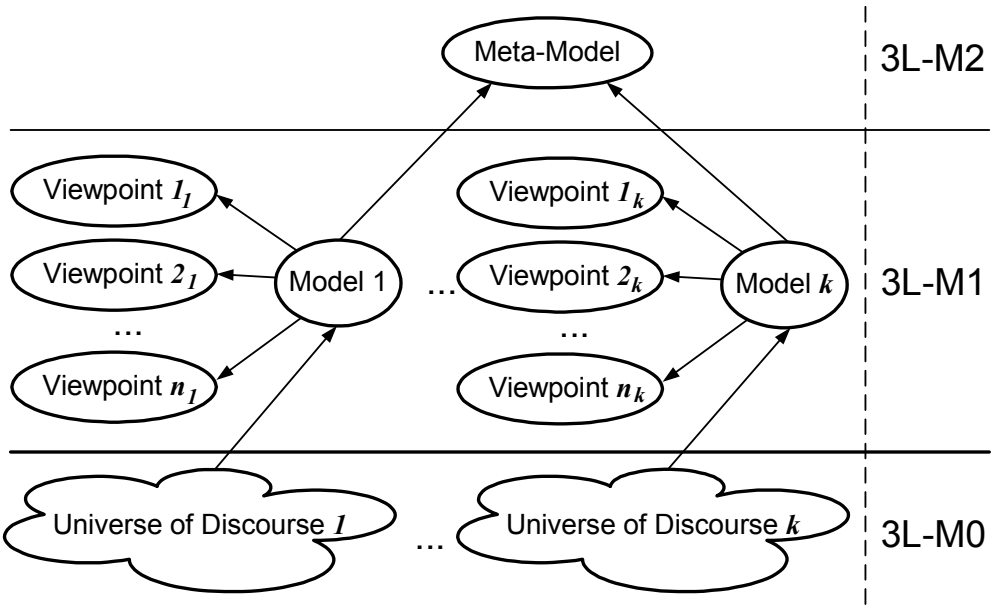

Figure 6.12: Three-level ontological approach (indexes k and n are natural numbers).

Let's demonstrate an application of the three-level approach on example of RM-ODP. In this case a 3L-M1 model represents an arbitrary universe of discourse, and should be constructed by means of the RM-ODP basic modelling and specification concepts presented in "RM-ODP part 2: Foundations" [21] that is a part of the RM-ODP meta-model. The RM-ODP meta-model, that is an example of 3L-M2 meta-model, contains definitions of concepts and conceptual categories from part 2 of the standard, including the definitions for: RM-ODP 2-5 ("*categorization of concepts*"), RM-ODP 2-6 ("*basic interpretation concepts*"), RM-ODP 2-8 ("*basic modelling concepts*") and

RM-ODP 2-9 ("*specification concepts*"). The formalized version of the RM-ODP meta-model was presented in Chapter 5. It includes formal definitions for all the three points that we mentioned in the beginning of this section when introducing the basic requirements for a meta-model. The 3L-M1 viewpoints in the case of RM-ODP defined in "RM-ODP part 3: Architecture" [21]. There are five viewpoints introduced by the standard: enterprise, information, computational, engineering and technology, each of them defining its corresponding domain of interest in relation to the RM-ODP models.

6.4.3. Comparative Analysis

Now, as we have introduced two ontological approaches and illustrated them on examples, we can make their comparative analysis. But before starting with the comparison let us emphasize one important property of meta-modeling.

Namely, we would like to mention that a meta-model is always defined for a specific domain of modeling interest, and all the domains that have no intersection with the domain of interest will be beyond the scope of the meta-model applications. So-called "domain-independent" meta-models introduce a conceptual framework that is general enough to be instantiated in any specific domain of modeling interest. But of course, the definition of a domain-independent meta-model requires a definition of the scope for all the domains considered as interesting for modeling. Thus, the domain-independent meta-model is not applicable for the irrelevant domains, which ensures its completeness with regard to its application scope.

Now we can look at the difference in the structural organization of the presented ontological approaches. Both of them have a structure of diverse modeling perspectives: the models from 4L-M1 and the viewpoints from 3L-M1. Their principal difference here is the subjects of modeling for the 3L-M1 viewpoints and for the 4L-M1 models. The models from 4L-M1 are the diverse views on the universes of discourse from 4L-M0. While 3L-M1 viewpoints do not refer to the universes of discourse from 3L-M0; instead they are the views on the 3L-M1 models that, in their turn, are the uniform representations of their universes of discourse. Each of the two choices has an advantage in comparison with the other.

In the three-level approach, a 3L-M1 model is already the result of a universe of discourse modeling. Thus, within a modeling project there will be an authority that is responsible for the content of the 3L-M1 model. This ensures determinism in the model that is, in its turn, the subject of modeling for the diverse viewpoints. Hence the 3L-M1 model determinism makes possible to establish formal correspondences across the viewpoints.

In the four-level approach the models from 4L-M1 are the direct representations of the universes of discourse from 4L-M0. A universe of discourse is just a subject

for modeling; in general it is not a result of a prior modeling and thus cannot be controlled by a modeler. Hence there cannot be an authority that is responsible for the universe of discourse content, which makes it never possible to formally assert that different 4L-M1 models do model the same universe of discourse. And in practical applications of the four-level ontological approach we find quite often the situation when the 4L-M1 models are assumed to model the same universe of discourse. Unfortunately, as we explained, in this situation there is no authority that is responsible to give the same subject as modeling input for the different models. So, here nobody can insure that different modelers produce models of the same universe of discourse. Even in the cases when it may seem intuitively obvious to the different modelers that they consider the same subject for the inputs of their corresponding different models, they should not rely on this because nobody can take a responsibility to guarantee this.

Thus we showed that a formal consistency across multiple 4L-M1 models is unreachable, while it is reachable across multiple 3L-M1 viewpoints due to the determinism of 3L-M1 models. Of course, this advantage of the three-level ontological approach doesn't come for free. Flexibility is the price that this approach has to pay for the mentioned determinism.

Namely, in the three-level case 3L-M1 viewpoints always depend on the 3L-M1 models. Thus in a particular ontology the 3L-M1 viewpoints have to be concretely defined for the corresponding ontology-specific 3L-M1 models. And since the scope of any 3L-M1 model is limited, the 3L-M1 viewpoints will correspondingly have pre-defined limits in their scopes. Hence it is impossible to consider any viewpoint that would go beyond these pre-defined limits. Of course, here we mean the limits in broadness of a viewpoint and not in its profoundness. As we explained when introducing the three-level approach, the context of a viewpoint is less broad and more profound than the context of its originating model. It clarifies that the pre-defined scope limits that we are discussing here apply on the broadness of viewpoints and not on their profoundness. As we showed, the RM-ODP example demonstrates this with the definitions of five viewpoints within the broadness of the scope of RM-ODP models.

The four-level approach doesn't have this limitation. Here a 4L-M1 model may have an arbitrary scope that will be determined by its corresponding 4L-M2 meta-model. However, the 4L-M2 meta-model should be integrated within the 4L-M3 meta-meta-model. So, if the arbitrary scope from the 4L-M2 meta-model did not exist in the 4L-M3 meta-meta-model, then the meta-meta-model should be extended. Therefore, if a meta-meta-model of the four-level framework is extendable, then the scope limits for a 4L-M1 model are not pre-defined.

Thus, the four-level approach is more flexible then the three-level approach. As we can conclude from the two previous paragraphs the gain in flexibility of the four-

level approach is in fact possible because here we can define additional 4L-M2 meta-models and then extend the 4L-M3 meta-meta-model. While with the three-level approach it is not possible to define additional 3L-M1 models or viewpoints, because their scopes are pre-defined in the 3L-M2 meta-model, and any domain that is out of the pre-defined scope considered to be beyond the modeling interest.

The explained flexibility even supports a potential possibility of the three-level approach integration within the frame of the four-level approach. Indeed, a 3L-M1 model could be considered as one of the 4L-M1 models, and the 3L-M2 meta-model as one of the 4L-M2 meta-models. However this integration would not be reasonable in the general case, because overall objectives of both approaches are the same. And both approaches succeed to achieve the objectives with the similar degrees of success. Particularly the 3L-M2 meta-model and the 4L-M3 meta-meta-model have similar generalities of their scopes, as well as diverse 3L-M1 viewpoints and diverse 4L-M1 models. Thus, in the general case it is not reasonable to consider one of the approaches as a subordinate part of another.

