Proposal for a formal foundation of RM-ODP concepts

Andrey Naumenko[1], Alain Wegmann[1], Guy Genilloud[2], William F. Frank[2]

[1]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
[2] Financial Systems Architect
53-102 East Shearwater Court, Jersey City NJ 07305, USA
{guy.genilloud, wff}@fsarch.com

Abstract. This paper presents an approach for formalizing the RM-ODP (Reference Model for Open Distributed Processing), an ISO and ITU standard. The goal of this formalization is to clarify the RM-ODP modeling framework to make it more accessible to modelers such as system architects, designers and implementers, while opening the way for the formal verification of RM-ODP models, either within an ODP viewpoint or across multiple ODP viewpoints. Our formalization is based on set theory and the usual predicate logic, and is expressed in the Alloy language.

1 Introduction

The RM-ODP international standard [5] presents a very good architectural framework for modeling distributed systems. In our experience, unfortunately at the present time not many modelers use the standard in their everyday practice. It's a pity, considering the amount of highly qualified experts' knowledge invested in the project and the big constructive potential that its results might bring to practice if they were adequately used. We see one of the ways to promote the use of RM-ODP in formalization of its framework. The formalization requires a careful and attentive translation of the standard definitions into formal logical constructions, but once done it would allow creation of ODP-based software toolsets that could bring to modelers an "easy to be applied" version of the standard.

This work is done in the context of our research that targets the development and evolution of complex systems. An example of what we consider as a complex system is a group of companies, active in the same supply chain, which need to redefine their business models leading to the redeployment of their technological infrastructure and to the development of component-based software applications. We use UML for modeling the various organizational layers [7] present in a complex system (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 our goal 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. To use ODP concepts for a concrete application a modeler is supposed to choose some particular kind of semantics and modeling language. The definition of any concrete context of interest for modeling will necessarily define the limits of the context, so will automatically exclude from the model everything that is beyond these limits. Our goal is to present the ODP framework in a formal way while keeping its generic essence in relation to any particular applications. Such a conceptual framework applied in a particular modeling process would give an easy to follow guideline for building complete and consistent system specifications, and allow formal verification of resulting models, which 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 show results that represent formalization of RM-ODP part 2 clauses 5, 6, 8. We plan to complete this work with the other ODP clauses in the near future, but nevertheless our current results can already be used in practice.

Let us position the paper with regard to the different parts of RM-ODP. Part 1 of the standard will not be considered for formalization since it contains a motivation overview of ODP and is not normative. The standard part 2 introduces ODP concepts and is the core of the formalization work that was performed. The work will be finalized in future and possibly continued with part 3 that presents the constraints to which ODP standards must confirm. It’s interesting to mention here the RM-ODP part 4, which describes the attempts to approach the standard formalization with LOTOS, ACT ONE, SDL-92, Z and ESTELLE languages. Particularly, the chapter 4.1 of the part 4 presents the standard architectural semantics in LOTOS [6] that is oriented towards the simulation of ODP models’ execution. Our goal is different; we present a formal meta-model of the standard that allows ODP models to be verified and checked for consistency, and stays on the same conceptual level as UML meta-model that might be potentially influenced by RM-ODP. Positioning itself as the standard meta-model, our approach presents a significant advantage in comparison with those described in the chapters 4.1, 4.2, 4.3, 4.4 and 4.5 of the part 4. Namely, having the goal to formalize definitions and mutual relations not only for the modeling concepts (basic modeling concepts and specification concepts) but for all the ODP concept categories, 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 will demonstrate the approach that we took in formalization, that was essentially classification of concepts with the aid of the set theory using regular predicate logic [1] and their interpretation with Alloy [4], 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” [4]. It is interesting that Alloy was developed having Z as starting point and adopting it for the object modeling (for the comparison of Alloy and Z see [3]). Z being the origin of Alloy was used in the standard part 4, - this gives an additional argument that supports our language choice. Another advantage of this choice is the public availability of the associated tool for checking specifications written in Alloy. Thanks to the tool, all the Alloy models presented in our paper can be tested and used in future research. We have to mention that the particular language choice does not impose any restriction on the conceptual reasoning presented in the paper. We just picked the one that helps for our presentation needs and we do not intend to discuss general advantages or disadvantages of the given language in comparison with others. Our framework can also be expressed in some other formal language such as, for example CTL* (Computation Tree Logic), a propositional temporal logic specification language [2].

