A Formal Foundation of the RM-ODP Conceptual Framework

Andrey Naumenko, Alain Wegmann

Institute for computer Communication and Applications,
Swiss Federal Institute of Technology – Lausanne.
EPFL-DSC-ICA, CH-1015 Lausanne, Switzerland
{Andrey.Naumenko, Alain.Wegmann}@epfl.ch

Abstract. This paper presents an approach for formalizing the Reference Model for Open Distributed Processing (RM-ODP), an ISO and ITU standard for the modeling of distributed system. The goals of this formalization are to clarify the RM-ODP modeling framework thus making it more accessible to modelers such as system architects, designers and implementers and to open the way for the formal verification of RM-ODP models (either within an ODP viewpoint or across multiple ODP viewpoints). RM-ODP officially declared as one of its goals to create a formal representation of Part 2: Foundations. The result of our work is a complete and truly consistent formal representation, until now non-existent, of clauses 5, 6, 8 and 9 of part 2 of RM-ODP in their inter-relations. Our formalization is based on set theory, Russell's theory of types and the classical predicate logic. The formalization is expressed in the Alloy language.

1 Introduction

The RM-ODP international standard [17] presents a useful architectural framework for modeling distributed systems. In our experience, at the present time not many modelers use the standard in their everyday practice. This is unfortunate, considering the amount of knowledge that has been invested in the project by highly qualified experts and the constructive potential that its results would bring to practice if they were adequately used. Particularly, one of the experienced problems is the difficulty in agreeing on a coherent interpretation of RM-ODP conceptual framework. We see a way to promote the use of RM-ODP in the formalization of its framework. This requires an attentive and painstaking translation of the standard definitions into formal logical constructions. This formalization would then allow for the creation of ODP-based software toolsets that could bring to modelers a more easily applicable version of the standard.

This work is done in the context of our research that targets the development and modeling of the evolution of complex systems. An example of what we consider a complex system is a group of companies, active in the same supply chain, which need to redefine their business models. This new definition leads to the redeployment of their technological infrastructure and to the development of component-based software applications. We use UML [30] for modeling the various organizational layers [11] present in complex systems (e.g. market, company, IT, components, etc…). Our experience in system modeling shows the importance of solid definitions for the fundamental modeling concepts we use. RM-ODP part 2 provides such definitions. As one of our goals is to influence UML and to develop ODP compatible methodologies and toolsets, we experienced the need for a more formal definition of RM-ODP.

RM-ODP introduces general terms that apply "*to any form of modeling activity*" ([17] part 2). To use ODP concepts for a concrete application, a modeler is supposed to choose a particular kind of semantics and modeling language ([17] part 2). This choice will necessarily define the limits for the concrete context of interest for modeling; thus automatically excluding from the model everything that is beyond these limits. Avoiding this problem, our goal is to present the ODP framework in a formal way while keeping its generic essence in relation to any particular applications. This formalization applied in a particular modeling process would give an easy-to-follow guideline for building complete and consistent system specifications, and allow for the formal verification of resulting models. The possibility of the verification would significantly facilitate localization of specification faults. We consider RM-ODP to be a standard, so we will not modify any of the ODP definitions in the process of their formalization.

In this paper we present the formalization of the conceptual kernel of RM-ODP that is Part 2: Foundations. So, as it is assured by the standard, the concept categories and their inter-relations contained in the resulting formal framework are sufficient for any modeling problems. This sufficiency guarantees the potential for practical applications of the formalization. The claim that the formalization has a practical value has been confirmed through our experience with the case studies on behavior modeling [2] where, based on the formalized version of RM-ODP, we have specified some of the behavioral constraints used for distributed systems modeling.

Partial results of this work were previously reported in ([29], [38]). This paper is the completion of our work.

EPFL-DSC/2001/040

In Section 2, we will explain the motivation behind our research. To provide the reading aid for our formal models, Section 3 will contain a summary of Alloy language. In Section 4, we will present the explanations that are necessary to introduce categorization of ODP concepts together with a discussion on “Basic interpretation concepts”. In Section 5, we will elaborate on the formalization of ODP “Basic modelling concepts” and provide supporting explications. Section 6 will be dedicated to the “Specification concepts” category. At the end of the paper the reader will find several concluding remarks and references to the bibliography that we used in our research.