PART III SUMMARY

In this part of the thesis we presented an application of Triune Continuum Paradigm (that was defined in Part I of the thesis) on the RM-ODP ("Reference Model of Open Distributed Processing") ISO/ITU standard conceptual framework. By means of this application we reinforced the standard reference model making use of the formal interpretation constraints provided by the paradigm. Because of the intrinsic internal consistency of the foundations of RM-ODP, for the standard conceptual framework it was possible to benefit to the full extend from advantages of the paradigm (in particular, from: the internal consistency of the paradigm's metamodel, its logical coherency of interpretation of subjects of modeling, its formalized semantics and its theoretically justified foundations). Thus with the aid of our defined Triune Continuum Paradigm we were able to formally interpret RM-ODP standard conceptual framework as a single consistent construction.

So this paradigm application realizes an important result that was never achieved previously: a single consistent formalization of the RM-ODP standard conceptual framework. Such formalization was officially defined as one of the standard's goals, but this goal was never achieved previously (neither by RM-ODP standard itself, nor by the standard-related research). Our formalization successfully accomplishes this defined goal and thus presents a concrete example of formal ontology for general system modeling.

This internally consistent formal view on RM-ODP helps us to clarify the RM-ODP modeling framework to make it more accessible to modelers such as system architects, designers and implementers, while opening the way for the formal verification of RM-ODP models, either within an ODP viewpoint or across multiple ODP viewpoints.

We expressed our formalization in a computer-interpretable form with Alloy, the language for description of structural properties of a model. Thus different models that use our RM-ODP-based formal ontology for system modeling can be simulated and checked for consistency with the aid of the corresponding software tool (Alloy constraint analyzer).

Also in this part of the thesis we presented an example of practical modeling application of our defined formal ontology for system modeling. This example demonstrated the practical value of logical rigor that this RM-ODP-based ontology brings to the process of object-oriented modeling.

To facilitate the everyday use of our proposed formalism within the system modeling community we have defined a systemic notation that successfully

represents the formal constructs of the RM-ODP-based ontology for system modeling. The notation is based on traditional UML artifacts, - this gives to modelers a possibility to operate with the formal ontological constructs by means of simple UML-style drawings. Our notation features several advantages in comparison with the conventional UML notation. In particular, our notation:

- supports multiple viewpoints on a presented object (e.g. a subsystem seen from its own perspective, from the perspectives of other subsystems participating in relations with it, from the perspective of its super-system);
- supports contextual object representation (object with its environment that is presented explicitly);
- properly presents information about objects (static and dynamic information about objects and mutual dependency of the two informational aspects).

For the additional promotion of the defined RM-ODP based formal ontology for system modeling we presented a comparative analysis of two ontological approaches:

- the three-level ontological approach (that is exhibited by the framework of RM-ODP modeling viewpoints),
- and the four-level ontological approach (that is exhibited by MDA ("Model Driven Architecture"), a vision on system modeling that targets integration of different successful industrial solutions for system architectures).

We analyzed the comparative strengths and weaknesses of the two approaches; in particular we showed that in system modeling a concreteness of definitions of the modeling viewpoints allows for a gain in consistency of the represented system architecture, but at the same time it brings a lack of the architecture flexibility. Thus our analysis helps to the readers to grasp the preferences for a concrete practical application of our defined RM-ODP-based ontology for system modeling.

Concluding Part III of the thesis, we would like to mention that our results could serve as a foundation for further development of RM-ODP-based software tools that would allow for the automation of management of modeling complexity during the design phase of a software development process. In particular, we are talking about the complexity of structural relations within a model, such as relations between different concept categories, within a concept category between different concepts, and between different levels of abstraction. Also, we think that our results can help to ODP researchers to better understand the standard, and hope that the results could help in a future promotion of the Reference Model to the everyday practices of software engineers, designers and architects.

CONCLUSION

In this thesis we defined Triune Continuum Paradigm that presents concrete solutions for a number of fundamental problems of general system modeling that cause manifest and visible difficulties in the current modeling practices.

Our paradigm is an object-oriented modeling paradigm that includes a formally defined metamodel and its supporting philosophical and natural science foundations.

The philosophical and natural science foundations allowed us to define coherently the domains of "modeling", "conceptual modeling" and "general system modeling", and to position the formal metamodel as an efficient utility in the scope of general system modeling. For the modeling concepts defined in the metamodel, the ambition to serve as necessary and sufficient for a representation of the general system modeling scope was justified by the definition of coherent Tarski's declarative semantics [50] and of non-declarative semantic constraints. The meaning of the principal object-oriented terms (e.g. "object", "state", "action", "type", "instance", etc.) was unambiguously defined.

A particular originality of our paradigm is in the organization of its observer-relational frame of reference. For this frame of reference we propose to use two principal independent dimensions: the first presenting the spatiotemporal continuum (e.g. time and space intervals within models) and the second presenting the model constitution conceptual continuum (e.g. objects and their respective environments within models).

By introducing the observer-relational model constitution continuum, we extend the traditional 4-dimensional Minkowski's space-time framework [17] into the 5-dimensional space-time-constitution framework.

By positioning space-time and model constitution as independent dimensions in the same frame of reference, we automatically introduce the third conceptual category that emerges out of the first two: information about the mutual relation of model constitution and space-time. The SpaceTime Continuum, Model Constitution Continuum and Information Continuum are the three intrinsic features of our paradigm, the three foundations that ensure its efficiency for general system modeling. That is the reason we call our paradigm as "Triune Continuum Paradigm".

The metamodel of Triune Continuum Paradigm exhibits its internal consistency (supported by Russell's theory of types [43]) as well as its coherency and unambiguity in the interpretations of subjects of modeling (supported by the defined kind of Tarski's declarative semantics [50]). The metamodel is flexible; it allows its adaptation by different object-oriented conceptual frameworks as soon as these frameworks are

internally consistent (destitute of self-contradictions). Thus different existing frameworks can benefit from the logical rigor, internal consistency, interpretation coherency, formal presentation and solid theoretical foundations of the defined paradigm.

In particular, we demonstrated how the paradigm provides a potential for the constructive influence on the evolution of the following two object-oriented frameworks that are currently popular in system modeling community:

- Unified Modeling Language (UML) [40] that is a proposition of the Object Management Group (OMG) emerging from industrial practices;
- Reference Model of Open Distributed Processing (RM-ODP) [21] that is an ISO/ITU international standard.

We demonstrated that a realization of this potential provides improvements to the current state of both UML and RM-ODP.

In particular, for UML we showed how the paradigm provides concrete solutions for the following problems:

- absence of an explicit structural organization defined for the UML metamodel;
- absence of a formal declarative semantics in the UML metamodel;
- absence of theoretical justifications for the UML metamodel to represent the modeling scope that is targeted by UML.

We demonstrated that in its current state the UML metamodel is inconsistent: it contains self-contradictions and baseless references (references to undefined concepts), which prevent UML from fully enjoying the advantages of our paradigm.

For RM-ODP we showed how the paradigm provides all the necessary constraints for the standard interpretation that make possible a single consistent formalization of the RM-OPD conceptual framework. Such formalization was targeted by ISO/ITU as one of the standard goals. But because the standard doesn't provide enough information on the interpretation of its conceptual framework, this goal was impossible to achieve (and thus it was never achieved).

Because:

- our paradigm explicitly defines and formally presents the necessary interpretation rules,
- and the currently existing metamodel of RM-ODP is internally consistent (destitute of self-contradictions),

we were able to implement and present in this thesis a concrete computer-interpretable formalization of the RM-ODP conceptual framework. The importance of this result was justified by the corresponding unachieved ISO/ITU standard's goal, but it was not the only achievement: the formalization also defined a concrete formal ontology for system modeling and thus gave to modelers a very powerful modeling utility. With our formalization the semantics for the standard object-oriented terminology of RM-ODP (e.g. for terms like "object", "action", "state",

"type", "instance", etc.) became computer-interpretable, which allows for a computer-supported consistency check of RM-ODP models and also provides the possibility for creation of semantics-aware Computer-Aided Software Engineering tools (CASE-tools) to support the software system development.

To explain the practical value of the defined RM-ODP-based formal ontology, we demonstrated several concrete advantages that it brings to practical applications. In addition, to support this practical value we performed a comparative analysis of the RM-ODP ontological approach and of the ontological approach that was taken by the Model Driven Architecture (MDA [14], [46], - a system modeling framework, proposed by OMG, that targets integration of different industrial solutions for system architecture).