In the Section 2 of the paper we will present the explanations that are necessary to introduce the categorization of ODP concepts together with discussion on Basic Interpretation concepts. Section 3 will elaborate the formalization of ODP Basic Modelling Concepts accompanied with supporting explications. Section 4 will contain the overall results of our work on RM-ODP formalization. Finally we will conclude with review of the most interesting points found in the formalization process and with overview of the future work towards the completion of the standard formalization. In the end of the paper we present the bibliography references and the appendix with Alloy grammar that can help reading our Alloy models.

2 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. In this section of our paper we would like to formalize the relations between some of these concept categories. We will construct the Alloy model introducing interrelated concept categories, later on they will be elaborated and will have their own Alloy models corresponding to the clauses definitions from RM-ODP.

2.1 Introduction of basic interpretation concepts

Basic interpretation concepts described in part 2 clause 6 define the universe of discourse being modeled (6.1, 6.2, 6.5) and introduce for a modeler possibilities of interpretation of the universe of discourse (6.3, 6.4, 6.6).

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

```
model RM-ODP { // corresponds to RM-ODP 2-5
domain {ODP_Concepts}
state {
        BasicInterpretationConcepts : ODP_Concepts
        partition UniverseOfDiscourse, InterpretationPossibilities  : BasicInterpreta-
tionConcepts

        // … to be completed with 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. Let us consider just the concepts related to the universe of discourse 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). Concept of system is defined as kind of entity and concept of subsystem as kind of system (definition 6.5). This allows presenting the domain that corresponds to the UniverseOfDiscourse in Alloy model for the BasicInterpretationConcepts:

```
model BasicInterpretationConcepts { // corresponds to RM-ODP 2-6
domain {UniverseOfDiscourse}
state {
        Entity : UniverseOfDiscourse
        Proposition : UniverseOfDiscourse
        holds : Proposition –> Entity+        //  "+" in Alloy stands for "one or more"
        doesNotHold : Proposition –> Entity+
        System : Entity
        Sybsystem : System
        }
inv Exclusion {
        no p | p.holds /in p.doesNotHold
        }
}
```

2.2 Introduction of basic modelling concepts

Basic modelling concepts defined in part 2 clause 8 introduce the means to be used by a modeler for describing the content of the universe of discourse being modeled. This content is defined by propositions about the entities of interest from the universe of discourse (see section 2.1). With the introduction of the basic modeling concepts category the RM-ODP model in Alloy will change as follows (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 RM-ODP { // corresponds to RM-ODP 2-5
domain {ODP_Concepts}
state {
        partition BasicInterpretationConcepts, BasicModellingConcepts :
ODP_Concepts
        partition UniverseOfDiscourse, InterpretationPossibilities : BasicInterpreta-
tionConcepts
        content : UniverseOfDiscourse -> BasicModellingConcepts

        // … to be completed with 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 content between UniverseOfDiscourse and BasicModellingConcepts. The last relation represents the fact that content of the universe of discourse has to be described by means of basic modeling concepts.

2.3 Introduction of specification concepts

Specification concepts as defined in part 2 clause 9 are not intrinsic to the ODP systems; they introduce the means to be used by a modeler for describing the qualities of content of the universe of discourse being modeled. These qualities are defined by propositions about the propositions describing the entities of interest from the universe of discourse. With the introduction of this category of concepts the RM-ODP model in Alloy will change as follows:

```
model RM-ODP { // corresponds to RM-ODP 2-5
domain {ODP_Concepts}
state {
        partition BasicInterpretationConcepts, BasicModellingConcepts, SpecificationConcepts : ODP_Concepts
        partition UniverseOfDiscourse, InterpretationPossibilities  : BasicInterpretationConcepts
        content : UniverseOfDiscourse –> BasicModellingConcepts
        contentQuality : UniverseOfDiscourse -> SpecificationConcepts

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

Which is to say that now we have yet another category in ODP_Concepts, called SpecificationConcepts, and that we can define a relation named contentQuality between UniverseOfDiscourse and SpecificationConcepts. The last relation represents the fact that qualities of content of the universe of discourse have to be described by means of specification concepts.