2 Motivation Overview

2.1 Analysis of the RM-ODP Standard

Let us first position the paper with regard to the different parts of RM-ODP[1]. 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 confirm. Part 4 of RM-ODP describes the recommendations for approaching the standard formalization with LOTOS ([16], [25]), ACT ONE [13], SDL-92 [18], Z ([10], [36]) and ESTELLE [15] languages. Unfortunately, these are just recommendations or, in the best cases, small unrelated pieces of formalization presented with different formal techniques for the part 2 concepts. 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 [17]), 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 [31], 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 [31] leaves out the part 4 in its consideration. All of this provided us with excellent motivation for this research 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 was taken in part 4 hardly favors this kind of formalization. Detailed explanations of the approach may be found in [33]. 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 ([16], [25]). As a result, they are oriented towards the simulation of the execution of ODP models. Our goal is different; we present a formal meta-model of the standard that allows ODP models to be verified and checked for consistency. Our meta-model of RM-ODP part 2 stays on the same conceptual level as the UML meta-model [30] thus providing potential for it 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 meta-model, 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 (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.

[1] Each time when we mention RM-ODP in our paper, we refer to [17].

2.2 Analysis of Previous Research on 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, [8] presented a nice discussion on the requirements that the RM-ODP framework imposes on different Formal Description Techniques (FDTs) for its formal interpretation. [8] considered a set of general ODP requirements and elaborated on the requirements for specific ODP viewpoints. Further research ([6], [7], [24]) 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 [14]. In the case of RM-ODP, the viewpoints are well defined in the standard, - this allowed for publication of some successful case studies. For instance, studying the interrelations of the viewpoints [3] presents mappings between the information viewpoint and the computational viewpoint languages; [5] relates the computational viewpoint with the engineering viewpoint. At the same time another research thread concentrates only on formalizations for specific viewpoints. [12], [37] propose approaches for the enterprise specifications and [26], [27], [34] formalize the ODP computational model. Examples of approaches for computational, engineering and technology viewpoints can be found in [4]. 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. Particularly, [33], 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 ([22], [23]). Specifically, [22] 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.

2.3 RM-ODP Part 2: Scope for Formalization

As we explained the motivation for our work, 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 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 paper.

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 meta-model (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 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*”.

3 Formalization Language

The approach that we took in our RM-ODP formalization is essentially the classification of concepts, (with the aid of the set theory using Russell’s theory of types [32] and classical predicate logic [9]) and their interpretation with Alloy [20], (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*” [20]. 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 [19]). 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 our paper 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 the conceptual reasoning presented in the paper. 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 vision that we present in this paper, 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*” [21]. 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*” [21]. 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*” [21]. 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* *****.” [19]. Logical operators and words reserved for Alloy quantifiers and relations are [20]:

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 [19], [20], [21].

4 Categorization of ODP Concepts

"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 2.3 of our paper.

First of all, we stress the importance of a formal view on the categorization of ODP concepts. The corresponding standard clause (RM-ODP 2-5) 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 introduced in 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 [31], which in the very paragraph that emphasizes the importance of formalism for specifications ([31]: 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 [31], in the context of RM-ODP, tries to put entities and propositions about them within the model. There is confusion here because entities and propositions are defined 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. We will show the corresponding relation later in the paper.

As the consequence of the vagueness of clause 5, we have no choice but to formalize the relations (between the clauses 2-6, 2-8 and 2-9) that we find logical and that we understand to be implied by the standard.

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

4.1 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 modelling the universe of discourse;
- Atomicity (2-6.4), that allows for 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 meta-model 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 the Section 2.3 of our paper. Since here our goal is to formalize RM-ODP meta-model 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 Russell's theory of types [32] (that has individuals and propositions over individuals). [32] 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 RM-ODP clause 6 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 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
        Sybsystem : 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 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".

4.2 Introduction of Basic Modelling Concepts

"*Basic modelling concepts*" defined in part 2 clause 8 is the first conceptual category that allows for construction of 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.