With the aid of the defined fundamental principles of our paradigm and with the aid of RM-ODP definitions, we proposed a UML-based Systemic Notation as yet another support for the RM-ODP-based formal ontology. The notation (realized by means of drawings that are familiar to readers who use UML notation) allows the graphical representation of essential systemic features that are important in general system modeling. In particular, it:

- supports multiple viewpoints on a presented object (e.g. a subsystem seen from its own perspective, from the perspectives of other subsystems participating in relations with it, from the perspective of its super-system);
- supports contextual object representation (object with its environment that is presented explicitly);
- properly presents information about objects (static and dynamic information about objects and mutual dependency of the two informational aspects).

Thus we presented concrete arguments that support and promote practical applications of the defined formal ontology for general system modeling. Hence the formal ontology proved the practical value of Triune Continuum Paradigm. And the theoretical importance of the paradigm was supported more than once in this thesis by the results showing how the logical rigor, internal consistency, interpretation coherency, formal presentation and solid theoretical foundations of the defined paradigm make a positive difference in favor of the paradigm compared with the current state of the art in the object-oriented terminology.

APPENDIX A: TRIUNE CONTINUUM PARADIGM AND AXIOMATIC METHOD

This appendix is presented for the readers who are interested to see the correspondence of Triune Continuum Paradigm to the axiomatic method (see [1] on "axiomatic method").

The formal vision of relations between the universe of discourse and its models, defined in the scope of Triune Continuum Paradigm, is in agreement with basic principles of philosophy of science (see [1] on "philosophy of science").

In sciences we find a subject of investigation observed from the perspective of a given science, and posited truths proposed by the science in relation with the subject. Respectively, in the paradigm we find the universe of discourse as the subject of investigation from one side, and the model built by means of the basic modeling and specification concepts from the other side. So the model here is analogous to the "*posited truths concerning the world*" ([1] on "philosophy of science").

A science proposes the framework to model the subject of scientific investigation; analogously Triune Continuum Paradigm proposes the framework to model the universe of discourse. Any science is based on an application of the axiomatic method (see [1] on "axiomatic method"), analogously in Triune Continuum Paradigm (and consequently in RM-ODP) we also have all the necessary parts of the method application. Specifically:

- model elements within a model of the universe of discourse defined by the paradigm's (and RM-ODP's) basic modeling and specification concepts are the same as in the terminology of [1] on "axiomatic method" [17]: "*(1) the "universe of discourse" (domain, genus) of entities constituting the primary subject matter of the science*" in the application of axiomatic method [1].
- the SpaceTime and Model Constitution continuums of Triune Continuum Paradigm essentially distinguish the spatiotemporal and the non-spatiotemporal essences in the models. These two essences are analogous to "*(2) the "primitive concepts" that can be grasped immediately without the use of definition*" in the application of axiomatic method ([1] on "axiomatic method").
- the decisions:

[17] Obviously, the terminology used in [1] for the axiomatic method introduction is different from the terminology that we use in Triune Continuum Paradigm. So, "universe of discourse", "entities", etc. from [1]'s axiomatic method have different meanings in respect to the meanings of our analogous terms. Thus here we describe the relation between the two terminologies.

i. to model the perceived entities from the universe of discourse by means of the model elements in the model,
ii. to model the perceived first-order propositions from the universe of discourse by means of the first-order propositions in the model (that is by means of the basic modeling concepts),
iii. to model the perceived higher-order propositions from the universe of discourse by means of the higher-order propositions in the model (that is by means of the specification concepts)

in our paradigm are analogous to: "*(3) the "primitive propositions" (or "axioms"), whose truth is knowable immediately without the use of deduction*" in the application of axiomatic method ([1] on "axiomatic method").

- all the definitions for the concrete basic modeling and specification concepts in the paradigm are the same as "*(4) an immediately acceptable "primitive definition" in terms of primitive concepts for each non-primitive concept*" in the application of axiomatic method ([1] on "axiomatic method").
- the usual first-order predicate logic that is used in the paradigm (for example for construction of complex specification concepts) is the same as "*(5) a deduction (...) for each non-primitive accepted proposition*" in the application of axiomatic method ([1] on "axiomatic method").

This demonstration shows that Triune Continuum Paradigm exhibits axiomatic method defined in [1] as "*a method for reorganizing the accepted propositions and concepts of an existing science in order to increase certainty in the propositions and clarity in the concepts*".

APPENDIX B: RM-ODP STANDARD PART II: FOUNDATIONS, CLAUSES 5-9

This appendix contains an extract from the RM-ODP standard [21]. The extract includes the complete clauses 5, 6, 7, 8 and 9 of the RM-ODP Part II: Foundations.

/* ---------- beginning of the quote from [21] ----------*/

5 Categorization of concepts

The modelling concepts defined in this Recommendation | International Standard are categorized as follows:

a) Basic interpretation concepts: concepts for the interpretation of the modelling constructs of any ODP modelling language. These concepts are described in clause 6.
b) Basic linguistic concepts: concepts related to languages; the grammar of any language for the ODP Architecture must be described in terms of these concepts. These concepts are described in clause 7.
c) Basic modelling concepts: concepts for building the ODP Architecture; the modelling constructs of any language must be based on these concepts. These concepts are described in clause 8.
d) Specification concepts: concepts related to the requirements of the chosen specification languages used in ODP. These concepts are not intrinsic to distribution and distributed systems, but they are requirements to be considered in these specification languages. These concepts are described in clause 9.
e) Structuring concepts: concepts that emerge from considering different issues in distribution and distributed systems. They may or may not be directly supported by specification languages adequate for dealing with the problem area. Specification of objects and functions that directly support these concepts must be made possible by the use of the chosen specification languages. These concepts are described in clauses 10 to 14.
f) Conformance concepts: concepts necessary to explain the notions of conformance to ODP standards and of conformance testing. These concepts are defined in clause 15.

ITU-T Recommendation X.903 | ISO/IEC 10746-3 uses the concepts in this Recommendation | International Standard to specify the characteristics for distributed processing to be open. It is organized as a set of viewpoint languages. Each viewpoint language refines concepts from the set defined in this Recommendation | International Standard . It is not necessary for all viewpoint languages to adopt the same notations. Different notations may be chosen as appropriate to reflect the requirements of the viewpoint. These notations may be natural,

formal, textual or graphical However, it will be necessary to establish correspondences between the various languages to ensure overall consistency.

6 Basic interpretation concepts

Although much of the ODP Architecture is concerned with defining formal constructs, the semantics of the architectural model and any modelling languages used have to be described. These concepts are primarily meta-concepts, i.e. concepts which apply generally to any form of modelling activity. It is not intended that these concepts will be formally defined, nor that they be used as the basis of formal definition of other concepts.

Any modelling activity identifies :

a) elements of the universe of discourse;
b) one or more pertinent levels of abstraction.

The elements of the universe of discourse are entities and propositions.

6 . 1 Entity: Any concrete or abstract thing of interest. While in general the word entity can be used to refer to anything, in the context of modelling it is reserved to refer to things in the universe of discourse being modelled.

6 . 2 Proposition: An observable fact or state of affairs involving one or more entities, of which it is possible to assert or deny that it holds for those entities.

6 . 3 Abstraction: The process of suppressing irrelevant detail to establish a simplified model, or the result of that process.

6 . 4 Atomicity: An entity is atomic at a given level of abstraction if it cannot be subdivided at that level of abstraction.
Fixing a given level of abstraction may involve identifying which elements are atomic.

6 . 5 System: Something of interest as a whole or as comprised of parts. Therefore a system may be referred to as an entity. A component of a system may itself be a system, in which case it may be called a subsystem.

NOTE - For modelling purposes, the concept of system is understood in its general, system-theoretic sense. The term "system" can refer to an information processing system but can also be applied more generally.

6 . 6 Architecture (of a system): A set of rules to define the structure of a system and the interrelationships between its parts.

7 Basic linguistic concepts

Whatever the concepts or semantics of a modelling language for the ODP Architecture, it will be expressed in some syntax, which may include linear text or graphical conventions. It is assumed that any suitable language will have a grammar defining the valid set of symbols and

well-formed linguistic constructs of the language. The following concepts provide a common framework for relating the syntax of any language used for the ODP Architecture to the interpretation concepts.