2.4 Relation between basic modelling and specification concepts

We would like to emphasize particularly the independence of two categories introduced in the sections 2.2 and 2.3. Indeed, in part 2 clause 5 RM-ODP standard defines them independently, that is basic modeling concepts can very well exist and be used 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).

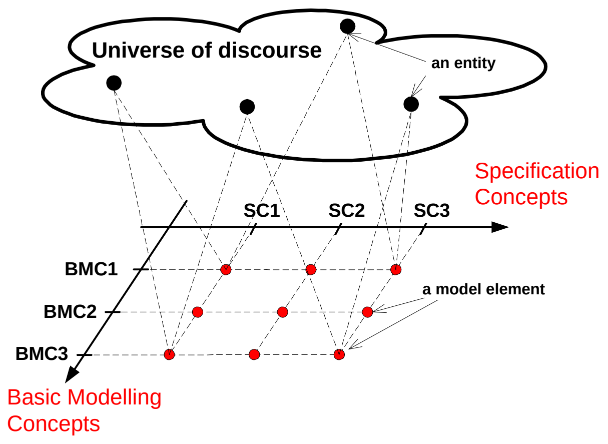

Fig. 1. Relation between the *entities* described by *propositions* from the *universe of discourse* and the corresponding *model elements* described by the corresponding *basic modeling concepts* (*BMC*) and *specification concepts* (*SC*).

Specification concepts defining qualities of content for the universe of discourse do not depend on the content to which they can be potentially applied, and the content of the universe of discourse being modeled by basic modeling concepts does not depend on the qualities that it can potentially have. Any result of modeling something in the universe of discourse (a model element) will have both characteristics: corresponding basic modeling concepts and corresponding specification concepts.

As we show on the Figure 1, an entity from the universe of discourse corresponds to a model element in the model. Essentially the content relation in the Alloy script defines correspondence of propositions that hold for the entities of interest found in the universe of discourse to the basic modeling concepts that characterize the model elements in the model. For example, a basic modeling concept such as “object” or “environment” characterizes a model element as the predicate for its being a “model of an entity”; “action” characterizes a model element for its being “something that happens”. In this case in the universe of discourse we will be able to find the propositions, which were modeled as these predicates, and the entities, which were represented by these model elements in the model.

As for the contentQuality relation in the Alloy script, it defines the specification concepts in the model to correspond to the propositions about the propositions describing the entities of interest from the universe of discourse. In the model this allows to have specification concepts as predicates about the predicates describing the model elements (in other words, as predicates about basic modeling concepts describing the model elements). For example, a specification concept such as “composition”, “type”, “class”, etc defines a predicate that can be applied to a basic modeling concept (such as “object”, “action”, “state”, etc) that describes a model element. In this case in the universe of discourse we will be able to find: the propositions, which were modeled as these predicates thus holding for other propositions; the propositions to which the former propositions were applied; and the entities characterized by the latter propositions and represented by the corresponding model elements in the model.

As we see, 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 be applied to the same model element. 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, Specifica-
tionConcepts : ODP_Concepts
        partition UniverseOfDiscourse, InterpretationPossibilities : BasicInterpreta-
tionConcepts
        content : UniverseOfDiscourse –> BasicModellingConcepts
        contentQuality : UniverseOfDiscourse –> SpecificationConcepts
        mappedToBMC : SpecificationConcepts -> BasicModellingConcepts
        mappedToSC : BasicModellingConcepts -> SpecificationConcepts