For the beginning, let's assume that Russell's theory of types [32] may be applied to the model as well as it was applied to the universe of discourse in the previous section of our paper. 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 basic modelling concept. So 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, 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 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 modeling concepts within the model. And basic modeling concepts are essentially the first-order propositions about model elements.

Now, as we understood the positioning of basic modelling concepts within the model, we may define a correspondence between them and the universe of discourse that is modeled. We suggest 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 basic modelling concepts). The justification of this suggestion will be discussed further in Section 4.4.

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.

4.3 Introduction of Specification Concepts

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

The same reasoning that was used in the previous section for the basic modelling concepts and the first-order propositions may be successfully employed again, but now for the 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 the model specification concepts are essentially the higher-order propositions applied to the first-order propositions about the model elements. That is specification concepts may apply either to the basic modeling 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. We suggest 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 specification concepts). The justification of this suggestion will be discussed further in Section 4.4.

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 specification concepts in the model.

4.4 Relation between the Universe of Discourse and its Models

Three previous sections of our paper introduced one of the most important ideas of the RM-ODP modeling framework: 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 basic modelling and specification concepts). Let us review here 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 proposed formal vision of relations between the universe of discourse and its models is in agreement with basic principles of natural sciences (see [1] on "philosophy of science"). In natural 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. A science proposes the framework to model the subject of scientific investigation, analogously RM-ODP 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 RM-ODP we also have all the necessary parts of the method application. Specifically:

- entities characterized by propositions from the universe of discourse as well as model elements characterized by basic modelling concepts and specification concepts are “*the “primitive concepts” that can be grasped immediately without the use of definition*” [1].
- the decisions:
  - i. from Section 4.2: to model entities from the universe of discourse by means of model elements in the model
  - ii. from Section 4.2: 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 basic modelling concepts)
  - iii. from Section 4.3: 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 specification concepts)

  present us three axioms “*whose truth is knowable immediately without the use of deduction*” [1].

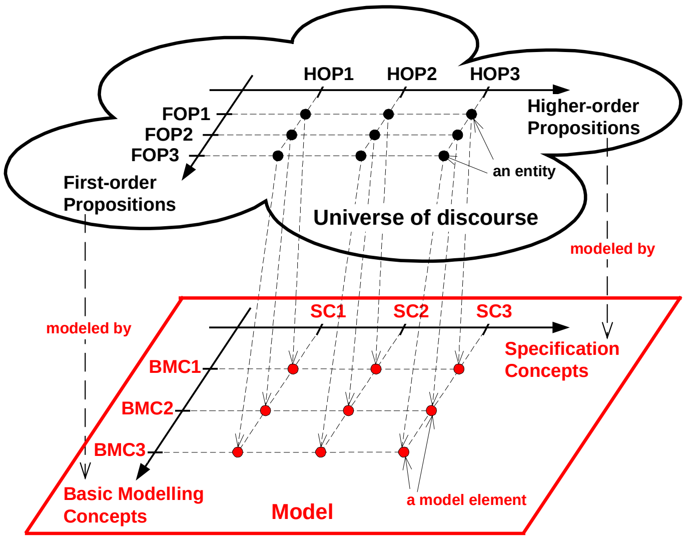

Fig. 1. Relation between the *entities*, *first-order propositions* (*FOP*) and *higher-order propositions* (*HOP*) from the *universe of discourse* and the corresponding *model elements*, *basic modeling concepts* (*BMC*) and *specification concepts* (*SC*) in the model.

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 1 by means of dashed arrows.

At this point we can return to the sections 4.2 and 4.3 where we promised to provide justifications for the suggestions (i), (ii) and (iii). Now it’s clear that these are axioms that we decided to postulate 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. Thus 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. Our chosen axioms allow for successful practical applications of the RM-ODP part 2 clauses 6, 8 and 9.

From our point of view, the relation between the universe of discourse and its model is important and should be mentioned explicitly in the RM-ODP categorization of concepts part.