7 . 1 Term: A linguistic construct which may be used to refer to an entity.
The reference may be to any kind of entity including a model of an entity or another linguistic construct.

7 . 2 Sentence: A linguistic construct containing one or more terms and predicates; a sentence may be used to express a proposition about the entities to which the terms refer.
A predicate in a sentence may be considered to refer to a relationship between the entities referred to by the terms linked.

8 Basic modelling concepts

The detailed interpretation of the concepts defined in this clause will depend on the specification language concerned, but these general statements of concept are made in a language-independent way to allow the statements in different languages to be interrelated.
The basic concepts are concerned with existence and activity: the expression of what exists, where it is and what it does.

8 . 1 Object: A model of an entity. An object is characterized by its behaviour (see 8.6) and, dually, by its state (see 8.7). An object is distinct from any other object. An object is encapsulated, i.e. any change in its state can only occur as a result of an internal action or as a result of an interaction (see 8.3) with its environment (see 8.2).
An object interacts with its environment at its interaction points (see 8.11).
Depending on the viewpoint, the emphasis may be placed on behaviour or on state. When the emphasis is placed on behaviour, an object is informally said to perform functions and offer services (an object which makes a function available is said to offer a service). For modelling purposes, these functions and services are specified in terms of the behaviour of the object and of its interfaces (see 8.4). An object can perform more than one function. A function can be performed by the cooperation of several objects.

NOTES

1 - The concepts of service and function are used informally to express the purpose of a piece of standardization. In the
ODP family of standards, function and service are expressed formally in terms of the specification of the behaviour of objects and of the interfaces which they support. A "service" is a particular abstraction of behaviour expressing the guarantees offered by a service provider.
2 - The expression "use of a function" is a shorthand for the interaction with an object which performs the function.

8 . 2 Environment (of an object): the part of the model which is not part of that object.

NOTE - In many specification languages, the environment can be considered to include at least one object which is able to participate without constraint in all possible interactions (see 8.3), representing the process of observation.

8 . 3 Action: Something which happens.

Every action of interest for modelling purposes is associated with at least one object.
The set of actions associated with an object is partitioned into **internal actions** and **interactions**. An internal action always takes place without the participation of the environment of the object. An interaction takes place with the participation of the environment of the object.

NOTES

1 - "Action" means "action occurrence". Depending on context, a specification may express that an action has occurred, is occurring or may occur.

2 - The granularity of actions is a design choice. An action need not be instantaneous. Actions may overlap in time.

3 - Interactions may be labelled in terms of cause and effect relationships between the participating objects. The concepts that support this are discussed in 13.3.

4 - An object may interact with itself, in which case it is considered to play at least two roles in the interaction, and may be considered, in this context, as being a part of its own environment.

5 - Involvement of the environment represents observability. Thus, interactions are observable whereas internal actions are not observable,because of object encapsulation.

8 . 4 Interface: An abstraction of the behaviour of an object that consists of a subset of the interactions of that object together with a set of constraints on when they may occur.
Each interaction of an object belongs to a unique interface. Thus the interfaces of an object form a partition of the interactions of that object.

NOTES

1 - An interface constitutes the part of an object behaviour that is obtained by considering only the interactions of that interface and by hiding all other interactions. Hiding interactions of other interfaces will generally introduce nondeterminism as far as the interface being considered is concerned.

2 - The phrase "an interface between objects" is used to refer to the binding (see 13.4.2) between interfaces of the objects concerned.

8 . 5 Activity: A single-headed directed acyclic graph of actions, where occurrence of each action in the graph is made possible by the occurrence of all immediately preceding actions (i.e. by all adjacent actions which are closer to the head).

8 . 6 Behaviour (of an object): A collection of actions with a set of constraints on when they may occur.
The specification language in use determines the constraints which may be expressed. Constraints may include for example sequentiality, non-determinism, concurrency or real-time constraints.
A behaviour may include internal actions.
The actions that actually take place are restricted by the environment in which the object is placed.

NOTES

1 - The composition (see 9.1) of a collection of objects implicitly yields an equivalent object representing the composition. The behaviour of this object is often referred to simply as the behaviour of the collection of objects.

2 - Action and activity are degenerate cases of behaviour.

3 - In general several sequences of interactions are consistent with a given behaviour.

8 . 7 State (of an object): At a given instant in time, the condition of an object that determines the set of all sequences of actions in which the object can take part.
Since, in general, behaviour includes many possible series of actions in which the object might take part, knowledge of state does not necessarily allow the prediction of the sequence of actions which will actually occur.
State changes are effected by actions; hence a state is partially determined by the previous actions in which the object took part.
Since an object is encapsulated, its state cannot be changed directly from the environment, but only indirectly as a result of the interactions in which the object takes part.

8 . 8 Communication: The conveyance of information between two or more objects as a result of one or more interactions, possibly involving some intermediate objects.
NOTES
1 - Communications may be labelled in terms of a cause and effect relationship between the participating objects.
Concepts to support this are discussed in 13.3.
2 - Every interaction is an instance of a communication.

8 . 9 Location in space: An interval of arbitrary size in space at which an action can occur.

8 . 1 0 Location in time: An interval of arbitrary size in time at which an action can occur.
NOTES
1 - The extent of the interval in time or space is chosen to reflect the requirements of a particular specification task and the properties of a particular specification language. A single location in one specification may be subdivided in either time or space (or both) in another specification. In a particular specification, a location in space or time is defined relative to some suitable coordinate system.
2 - By extension, the location of an object is the union of the locations of the actions in which the object may take part.

8 . 1 1 Interaction point: A location at which there exists a set of interfaces .
At any given location in time, an interaction point is associated with a location in space, within the specificity allowed by the specification language in use. Several interaction points may exist at the same location. An interaction point may be mobile.

9 Specification concepts

9 . 1 Composition:

a) (of objects) A combination of two or more objects yielding a new object, at a different level of abstraction. The characteristics of the new object are determined by the objects being combined and by the way they are combined. The behaviour of a composite object is the corresponding composition of the behaviour of the component objects.
b) (of behaviours) A combination of two or more behaviours yielding a new behaviour. The characteristics of the resulting behaviour are determined by the behaviours being combined and the way they are combined.

NOTES

1 - Examples of combination techniques are sequential composition, concurrent composition, interleaving, choice, and hiding or concealment of actions. These general definitions will always be used in a particular sense, identifying a particular means of combination.
2 - In some cases, the composition of behaviours may yield a degenerate behaviour, e.g. deadlock, due to the constraints on the original behaviours.

9 . 2 Composite object: An object expressed as a composition.

9 . 3 Decomposition:

a) (of an object) The specification of a given object as a composition.
b) (of a behaviour) The specification of a given behaviour as a composition.

Composition and decomposition are dual terms and dual specification activities.

9 . 4 Behavioural compatibility: An object is behaviourally compatible with a second object with respect to a set of criteria (see notes) if the first object can replace the second object without the environment being able to notice the difference in the objects' behaviour on the basis of the set of criteria.
Typically, the criteria impose constraints on the allowed behaviour of the environment. If the criteria are such that the environment behaves as a tester for the original object, i.e. the environment defines the smallest behaviour that does not constrain the behaviour of the original object, the resulting behavioural compatibility relation is called extension.
The criteria may allow the replacement object to be derived by modification of an otherwise incompatible object in order that it should be an acceptable replacement. An example of such a modification might be hiding of additional parameters on certain interactions. In this way, an interaction of the new object can be made to look like an interaction of the original object. In such cases behavioural compatibility is called **coerced behavioural compatibility**. If no modification is necessary, behavioural compatibility is called **natural behavioural compatibility**.
The concept of behavioural compatibility defined above on objects applies equally well to the behavioural compatibility of templates and of template types.
Behavioural compatibility is reflexive, but not necessarily symmetric or transitive (though it may be either or both).

NOTES

1 - The set of criteria depends on the language in use and the testing theory applied.
2 - Behavioural compatibility (with respect to a set of criteria) can be defined on template (see 9.11) and template types (see 9.19) thus:

a) if S and T are object templates, S is said to be behaviourally compatible with T if and only if any S-instantiation is behaviourally compatible with some T-instantiation (see 9.13);
b) if U and V are object template types, U and V are said to be behaviourally compatible if their corresponding templates are *behaviourally compatible*.

9 . 5 Refinement: The process of transforming one specification into a more detailed specification. The new specification can be referred to as a refinement of the original one. Specifications and their refinements typically do not coexist in the same system description. Precisely what is meant by a more detailed specification will depend on the chosen specification language.