        // … to be completed with other concept categories
        }
// if there exist additional conditions, definitions of concepts are used in Alloy
// in addition to their declaration in the state part
def mappedToBMC {
        all a,b,c | b in a.mappedToBMC <-> ((c.contentQuality = a) && (c.content
= b))
        }
def mappedToSC {
        all a,b,c | a in b.mappedToSC <-> ((c.contentQuality = a) && (c.content =
b))
        }
}
```

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. In definitions variable “a“ corresponds to a specification concept, variable “b“ to a basic modelling concept and variable “c“ to an element from the universe of discourse.

3 Basic modelling concepts

In this section we will formally describe most of the basic modelling concepts 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 being explicitly stated these relations are often at risk of being overlooked in different practical interpretations of RM-ODP. And sometimes it happens that 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.

As defined by RM-ODP, the basic modeling concepts "*are concerned with existence and activity: the expression of what exists, where it is and what it does*". In other words with the aid of the basic modeling concepts we describe the situation when: "it is", "it is somewhere" and "it does something". Considering the concepts introduced by ODP we can map them to three basic categories that correspond to partitioning of quoted design goals for the basic modeling concepts. Among these three we should have a category responsible for the modeling of the content of the universe of discourse. This category would provide us with "it". Then we need a category for positioning "it" somewhere in time and space. And the last category would give us an information about what "it does" and how "it is". These categories will be presented as Alloy domains with names UODContent, SpaceTime, Information in our Alloy formalized version of the standard:

```
model BasicModellingConcepts { // corresponds to RM-ODP 2-8
domain {UODContent, SpaceTime, Information}
state {
// state is to be completed further in the paper.
}}
```

Let us consider UODContent concepts. They are defined for modeling of the content of the universe of discourse, particularly for entity modeling. In clauses 8.1 and 8.2 of the part 2 RM-ODP defines two particular kinds of concepts used for entity modeling in the universe of discourse: object and its environment. Corresponding to the definitions, having a model of the universe of discourse and pointing a particular entity in the universe of discourse, in the model we can have object, which would trace the entity, and environment of this object.

Formally put, the model of the universe of discourse 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, which is the model. 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 UODContent
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 of entity models that would be different from object and its environment. But it leaves a possibility of their existence by defining object being "*a model*" of an entity.

RM-ODP part 2 clause 8.1 defines an object to be characterized by behavior and dually by 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 two information aspects: in the first we can have essentially time-dependant concepts. Those are the concepts that can't be realized in a single instant in time, any of them would assume a time evolution at least for two different moments. Such as for the action concept that is defined as "something which happens" (RM-ODP 2-8.3), we can not tell anything about happening without having two instants: before and after the happening. Having 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 timeless information, such as state. Indeed a state can be defined only having one particular instant in time, for the other instant we will have a different state. The differentiation of state per instant in time gives us the possibility of modeling any kind of action, including those that do not influence the state. So, now we may introduce the corresponding changes to our Alloy model:

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

State_ is written with "_" symbol just to distinguish the ODP state concept from the "state" that is reserved as Alloy special term. We may now associate Object with its State_ and Behavior. In addition to this we may also associate Environment and UODContent that 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
uod_content_state: UODContent! –> 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 the existence of a particular object state will be declared later after the time introduction.

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 space within themselves. “Interaction point” is another concept representing location. It is linked by definition (RM-ODP 2-8.11) with the interface concept that will be introduced in a little while.

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

The introduction of time allows us to define all the time-dependent information concepts (those from BehavioralInfo category), as well as the complete 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 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, t  |  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 in Action and BehavioralConstraint. Producing an Action is the responsibility of the acting Object and constraining is the responsibility of its Environment. The BehavioralConstraint can be considered as a reaction of the environment of an object to the object action. As was already mentioned, ODP Action (RM-ODP 2-8.3) for its definition requires 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 : Action –> BehavioralConstraint
constraining : Environment! –> BehavioralConstraint
partition InternalAction, Interaction : static Action
participant : Action! –> UODContent
participating_object : Action! –> Object!
instant_begin : Action! –> Time!
instant_end : Action! –> Time!
```

Now, after having declared all the necessary concepts we may proceed with their definitions in Alloy. 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 paragraph:

```
def participant {
	all a,b | b in a.participant <–> (a.instant_begin.state_existence in
b.uod_content_state) && (a.instant_end.state_existence in b.uod_content_state)
}
```

That is to say, something from UODContent is defined as 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 UODContent under consideration.