4.5 Relation between Basic Modelling and Specification Concepts

In this section we would like to emphasize the independence of the basic modelling concepts and the specification concepts categories. Indeed, in part 2 clause 5 the RM-ODP standard defines them independently. That is basic modeling concepts can very well exist without any notion of 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 1).

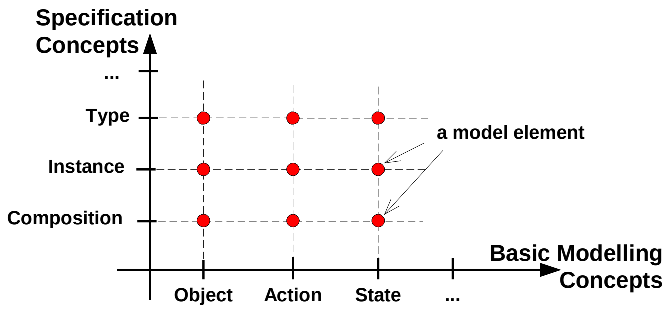

Fig. 2. *Set of model elements* as Cartesian product of the *basic modeling concepts set* and the *specification concepts set*.

The 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, specification concepts do not depend on the basic modelling concepts that they can characterize, and basic modelling concepts do not depend on the specification concepts that can characterize them. For example (see Figure 2), "Type" specification concept (RM-ODP 2-9.7) can be applied to any of the basic modelling 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[2] (such as "Composition", "Instance", "Type", etc).

Thus, the link between basic modeling concepts and specification concepts 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 modeling concept and a specification concept is equivalent to the concrete assertion (about a model element) that uses this basic modeling 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 with BasicModellingConcepts and vice versa within the context of the same model element corresponding to some entity from UniverseOfDiscourse.

[2] Sub-categorization of specification concepts into generic and specific specification concepts will be introduced in Section 6.

4.6 Model Elements

As we show on Figure 1, an entity from the universe of discourse is modeled by a model element in the model. And as we explained in sections 4.1-4.3, both entity and model element correspond to the Russell's definition for "*individual*" [32], 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 "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 "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 [9] 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 [35]).
- Second, to consider model elements being "*destitute of complexity*" is consistent with Russell's theory of types [32].
- 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 modeling 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 Basic Modelling Concepts

In this section we will formally describe 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.

5.1 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 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 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 space and time concepts, which define the corresponding SpaceTime category. Here 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.

Now, for our Alloy formalization, we can partition BasicModellingConcepts (that was introduced in Section 4.2) in these three categories (Constitution, SpaceTime and Information):

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

5.2 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. 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.

5.3 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. 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_[3] 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.

5.4 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 time or some 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

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

SpaceTime category. InteractionPoint should 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
```

5.5 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, Behavior is 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, 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!
```

5.6 Definition of Action–related Concepts

Now, after having declared all the necessary concepts we may proceed with their definitions in Alloy as it was explained in Section 3 of our paper. For the definition of the Action-related concepts we will need first to define an auxiliary concept of participant that was introduced in the previous subsection:

```
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))
}
```

5.7 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
}
```

6 Specification Concepts

In the previous section 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 4.3, 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 4.1, 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 modeling 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 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 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 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 4.3, 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, irrespective of its complexity, will be a single assertion about the model element.

6.1 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 4.3) 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 4.5 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)
}
```

6.2 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_template_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 our paper, 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: Instantia-
tion', 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: Instantia-
tion', 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
}
```

6.3 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)
}
```

6.4 Composition, Decomposition

The previous subsection 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. 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 modeling 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 modeling 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
}
```

6.5 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: SpecificationCon-
cepts
        }
```

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 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 5.6.

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 exist 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 6.2, 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 modeling 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 the Section 5.3, 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.

7 Concluding Remarks

We have presented an approach to the formalization of the RM-ODP standard with the aid of Alloy, the language for description of structural properties of a model. Our formalized structures corresponding to part 2 clauses 5, 6, 8 and 9 of RM-ODP can be used for modeling of ODP systems in practice. The resulting models can be verified with the aid of the Alloy Constraint Analyzer utility. This is the practical value that our work provides to any modeler who would be interested in trying a rigorous application of the RM-ODP standard.