For each meaning of behavioural compatibility determined by some set of criteria (see 9.4), a specification technique will permit the definition of a refinement relationship. If template X refines a template Y, it will be possible to replace an object instantiated from Y by one instantiated from X in the set of environments determined by the selected definition of behavioural compatibility. Refinement relationships are not necessarily either symmetric or transitive.

9 . 6 Trace: A record of an object's interactions, from its initial state to some other state.
A trace of an object is thus a finite sequence of interactions. The behaviour uniquely determines the set of all possible traces, but not vice versa. A trace contains no record of an object's internal actions.

9 . 7 Type (of an <X>): A predicate characterizing a collection of <X>s. An <X> is of the type, or satisfies the type, if the predicate holds for that <X>. A specification defines which of the terms it uses have types, i.e. are <X>s. In RM-ODP, types are needed for, at least, objects, interfaces and actions.
The notion of type classifies the entities into categories, some of which may be of interest to the specifier (see the concept of class in 9.8).

9 . 8 Class (of <X>s): The set of all <X>s satisfying a type (see 9.7). The elements of the set are referred to as members of the class.
NOTES
1 - A class may have no members.
2 - Whether the size of the set varies with time depends on the definition of the type.

9 . 9 Subtype/supertype: A type A is a subtype of a type B, and B is a supertype of A, if every <X> which satisfies A also satisfies B.
The subtype and supertype relations are reflexive, transitive and anti-symmetric.

9 . 1 0 Subclass/superclass: One class A is a subclass of another class B, and B is a superclass of A, precisely when the type associated with A is a subtype of the type associated with B.
NOTE - A subclass is by definition a subset of any of its superclasses.

9 . 1 1 <X> Template: The specification of the common features of a collection of <X>s in sufficient detail that an <X> can be instantiated using it. <X> can be anything that has a type (see 9.7).
An <X> template is an abstraction of a collection of <X>s.
A template may specify parameters to be bound at instantiation time.
The definition given here is generic; the precise form of a template will depend on the specification technique used. The parameter types (where applicable) will also depend on the specification technique used.
Templates may be combined according to some calculus. The precise form of template combination will depend on the specification language used.

9 . 1 2 Interface signature: The set of action templates associated with the interactions of an interface.
An object may have many interfaces with the same signature.

9 . 1 3 Instantiation (of an <X> template): An <X> produced from a given <X> template and other necessary information. This <X> exhibits the features specified in the <X> template. <X> can be anything that has a type (see 9.7).
The definition given here is generic: how to instantiate an <X> template depends on the specification language used.
Instantiating an <X> template may involve actualization of parameters, which may in turn involve instantiating other <X> templates or binding of existing interfaces (see 12.4).
NOTES
1 - Instantiating an action template just results in an action occurring. The phrase "instantiation of an action template" is deprecated. "Occurrence of an action" is preferred.
2 - If <X> is an object, it is instantiated in its initial state. An object can participate in interactions immediately after its instantiation.
3 - Instantiations from different templates may satisfy the same type. Instantiations from the same template may satisfy different types.

9 . 1 4 Role: Identifier for a behaviour, which may appear as a parameter in a template for a composite object, and which is associated with one of the component objects of the composite object.
Specification of a template as a composition of roles enables the instantiation process to be explained as the association of a specific component of the resultant composite object with each role. The association of a component object with a role may result from the actualization of a parameter.

9 . 1 5 Creation (of an <X>): Instantiating an <X>, when it is achieved by an action of objects in the model.
<X> can be anything that can be instantiated, in particular objects and interfaces.
If <X> is an interface, it is either created as part of the creation of a given object, or as an additional interface to the creating object. As a result, any given interface must be part of an object.

9 . 1 6 Introduction (of an <X>): Instantiating an <X> when it is not achieved by an action of objects in the model.
NOTES
1 - An <X> can be instantiated either by creation or introduction but not both .
2 - Introduction does not apply to interfaces and actions since these are always supported by objects.

9 . 1 7 Deletion (of an <X>): The action of destroying an instantiated <X>. <X> can be anything that can be instantiated, in particular objects and interfaces.
If <X> is an interface, it can only be deleted by the object to which it is associated.
NOTE - Deletion of an action is not meaningful: an action just happens.

9 . 1 8 Instance (of a type): An <X> that satisfies the type.

9 . 1 9 Template type (of an <X>): A predicate defined in a template that holds for all the instantiations of the template and that expresses the requirements the instantiations of the template are intended to fulfill.

The object template subtype/supertype relation does not necessarily coincide with behavioural compatibility. Instances of a template type need not be behaviourally compatible with instantiations of the associated template. They do coincide if

a) a transitive behavioural compatibility relation is considered, and
b) template subtypes are behaviourally compatible with their template supertypes.

NOTES

1 - This concept captures the notion of substitubility by design.

2 - The form of the predicate that expresses the template type depends on the specification language used.

3 - As a shorthand, "instances of a template T" are defined to be "instances of the template type associated with template T".

4 - Figure 1 [For Figure 1 please see [21] – note by A. Naumenko] illustrates the relationships between some of the concepts: template type, template class, etc. The set of instances of **t** contains both the set of instantiations of **t** and the sets of all instantiations of subtypes of **t**. The sets of instantiations of different templates are always disjoint.

9 . 2 0 Template class (of an <X>): The set of all <X>s satisfying an <X> template type, i.e. the set of <X>s which are instances of the <X> template. <X> can be anything that has a type (see 9.7).

Each template defines a single template class, so we may refer to instances of the template as instances of the template-class.

The notion of class is used to refer to a general classification of <X>s. Template class is a more restrictive notion where the members of a template class are limited to those instantiated from the template (or any of its subtypes), i.e. those <X>s which satisfy the <X> template type.

NOTE - Given a template type, we may shorten statements of the form "the template class associated with template A
is a subclass of the template class associated with template B" to "template A is a subclass of template B" or "template A is a subtype of template B".

9 . 2 1 Derived class/base class: If a template A is an incremental modification of a template B, then the template class CA of instances of A is a derived class of the template class CB of instances of B, and the CB is a base class of CA.

The criteria for considering an arbitrary change to be an incremental modification would depend on metrics and conventions outside of this Recommendation | International Standard. If the criteria allow, a derived class may have several base classes.

The incremental modification relating templates must ensure that self-reference or recursion in the template of the base class becomes self-reference or recursion in the template of the derived class.

The incremental modification may, in general, involve adding to or altering the properties of the base template to obtain the derived template.

Classes can be arranged in an inheritance hierarchy according to derived class/base class relationships. This is the interpretation of inheritance in the ODP Reference Model. If classes

can have several base classes, inheritance is said to be multiple. If the criteria prohibit suppression of properties from the base class, inheritance is said to be strict.
It is possible for one class to be a subclass of a second class without being a derived class, and to be a derived class without being a subclass. The inheritance hierarchy (where arcs denote the derived class relation) and the type hierarchy (where arcs denote the subtype or subclass relation) are therefore logically distinct, though they may coincide in whole or in part.

9 . 2 2 Invariant: A predicate that a specification requires to be true for the entire life time of a set of objects.

9 . 2 3 Precondition: A predicate that a specification requires to be true for an action to occur.

9 . 2 4 Postcondition: A predicate that a specification requires to be true immediately after the occurrence of an action.

/* ---------- end of the quote from [21] ----------*/

APPENDIX C: IF THE RM-ODP STANDARD WAS LIABLE TO MODIFICATION

This appendix is presented for the readers who are interested to see what kind of modifications would there be necessary to introduce in the RM-ODP standard part 2, for "RM-ODP part 2: Foundations" to conform perfectly to Triune Continuum Paradigm.

Chapter 1 (in particular, Section 1.3.2) shows that Triune Continuum Paradigm can provide theoretical foundations not only for the RM-ODP conceptual framework, but also for other object-oriented conceptual frameworks. And RM-ODP case is presented as one of the possible examples in Figure 1.10.

The current state of the RM-ODP conceptual structure [21] exhibits a significant degree of conformance to Triune Continuum Paradigm. But, as it is clear if comparing Figure 1.10 with the standard, to conform completely to Triune Continuum Paradigm, RM-ODP conceptual structure needs to be slightly modified. In particular:

Modification 1: The conceptual category for the analogs of "Model Elements" from Triune Continuum Paradigm should be added into the RM-ODP standard part 2.