Now we can define Action:

```
def Action{
	some a |  (a.instant_begin != a.instant_end) –>
(a.instant_begin.state_existence != a.instant_end.state_existence) &&
(a.participating_object in a.participant)
}
```

We can notice two parts in the Action definition. The first is state difference in the beginning of it (a.instant_begin) and in the end of it (a.instant_end) that reflects RM-ODP 2-8.3 definition statement that something should happen to be an action. And the second 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 {
	some a | a.participating_object in a.participant –>
a.participating_object.environment not in a.participant
}
def Interaction {
        some a | 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{
        some a, bc | a.corresponding_constraint = bc
}
```

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 in our definition of the Interface we will need to include two conditions: the first defining behavior and the second defining interaction.

```
def Interface {
        some a, bc | (a.corresponding_constraint = bc) && (a.participating_object in
a.participant –> a.participating_object.environment in a.participant)
}
```

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

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

And at last, for the InteractionPoint we will just need to condition it as a location (that is LocationInSpace and LocationInTime simultaneously), since its relation to the set of interfaces is already defined in the part of Alloy state declaration. Finally:

```
def InteractionPoint {
        some ls, lt | some a | (a.instant_begin.state_existence.state_location in
ls.space_within_interval) && (a.instant_end.state_existence.state_location in
ls.space_within_interval) && (a.instant_begin in lt.time_within_interval) &&
(a.instant_end in lt.time_within_interval)
}
```

4. Overall meta-model of RM-ODP in Alloy

In this section we present the overall result that we obtained while formalizing RM-ODP in the previous sections. We can see here the parts corresponding to RM-ODP 2.5 (Categorization of concepts), 2.6 (Basic interpretation concepts) and 2.8 (Basic modeling concepts). The last part is obtained by compiling all the statements introduced in the section 3.

```
// ** Categorization of concepts **

model RM-ODP { // corresponds to RM-ODP 2-5
domain {ODP_Concepts}
state {
        partition BasicInterpretationConcepts, BasicModellingConcepts, Specifica-
tionConcepts : ODP_Concepts
        partition UniverseOfDiscourse, InterpretationPossibilities : BasicInterpreta-
tionConcepts
        content : UniverseOfDiscourse –> BasicModellingConcepts
        contentQuality : UniverseOfDiscourse –> SpecificationConcepts
        mappedToBMC : SpecificationConcepts –> BasicModellingConcepts
        mappedToSC : BasicModellingConcepts –> SpecificationConcepts

        // … to be completed with other concept categories
        }
def mappedToBMC {
        all a,b,c | b in a.mappedToBMC <–> ((c.contentQuality = a) && (c.content =
b))
        }
def mappedToSC {
        all a,b,c | a in b.mappedToSC <–> ((c.contentQuality = a) && (c.content = b))
        }
}

// ** Basic interpretation concepts **

model BasicInterpretationConcepts { // corresponds to RM-ODP 2-6
domain {UniverseOfDiscourse}
state {
        Entity : UniverseOfDiscourse
        Proposition : UniverseOfDiscourse
        holds : Proposition –> Entity+          //  "+" in Alloy stands for "one or more"
        doesNotHold : Proposition –> Entity+
        System : Entity
        Sybsystem : System
        }
inv Exclusion {
        no p | p.holds /in p.doesNotHold
        }
}
```