The resulting Alloy model of RM-ODP conceptual framework can be found in the separate document: [28]. This model is a simple compilation of all the formal statements that were discussed in our paper. Thus, we can find there the parts corresponding to RM-ODP 2.5 (Categorization of concepts), 2.6 (Basic interpretation concepts), 2.8 (Basic modelling concepts) and 2.9 (Specification concepts).

Concluding our paper, we would like to mention that our results could serve as a foundation for further development of 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.

References

[1] R. Audi (general editor). “The Cambridge Dictionary of Philosophy”, second edition; Cambridge University Press 1999.

[2] P. Balabko. “From RM-ODP to the formal behavior representation”. To be submitted for the 10-th OOPSLA Workshop on Behavioral Semantics, OOPSLA 2001. Tampa Bay, USA; October 2001.

[3] 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.

[4] G. S. Blair, J.-B. Stefani. “Open Distributed Processing and Multimedia”, Addison Wesley Longman Ltd, 1998.

[5] 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.

[6] 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

[7] H. Bowman, J. Derrick, P. Linington, M. Steen. “Cross-viewpoint consistency in Open Distributed Processing”. IEE Software Engineering Journal 11 (1): 1996, January 1996.

[8] H. Bowman, J. Derrick, P. Linington, M. Steen. “FDTs for ODP”. *Computer Standards and Interfaces*, 17(1995):457-479, September 1995.

[9] Boolos, G.: “Logic, Logic and Logic”. Harvard University Press, 1999.

[10] S. M. Brien, J. E. Nicholls. “*Z Base Standard version 1.0*”. Oxford University, Programming Research Group, Technical Monograph PRG-107, November 1992.

[11] P. Checkland. “Systems Thinking, Systems Practice: Includes a 30-Year Retrospective”. John Wiley & Sons, September 1999.

[12] 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.

[13] H. Ehrig and B. Mahr. “Fundamentals of algebraic specification”. EATCS Monographs on Theoretical Computer Science, vol. 6, Springer-Verlag, 1985.

[14] Finkelstein, A.; Gabbay, D.; Hunter, A.; Kramer, J.; & Nuseibeh, B. “Inconsistency Handling in Multiperspective Specifications”. IEEE Transactions on Software Engineering; 20, 8, pp 569-578 August 1994.

[15] ISO 9074. “Estelle, a Formal Description Technique Based on an Extended State Transition Model”, 1997.

[16] ISO/IEC 880. “LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behavior”, 1989.

[17] 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”. 1995-96.

[18] ITU–T Recommendation Z.100. “CCITT Specification and Description Language (SDL)”, 1993.

[19] Jackson D.: “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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Logrippo L., Faci M., Haj-Hussein M.: “An Introduction to LOTOS: Learning by Examples”. Computer Networks and ISDN Systems, 23: 325-342, 1992.

[26] E. Najm and J.-B. Stefani. “A Formal Semantics for the ODP Computational Model”. Computer Networks and ISDN Systems, 27:1305-1329, 1995.

[27] 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.

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

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

[30] OMG. Unified Modeling Language Specification. Version 1.3, June 1999, http://www.omg.org/uml.

[31] J. R. Putman.: “Architecting with RM-ODP”, Prentice Hall, 2001.

[32] B. Russell, Mathematical logic as based on the theory of types, American Journal of Mathematics, 30, 1908, pp. 222-262.

[33] 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.

[34] 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

[35] J. F. Sowa. “Knowledge Representation: Logical, Philosophical, and Computational Foundations”, Brooks/Cole, 2000.

[36] J.M. Spivey. “The Z Notation, A Reference Manual”. International Series in Computer Science, Second Edition, Prentice-Hall International, 1992.

[37] M. W. Steen and J. Derrick. ODP Enterprise Viewpoint Specification. Computer Standards and Interfaces, 22(3):165-189, August 2000.

[38] A. Wegmann, A. Naumenko. "Conceptual Modeling of Complex Systems Using an RM-ODP Based Ontology". Proceedings of the 5th International Enterprise Distributed Object Computing Conference - EDOC 2001, Seattle, USA, September 2001.