- *Comment*: This modification would not influence any of the existing concepts of RM-ODP.

Modification 2: The concepts that are analogous to the "Time Interval", "Space Interval", "Point in Time" and "Point in Space" from Triune Continuum Paradigm should be added to the "Basic modelling concepts" category of RM-ODP (RM-ODP 2-8);

- *Comment*: This modification would not influence any of the existing concepts of RM-ODP.

Modification 3: The definitions of the RM-ODP "Environment", "Action" and "State" should be clarified to explicitly conform to the definitions of the corresponding terms in Triune Continuum Paradigm;

- *Comment*: The current state of definitions of the RM-ODP concepts influenced by this modification conforms implicitly to Triune Continuum Paradigm. With this modification the existing semantics for the concepts would remain valid, and the semantics would be clarified and/or completed.

Modification 4: the definition of the RM-ODP "Object" should be modified to conform to the definition of "Object" in Triune Continuum Paradigm;
- *Comment*: The current state of definition of the RM-ODP "Object" does not conform to Triune Continuum Paradigm. With this modification the declarative part of existing semantics of the RM-ODP "Object" that is: "*Object: A model of an entity.*" (RM-ODP 2-8.1) would be changed. The rest of the RM-ODP "Object" semantics (the non-declarative part) would remain valid.

Modification 5: The "Specification concepts" category of RM-ODP (RM-ODP 2-9) should be partitioned into two subcategories: one - corresponding to the generic and another - to the specific specification concepts of Triune Continuum Paradigm. The existing RM-ODP specification concepts should be assigned to one of the two subcategories, respectively, according to their either generic or specific nature.
- *Comment*: This modification would not influence the semantics of any of the existing concepts of RM-ODP.

Modification 6: The concepts 2-8.4, 2-8.5, 2-8.6, 2-8.8, 2-8.9, 2-8.10, 2-8.11, and the concepts of "Interaction" and "Internal action" (parts of 2-8.3) should be moved out from the RM-ODP "Basic modeling concepts" category to the RM-ODP "Specification concepts" category as being essentially specific specification concepts with regard to Triune Continuum Paradigm.
- *Comment*: This modification would not influence the semantics of any of the existing concepts of RM-ODP.

Modification 7: The "Categorization of concepts" part of RM-ODP (RM-ODP 2-5) should be clarified to explicitly conform to the definitions of the corresponding categorization in Triune Continuum Paradigm.
- *Comment*: The current state of definitions of RM-ODP 2-5 is vague and thus this part should be rewritten.

With these modifications the RM-ODP standard would be completely conformant with Triune Continuum Paradigm.

We also suggest one modification that is not directly related with Triune Continuum Paradigm, but that would improve the coherency of presentation of the standard part 2:

Modification 8: The concepts 2-6.3, 2-6.4, 2-6.6 should be moved out from the "Basic interpretation concepts" category (RM-ODP 2-6). A separate category called, for example, "Interpretation possibilities" should be allocated for these concepts. This separate category together with "Basic linguistic concepts" category (RM-ODP 2-7) should be placed in a super-category that would

present meta-meta-level definitions (the meta-level definitions for RM-ODP metamodel). The categories 2-5, 2-8, 2-9 and the remaining part of 2-6 should be placed in another super-category that would present meta-level definitions (the definitions for RM-ODP metamodel).

- *Comment*: Indeed, the concepts 2-6.3, 2-6.4, 2-6.6 as well as the concepts from RM-ODP 2-7 are defined on the meta-level for the RM-ODP metamodel. Obviously, the three concepts cannot be a part of the universe of discourse that is defined in RM-ODP 2-6 and consists only of entities and propositions.