```
// ** Basic modelling concepts **

model BasicModellingConcepts { // corresponds to RM-ODP 2-8
domain {UODContent,  SpaceTime, Information}
state {
partition Object, Environment : static UODContent
environment (~object) : Object! –> Environment!
partition StructuralInfo, BehavioralInfo : static Information
Behavior : BehavioralInfo
State_ : StructuralInfo
partition Action, BehavioralConstraint: static Behavior
corresponding_constraint : Action –> BehavioralConstraint
partition InternalAction, Interaction : static Action
partition Space, Time : SpaceTime
Interface: Behavior
Activity: Behavior
LocationInSpace : Space
space_within_interval : LocationInSpace! –> Space+
state_location(~corresponding_state) : State_! –> Space! // to link Information and
                                             // Space for the definition of LocationInSpace
LocationInTime : Time
time_within_interval : LocationInTime! –> Time+
InteractionPoint: SpaceTime
interface_at_interaction_point: InteractionPoint! –> Interface
uod_content_state: UODContent! –> State_
object_state: Object! -> State_
environment_state: Environment! –> State_
potential_activity: State_ –> Activity+
object_behavior: Object! –> Behavior!
environment_behavior: Environment! –> Behavior!
instant: Time –> Time!
state_existence: Time! –> State_!
constraining : Environment! –> BehavioralConstraint
participant : Action! –> UODContent
participating_object : Action! –> Object!
instant_begin : Action! –> Time!
instant_end : Action! –> Time!
}
inv TimeDependance{
            all o, t  |  one t.instant –>one o.object_state
}

def participant {
            all a,b | b in a.participant <–> (a.instant_begin.state_existence in
b.uod_content_state) && (a.instant_end.state_existence in b.uod_content_state)
}
def Action{
            some a |  (a.instant_begin != a.instant_end) –>
(a.instant_begin.state_existence != a.instant_end.state_existence) &&
(a.participating_object in a.participant)
}
def InternalAction {
            some a | a.participating_object in a.participant –>
a.participating_object.environment not in a.participant
}
def Interaction {
            some a | a.participating_object in a.participant –>
a.participating_object.environment in a.participant
}
def Behavior{
            some a, bc | a.corresponding_constraint = bc
}
def Interface {
            some a, bc | (a.corresponding_constraint = bc) && (a.participating_object in
a.participant –> a.participating_object.environment in a.participant)
}
def LocationInSpace {
            some ls | some a | (a.instant_begin.state_existence.state_location in
ls.space_within_interval) && (a.instant_end.state_existence.state_location in
ls.space_within_interval)
}
def LocationInTime {
            some lt | some a | (a.instant_begin in lt.time_within_interval) &&
(a.instant_end in lt.time_within_interval)
}
def InteractionPoint {
            some ls, lt | some a | (a.instant_begin.state_existence.state_location in
ls.space_within_interval) && (a.instant_end.state_existence.state_location in
ls.space_within_interval) && (a.instant_begin in lt.time_within_interval) &&
(a.instant_end in lt.time_within_interval)
}
}
```

5. Research Experience and Conclusions

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 RM-ODP part 2 clauses 5, 6 and 8 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 and motivated to try a rigorous application of the RM-ODP standard.

We believe that apart from having practical interest, our research also carries certain theoretical potential for explanation and promotion of realization of RM-ODP. Many interesting observations become clearer when looking at the formalized version that is presented. For example, dealing with ODP basic modeling concepts one can see that object state and behavior are considered as structural and behavioral parts of information. And fixing the information we will get time-evolution of an object's state. The state will evolve till the limit of non-determinism of the evolution caused by the lack of the information being fixed. Fixing the time would give just one value of structural information, the state of an object.

Another interesting observation is that in the RM-ODP standard the predicate definition for the environment of an object does not contain, and in fact can not contain, a concrete description of content for the environment, because content of the environment is defined by a concrete model. In the general case we cannot say anything concrete about an environment of an object, neither what it contains nor what it does not contain. We only can say that it is everything in a model that is not the corresponding object and generally does not has to, although can, represent a set of objects.

Also, the term of behavior is defined using the reference to an object of interest participating in the behavior (RM-ODP part 2 clause 8.6 "Behaviour (of an object)"). So, in this context an action contributing to the behavior of an object is also seen from the object-centric perspective, and in agreement with the action term definition may be either an internal action or an interaction. The action may only involve the object and, in the case of interaction, its environment. Both, the object and the environment participate in the behavior of the object contributing to its actions and constraints correspondingly. That is, defined from the object-centric perspective, an action is the responsibility of the object and it has its corresponding behavioral constraint, which is the environment's responsibility and which might be considered as a reaction of the environment for the object's action. Of course, the environment may include other objects that would have their contributions to the behavioral constraints corresponding to the object's actions.

Other facts, already clear even without a formalized version of RM-ODP, now become absolutely indubitable. For example, we can see that an object under any circumstances cannot be a part of its environment, since object and environment are defined as two complementary nonintersecting sets for entity modeling in the universe of discourse.

Another example is that interaction with itself is just a normal interaction and it cannot be an internal action under any circumstances because interaction and internal action are also defined as two complementary nonintersecting sets for action modeling. This means that if an object interacts with itself, then there should necessarily be an involvement of the object environment in this action, either by the simple mediation of the interaction or by provision of environmental constraints of whatever kind.

These and many other observations become simpler to realize when having in hand the presented formalized model. We think that this work may facilitate practical applications of RM-ODP, the standard that carries a big constructive potential that if properly used would eliminate the modeling errors and alert potential conceptual confusions of practitioners.

6 Future Work

Although the presented results can already be used in practical applications, more work needs to be done to complete the formalization of the RM-ODP standard. Particularly RM-ODP part 2 section 9 "Specification concepts" are essential for the model to be used (see Section 2.4). Formalization of this and other parts that are currently missing in the paper is the topic of our ongoing research work.

7 Acknowledgments

We thank John M. Donaldson from Geneva branch of Compaq for his review of the paper.

References

1. Boolos, G.: “Logic, Logic and Logic”. Harvard University Press, 1999.
2. Clarke E. M., Emerson E. A., Sistla A. P.: “Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications”. ACM Transactions on Programming Languages and Systems, 8(2): 244-263, 1986.
3. 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
4. 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
5. ISO/IEC 10746-1, 2, 3, 4 | ITU-T Recommendation X.901, X.902, X.903, X.904. « Open Distributed Processing - Reference Model”. OMG, 1995-96.
6. Logrippo L., Faci M., Haj-Hussein M.: “An Introduction to LOTOS: Learning by Examples”. Computer Networks and ISDN Systems, 23: 325-342, 1992.
7. Miller J. G. : “Living Systems”, McGraw-Hill Book Company, New York, 1978.

Appendix: Alloy Grammar

From [3,4]: 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 *****. Vertical bar denotes choice; angle brackets indicate optional phrases; *x,* indicates a comma-separated lists of *x*’s. *STAR*, *BAR* and *PRIME* are terminals representing an asterisk, vertical bar and prime mark respectively.