APPENDIX D: ALLOY FORMALIZATION OF THE RM-ODP PART 2: FOUNDATIONS

This appendix presents the Alloy ([23], [24], [25]) code for the formalization of "RM-ODP part 2: Foundations". The formalization is defined and explained in the Chapters 4 and 5 of the thesis.

```
/* ---------------------------------------------- beginning of code ----------------------------------------------*/

model RM-ODP {
domain {ODP_Concepts}
state {

/* declaration of ODP concept categories (RM-ODP 2.5) */

  partition BasicInterpretationConcepts, BasicModellingConcepts, SpecificationConcepts : static
ODP_Concepts

/* declaration of "Basic interpretation concepts" (RM-ODP 2.6) */

partition UniverseOfDiscourse, InterpretationPossibilities  : static  BasicInterpretationConcepts
partition Entity, Proposition: UniverseOfDiscourse
partition FirstOrderProposition, HigherOrderProposition: Proposition
holds : Proposition -> UniverseOfDiscourse+
System : Entity
Sybsystem : System

/* introduction of relations between RM-ODP concept categories */

modeledByBMC : FirstOrderProposition -> BasicModellingConcepts
modeledBySC : HigherOrderProposition -> SpecificationConcepts
mappedToBMC : SpecificationConcepts -> BasicModellingConcepts
mappedToSC : BasicModellingConcepts -> SpecificationConcepts

/* declaration of "Basic modelling concepts" (RM-ODP 2.8) */

  partition Constitution,  SpaceTime, Information : static BasicModellingConcepts
  partition Object, Environment : static Constitution
  environment (~object) : Object! -> Environment!
  partition StructuralInfo, BehavioralInfo : static Information
  Behavior : BehavioralInfo
  State_ : StructuralInfo
  partition Action, BehavioralConstraint: static Behavior
  corresponding_constraint (~constrained_action) : Action -> BehavioralConstraint
```

```
  partition InternalAction, Interaction : static Action
  partition InteractionPoint, Space, Time : static SpaceTime
  Interface: Behavior
  Activity: Behavior
  LocationInSpace : Space
  space_within_interval : LocationInSpace -> Space+
  state_location(~corresponding_state) : State_! -> Space! /* introduced to link Information and
Space for the definition of LocationInSpace */
  LocationInTime : Time
  time_within_interval : LocationInTime -> Time+
  interface_at_interaction_point: InteractionPoint -> Interface
  space_location: InteractionPoint -> LocationInSpace!
  time_location: InteractionPoint -> LocationInTime!
  constitution_state: Constitution! -> State_
  object_state: Object! -> State_
  environment_state: Environment! -> State_
  potential_activity: State_ -> Activity+
  object_behavior: Object! -> Behavior!
  environment_behavior: Environment! -> Behavior!
  instant: Time -> Time!
  state_existence: Time! -> State_!
  constraining : Environment! -> BehavioralConstraint
  participant : Action -> Constitution
  participating_object : Action -> Object!
  instant_begin : Action -> Time!
  instant_end : Action -> Time!
  X : BasicModellingConcepts

/* declaration of "Specification concepts" (RM-ODP 2.9) */

  partition Type, Class, Instance, Composition, Decomposition: SpecificationConcepts
  TemplateType: Type
  Template, InstantiationRules : TemplateType
  Instantiation: Instance
  partition Creation, Introduction : Instantiation
  associated_type: Class! -> Type!
  member_of_class(~set_of): Instance+ -> Class!
  satisfies_type(~valid_for): Instance+ -> Type!
  TemplateClass: Class
  associated_template_type: TemplateClass! -> TemplateType!
  member_of_template_class(~set_of_instantiations): Instantiation+ -> TemplateClass!
  subtype(~supertype): Type -> Type
  subclass(~superclass): Class -> Class
  specification (~instantiation): Instantiation -> Template!
  derived_class(~base_class): TemplateClass -> TemplateClass
  incremental_modification: Template -> Template
  refinement: SpecificationConcepts -> SpecificationConcepts

/* declaration of "Specific Specification concepts" (RM-ODP 2.9) */
```

```
  Role, Invariant, Precondition, Postcondition : Type
  composite_object: Decomposition -> Object!
  interface_signature: Template -> Interface!

}

/* invariant for "Basic interpretation concepts" (RM-ODP 2.6) */

inv AssertOrDeny {
        all a: UniverseOfDiscourse, b: Proposition | (a in b.holds) || (a not in b.holds)
       }

/* invariant for "Basic modelling concepts" (RM-ODP 2.8) */

inv TimeDependance{
        all o: Object, t: Time  |  one t.instant ->one o.object_state
       }

/* definitions for "Basic interpretation concepts" (RM-ODP 2.6) */

def FirstOrderProposition {
all p: FirstOrderProposition | (p.holds: Entity)
       }
def HigherOrderProposition {
all p: HigherOrderProposition | (p.holds: Proposition)
       }

/* definitions for relations between RM-ODP concept categories  */

def mappedToBMC {
all bmc: BasicModellingConcepts, sc: SpecificationConcepts, fop: FirstOrderProposition, hop:
HigherOrderProposition | bmc in sc.mappedToBMC <-> (fop.holds=hop.holds.holds) &&
(fop.modeledByBMC = bmc) && (hop.modeledBySC = sc)
       }
def mappedToSC {
all bmc: BasicModellingConcepts, sc: SpecificationConcepts, fop: FirstOrderProposition, hop:
HigherOrderProposition | sc in bmc.mappedToSC <-> (fop.holds=hop.holds.holds) &&
(fop.modeledByBMC = bmc) && (hop.modeledBySC = sc)
       }

/* definitions for "Basic modelling concepts" (RM-ODP 2.8) */

def participant {
all a: Action, b: Constitution | b in a.participant <-> (a.instant_begin.state_existence in
b.constitution_state) && (a.instant_end.state_existence in b.constitution_state) /*pre & post
state are in the allowed states of the content;
introduced to characterize those entity models, which participate in an Action - to define Action
associated with an object, Internal Action and Interaction */
       }
def Action{
```

```
all a: Action |  (a.instant_begin != a.instant_end) && (a.instant_begin.state_existence != a.instant_end.state_existence) && (a.participating_object in a.participant) /* the last condition is here to show that there exist at least one object associated with an action*/
	}
def InternalAction {
all a: InternalAction | a.participating_object in a.participant -> a.participating_object.environment not in a.participant
	}
def Interaction {
all a: Interaction | a.participating_object in a.participant -> a.participating_object.environment in a.participant
	}
def Behavior {
all b: Behavior |  ((b in Action) && ( some b.corresponding_constraint) && ( b.corresponding_constraint in Behavior)) || ((b in BehavioralConstraint) && ( some b.constrained_action) && ( b.constrained_action in Behavior))
	}
def Interface {
all i: Interface | ((i in Interaction) && ( some i.corresponding_constraint) && ( i.corresponding_constraint in Interface)) || ((i in BehavioralConstraint) && ( some i.constrained_action) && ( i.constrained_action in Interface))
	}
def LocationInSpace {
all ls: LocationInSpace | some a | (a.instant_begin.state_existence.state_location in ls.space_within_interval) && (a.instant_end.state_existence.state_location in ls.space_within_interval)
	}
def LocationInTime {
all lt: LocationInTime | some a | (a.instant_begin in lt.time_within_interval) && (a.instant_end in lt.time_within_interval)
	}
def interface_at_interaction_point {
all ip: InteractionPoint, i: Interface | (i in ip.interface_at_interaction_point) <-> ((i.instant_begin.state_existence.state_location in ip.space_location.space_within_interval) && (i.instant_end.state_existence.state_location in ip.space_location.space_within_interval) && (i.instant_begin in ip.time_location.time_within_interval) && (i.instant_end in ip.time_location.time_within_interval))
	}
def InteractionPoint {
all ip: InteractionPoint | one ls: LocationInSpace | one lt: LocationInTime | ip.space_location = ls && ip.time_location = lt && some ip.interface_at_interaction_point
	}
def X {
all a: X | a.mappedToSC.mappedToBMC = a
	}

/* definitions for "Specification concepts" (RM-ODP 2.9) */

def Class {
```

```
 all c: Class | some i | one t | i.satisfies_type = t && i in c.set_of && i.member_of_class = c && c.associated_type = t
	}
def Instance {
all a: Instance | some t | a.satisfies_type = t
	}
def subtype {
all t1: Type, t2: Type | t1 in t2.subtype <-> ( t1.valid_for.satisfies_type=t2)
	}
def supertype{
all t1: Type, t2: Type | t2 in t1.supertype <-> ( t1.valid_for.satisfies_type=t2)
	}
def subclass {
all c1: Class, c2: Class | c1 in c2.subclass <-> ( c1.associated_type in c2.associated_type.subtype)
	}
def superclass {
all c1: Class, c2: Class | c2 in c1.superclass <-> ( c1.associated_type in c2.associated_type.subtype)
	}
def Template { /*recurrence is in RM-ODP: 2-9.11 references 2-9.13 and vice versa */
all tpl: Template |  tpl.instantiation.specification = tpl
	}
def associated_template_type {
all c: Class, t: Type, tc: TemplateClass, tt: TemplateType | ((c.associated_type = t) && (tc.associated_template_type = tt)) <-> ((tt = t) && (tc = c))
	}
def member_of_template_class {
all ii: Instantiation, tc: TemplateClass, i: Instance, c: Class | ((i.member_of_class = c) && (ii.member_of_template_class = tc)) <-> ((i = ii) && (tc = c))
	}
def derived_class {
all tc1: TemplateClass, tc2: TemplateClass | tc1.derived_class = tc2 <-> tc1.set_of_instantiations.specification.incremental_modification.instantiation.member_of_template_class = tc2
	}
def Composition {
all c: Composition | one a1:X | some a2:X | one a3:X  | a1+a2=a3 <-> (c = a1.mappedToSC) && (c = a2.mappedToSC)
	}
def Decomposition {
all d: Decomposition | one a1:X | some a2:X | one a3:X  | a1+a2=a3 <-> d = a3.mappedToSC
	}
def refinement {
all spec1: SpecificationConcepts, spec2: SpecificationConcepts | some d: SpecificationConcepts | (spec1.refinement = spec2) -> (spec1+d=spec2)
	}
def Type {
all t: Type | some x:X | x.mappedToSC = t
	}
```

```
/* definitions for "Specific Specification concepts" (RM-ODP 2.9) */

def composite_object {
all d: Decomposition | one o: Object | (d.composite_object = o) <-> ((d.mappedToBMC = o) &&
(o.mappedToSC = d))
	}
def interface_signature{
all t: Template, i: Interface, a: Interaction | (t.interface_signature = i) <-> a in i && t in
a.mappedToSC && a in t.mappedToBMC
	}
def Role{
all id: Role | one b:Behavior | b.mappedToSC = id
	}
def Invariant{
all i: Invariant | some o: Object | o.mappedToSC = i
	}
def Precondition{
all prec: Precondition | some a : Action | one s : State_ | a.mappedToSC = prec &&
a.instant_begin.state_existence = s
	}
def Postcondition{
all postc: Postcondition |some a : Action | one s : State_ | a.mappedToSC = postc &&
a.instant_end.state_existence = s
	}
/* Instantiate and Delition operations (RM-ODP 2.9) */

op Instantiate (i: Instantiation, bmc: BasicModellingConcepts, new_bmc:
BasicModellingConcepts', new_i: Instantiation', x:X'!, ix: Instantiation'!) {
x not in bmc
ix not in i
new_bmc = bmc + x
new_i = i + ix
x.mappedToSC' = ix
ix.mappedToBMC' = x
}
op Deletion (i: Instantiation, bmc: BasicModellingConcepts, new_bmc:
BasicModellingConcepts', new_i: Instantiation', x:X!, ix: Instantiation!) {
x in bmc
ix in i
x.mappedToSC = ix
ix.mappedToBMC = x
new_bmc = bmc - x
new_i = i - ix
x not in new_bmc
ix not in new_i
}
}
/*  ---------------------------------------------- end of code ----------------------------------------------*/
```

BIBLIOGRAPHY

[1] R. Audi (general editor). *The Cambridge Dictionary of Philosophy*, second edition; Cambridge University Press 1999.

[2] C. Bernardeschi, J. Dustzadeh, A. Fantechi, E. Najm, A. Nimour, and F. Olsen. "Transformations and Consistent Semantics for ODP Viewpoints". H. Bowman and J. Derrick, editors; *Proceedings of Second IFIP conference on Formal Methods for Open Object-based Distributed Systems - FMOODS'97*. Canterbery UK, Chapman & Hall, July 97.

[3] L. v. Bertalanffy. *General System Theory: foundations, development, applications*. George Braziller, New York, 1969, ISBN 0-8076-0453-4.

[4] J. Bezivin. "From Object Composition to Model Transformation with the MDA." *Proceedings of TOOLS*; USA, Santa Barbara, August 2001.

[5] J. Bezivin. "Who is afraid of ontologies?" *Proceedings of OOPSLA '98 Workshop: "Model Engineering, Methods and Tools Integration with CDIF"*; Vancouver, Canada, October 1998.

[6] G. S. Blair, J.-B. Stefani. *Open Distributed Processing and Multimedia*, Addison Wesley Longman Ltd, 1998.

[7] E. A. Boiten, H. Bowman, J. Derrick, P. F. Linington, and M. W. A. Steen. "Viewpoint consistency in ODP". *Computer Networks*, 34(3):503-537, August 2000.

[8] H. Bowman, E. A. Boiten, J. Derrick, M. W. A. Steen. "Viewpoint consistency in ODP, a general interpretation". *Proceedings of the First IFIP International Workshop on Formal Methods for Open Object-Based Distributed Systems* (editors: E. Najm and J.-B. Stefani), pages 189-204. Chapman & Hall, March 1996.

[9] H. Bowman, J. Derrick, P. Linington, M. Steen. "Cross-viewpoint consistency in Open Distributed Processing". *IEE Software Engineering Journal*, 11 (1): 1996, January 1996.

[10] H. Bowman, J. Derrick, P. Linington, M. Steen. "FDTs for ODP". *Computer Standards and Interfaces*, 17(1995):457-479, September 1995.

[11] G. Boolos, *Logic, Logic and Logic*. Harvard University Press, 1999.

[12] G. Bracha, J. Gosling, B. Joy, G. Steele. *The Java Language Specification*, Second Edition. Addison Wesley, June 2000, ISBN 0-201-31008-2.

[13] S. M. Brien, J. E. Nicholls. *Z Base Standard version 1.0*. Oxford University, Programming Research Group, Technical Monograph PRG-107, November 1992.

[14] D. DSouza. "*Model-Driven Architecture and Integration: Opportunities and Challenges*", Version 1.1, www.kinetiuym.com, February 2001.

[15] F. Durán and A. Vallecillo. "Writing ODP Enterprise Specifications in Maude". *Proceedings of ICEIS 2001, Workshop On Open Distributed Processing - WOODPECKER`2001, J. Cordeiro, H. Kilov (Eds.)*, Setúbal, Portugal, July 2001.

[16] H. Ehrig and B. Mahr. *Fundamentals of algebraic specification*. EATCS Monographs on Theoretical Computer Science, vol. 6, Springer-Verlag, 1985.

[17] A. Einstein, H. A. Lorentz, H. Minkowski, H. Weyl. "*Principle of Relativity*"; a collection of original memoirs on the special and general theory of relativity. New York, Dover Publications, 1952.

[18] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. "Inconsistency Handling in Multiperspective Specifications". *IEEE Transactions on Software Engineering*; 20, 8, pp 569-578 August 1994.

[19] ISO 9074. *Estelle, a Formal Description Technique Based on an Extended State Transition Model*. 1997.

[20] ISO/IEC 880. *LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behavior*. 1989.

[21] 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.

[22] ITU–T Recommendation Z.100. *CCITT Specification and Description Language (SDL)*. 1993.

[23] D. Jackson. "A Comparison of Object Modelling Notations: Alloy, UML and Z". *MIT Lab for Computer Science*. August 1999. http://sdg.lcs.mit.edu/~dnj/pubs/alloy-comparison.pdf

[24] D. Jackson. "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

[25] D. Jackson, Software Design Group. "The Alloy Constraint Analyzer. Online documentation: Short Guide to Alloy". http://sdg.lcs.mit.edu/alloy/docs/alloy-guide.html ; *MIT Laboratory for Computer Science*, Cambridge, MA.

[26] D. Johnson, H. Kilov. "An Approach to an RM-ODP Toolkit in Z". *Proceedings of the 1st Workshop on Component-Based Systems*. Zurich, Switzerland, 1997; in conjunction with European Software Engineering Conference (ESEC) and ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE), 1997.

[27] D. R. Johnson, H. Kilov. "Can a flat notation be used to specify an OO system: using Z to describe RM-ODP constructs". *In Proceedings of FMOODS96: IFIP WG 6.1 Conference on Formal Methods in Object-oriented Distributed Systems*; E. Najm, J-B. Stephani (editors), Chapman and Hall, Paris, March 1996, pp. 407-418.

[28] Lao Tsu. "*Tao Te Ching*". Translation by Gia-fu Feng and Jane English. Wildwood House 1991.

[29] P. F. Linington, J. Derrick, and H. Bowman. "The specification and conformance of ODP systems". *In 9th International Workshop on Testing of Communicating*

Systems, pages 93-114, IFIP TC6/WG6.1. Darmstadt, Germany, Chapman & Hall, September 1996.

[30] L. Logrippo, M. Faci, M. Haj-Hussein. "An Introduction to LOTOS: Learning by Examples". *Computer Networks and ISDN Systems*, 23: 325-342, 1992.

[31] A. N. Maliuta. *Hypercomplex Dynamic Systems*, Vischa Shkola, Lvov University Publishing, Lvov, 1989. ISBN 5-11-000571-0.
А. Н. Малюта. *Гиперкомплексные Динамические Системы*, Выща Школа, Изд-во при Львовском ун-те, Львов, 1991. ISBN 5-11-000571-0.

[32] A. N. Maliuta. *Invariant Modeling. Course of Lectures, International Academy "New Universum"*, CIC Severnaya Zvezda, Chernigov, 1999.
А. Н. Малюта. *Инвариантное Моделирование. Курс Лекций, Международная Академия «Новый Универсум»*, ЦГИ «Северная Звезда», Чернигов, 1999.

[33] A. N. Maliuta. *Regularities of Systems Development*, Naukova Dumka, Kiev, 1990. ISBN 5-12-001853-X.
А. Н. Малюта. *Закономерности Системного Развития*, Наукова Думка, Киев, 1990. ISBN 5-12-001853-X.

[34] A. N. Malyuta. *System of Activities*, Naukova Dumka, Kiev, 1991. ISBN 5-12-002607-9.
А. Н. Малюта. *Система Деятельности*, Наукова Думка, Киев, 1991. ISBN 5-12-002607-9.

[35] E. Najm and J.-B. Stefani. "A Formal Semantics for the ODP Computational Model". *Computer Networks and ISDN Systems*, 27:1305-1329, 1995.

[36] E. Najm and J.-B. Stefani. "Computational models for open distributed systems". H. Bowman and J. Derrick, editors, *Proc. of FMOODS'97*, Canterbury, 1997. Chapman &Hall.

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

[38] OMG. "*Meta-Object Facility (MOF) Specification, version 1.3*". OMG Document 2000-04-03. Object Management Group, Inc. http://www.omg.org/technology/documents/formal/mof.htm

[39] OMG. "*Model Driven Architecture (MDA) FAQ...*" Object Management Group, Inc. www.omg.org/mda/faq_mda.htm, July 2001.

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

[41] R. Poli, P. Simons (editors). "*Formal Ontology*", (Nijhoff International Philosophy Series, Vol 53). Kluwer Academic Publishers, Dordrecht 1996.

[42] J. R. Putman. *Architecting with RM-ODP*. Prentice Hall, 2001.

[43] B. Russell. "Mathematical logic as based on the theory of types". *American Journal of Mathematics*, 30, 1908, pp. 222-262.

[44] R. O. Sinnott, K. J. Turner. "Applying Formal Methods to Standard Development: The Open Distributed Processing Experience". *Computer Standards & Interfaces Journal*, volume 17, pages 615-630, 1995.

[45] R. O. Sinnott and K. J. Turner. "Specifying ODP Computational Objects in Z", *Proceedings of 1st International Workshop on Formal Methods for Open Object--Based Distributed Systems*, Paris, France, March 1996, pp. 375--390

[46] R. Soley and the OMG Staff Strategy Group. "*Model-Driven Architecture.*" White paper, Draft 3.2. http://www.omg.org/mda/papers.htm, November 2000.

[47] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks/Cole, 2000.

[48] J. M. Spivey. "The Z Notation, A Reference Manual". *International Series in Computer Science*, Second Edition, Prentice-Hall International, 1992.

[49] M. W. Steen and J. Derrick. "ODP Enterprise Viewpoint Specification". *Computer Standards and Interfaces*, 22(3):165-189, August 2000.

[50] A. Tarski. "*Logic, Semantics, Meta-mathematics.*" Oxford University Press, 1956.

[51] A. C. Varzi. "*Boundaries, Continuity, and Contact*". Blackwell Publishers Inc. 1997.