```
model ::= model model-name { domain {domdecl, *} para*}
para ::= state <name> { compdecl*}
/ inv <name> { formula* }
/ def comp { formula* }
/ cond name <arglist> { formula* }
/ assert <name> { formula*}
/ op name <arglist> { formula*}
domdecl ::= <fixed> set
compdecl ::= setdecl / reldecl
```

setdecl ::= <**disjoint / partition**> *set*, **:** <**fixed / static**> *set mult*
reldecl ::= *relx* <**(**~ *relx***)**> , **:** <**static**> *set mult* -> <**static**> *set mult*
relx ::= *rel* / *rel* **[***set***]**
mult ::= **? / ! / +**
arglist ::= **(** *argdecl,* **)**
argdecl ::= *arg,* **:** *set mult*
formula ::= *negate formula*
/ *formula logic-op formula*
/ *quantifier var-decl* , ***BAR*** *formula*
/ *expr comp-op expr*
/ *quantifier expr*
| **(** *formula* **)**
/ *name* <**(** *expr* , **)** >
var-decl ::= *var* , <**:** *expr*>
logic-op ::= **&& / || /** -> **/** <->
negate **::= not** | **!**
comp-op ::= **in /** = **/** *negate* **in /** *negate* = / /= **/** /**in**
quantifier ::= **all** | **some** | **no** | **sole** | **one**
expr ::= *var* / *arg* / *set* / *expr expr-op expr* / *expr* **.** *qualifier* / **{** *var-decl* ***BAR*** *formula* **}** |
(*expr* **)**
expr-op ::= **+ / - / &**
qualifier ::= *rel* / *rel* **[** *var* **] / ~** *qualifier* / **+** *qualifier* / ***STAR*** *qualifier*
arg ::= id
var ::= id
name ::= id
set ::= id / *id* ***PRIME***
rel ::= id / *id* ***PRIME***