A Semantics of Action Related Concepts in ODP Enterprise Language

Mohamed Bouhdadi, El Maati Chabbar, Youssef Balouki

***Abstract*—The Reference Model for Open Distributed Processing (RM-ODP) defines a framework for the development of Open Distributed Processing (ODP) systems in terms of five viewpoints. Each viewpoint language defines concepts and rules for specifying ODP systems from the corresponding viewpoint However the ODP viewpoint languages are abstract and do not show how these should be represented. We treat in this paper the need of formal notation for behavioral l concepts in the enterprise language. Using the Unified Modelling Language (UML)/OCL (Object Constraints Language) we define a formal semantics for a fragment of ODP behavior concepts defined in the RM-ODP foundations part and in the enterprise language. We mainly focus on time, action, sequentiality, non determinism, behavior constraints and permission, obligation and prohibition.**

***Index Terms*—RM-ODP, Enterprise Language, Behavior, Semantics, UML/OCL**

I. Introduction

The Reference Model for Open Distributed Processing (RM-ODP) [1-4] provides a framework within which support of distribution, networking and portability can be integrated. It consists of four parts. The foundations part [2] contains the definition of the concepts and analytical framework for normalized description of arbitrary distributed processing systems. These concepts are grouped in several categories which include structural and behavioral concepts. The architecture part [3] contains the specifications of the required characteristics that qualify distributed processing as open. It defines a framework comprising five viewpoints, five viewpoint languages, ODP functions and ODP transparencies. The five viewpoints are enterprise, information, computational, engineering and technology. Each viewpoint language defines concepts and rules for specifying ODP systems from the corresponding viewpoint. However, RM-ODP is a meta-norm [5] and can not be directly applicable. Indeed it defines a standard for the definition of other ODP standards. The ODP standards include modelling languages.

In this paper we treat the need of formal notation of ODP viewpoint languages. The languages Z, SDL,, LOTOS, and Esterel are used in RM-ODP architectural semantics part [4] for the specification of ODP concepts. However, no formal method is likely to be suitable for specifying every aspect of an ODP system.

Elsewhere, there had been an amount of research for applying the Unified Modelling Languages UML [6] as a notation for the definition of syntax of UML itself [7-9]. This is defined in terms of three views: the abstract syntax, well-formedness rules, and modeling elements semantics. The abstract syntax is expressed using a subset of UML static modelling notations. The well-formedness rules are expressed in Object Constrains Language OCL [10]. A part of UML meta-model has a precise semantics [11, 12] defined using denotational meta-modelling semantics approach. A denotational approach [13] is realized by a definition of the form of an instance of every language element and a set of rules which determine which instances are and are not denoted by a particular language element.

Furthermore, for testing ODP systems [2-3], the current testing techniques [14], [15] are not widely accepted and specially for the enterprise viewpoint specifications. A new approach for testing, namely agile programming [16], [17] or test first approach [18] is being increasingly adopted. The principle is the integration of the system model and the testing model using UML meta-modelling approach [19-20]. This approach is based on the executable UML [21]. In this context OCL can be used to specify the invariants [12] and the properties to be tested [17].

In this context we used the meta-modelling syntax and semantics approaches in the context of ODP systems. We used the meta-modelling approach to define syntax of a sub-language for the ODP QoS-aware enterprise viewpoint specifications [22]. We also defined a UML/OCL meta-model semantics for structural concepts in ODP computational language [23]. In this paper we use the same approach for behavioral concepts in the foundations part and in the enterprise language.

The paper is organized as follows. In Section 2, we define core behavior concepts (time, action, behavior, role, process). Section 3 describes behaviour concepts defined RM-ODP foundations part namely, time, and behavioural constrains. We focus on sequentiality, non determinism and concurrency constraints. In Section 4 we introduce the behaviour concepts defined in the enterprise language. We focus on behavioural policies. A conclusion ends the paper.

II. Core behavior concepts in RM-ODP foundation part

We consider the minimum set of modeling concepts necessary for behavior specification. There are a number of

Mohamed Bouhdadi, Department of Mathematics & Computer Science, University Mohammed V Rabat, Morocco, email: bouhdadi@ fsr.ac.ma).).
El maati Chabbar University Mohammed V Rabat, Morocco chabbar@fsr.ac.ma
Youssef Balouki, Department of Mathematics & Computer Science, University Mohammed V Morocco, email: balouki@cmr.gov.ma

approaches for specifying the behavior of distributed systems coming from people with different background and considering different aspects of behavior. We represent a concurrent system as a triple consisting of a set of states, a set of action and a set of behavior. Each behavior is modeled as a finite or infinite sequence of interchangeable states and actions. To describe this sequence there are mainly two approaches [24].

1. "Modeling systems by describing their set of actions and their behaviors".

2. "Modeling systems by describing their state spaces and their possible sequences of state changes".

These views are dual in the sense that an action can be understood to define state changes, and state occurring in state sequences can be understood as abstract representations of actions. We consider both of these approaches as abstraction of the more general approach based on RMODP. We provide the formal definition of this approach that expresses the duality of the two mentioned approaches.

We use the formalism of the RM-ODP model, written in UML/OCL. We mainly use concepts taken from the clause 8 "Basic modelling concepts" of the RM-ODP part 2. These concepts are: behavior, action, time, constraints and state (see figure 1). the latter are essentially the first-order propositions about model elements. We define concepts (type, instance, pre-condition, post-condition) from the clause 9 "Specification concepts". Specification concepts are the higher-order propositions applied to the first-order propositions about the model elements. Although basic modelling concepts and generic specification concepts are defined by RMODP as two independent conceptual categories [25].The behavior definition uses two RM-ODP modeling concepts: action and constraints. Behavior (of an object): "A collection of actions with a set of constraints on when they may occur". That is, a behavior consists of a set of actions, a set of constraints. An action is something which happens. RM-ODP does not give the precise definition of behavioral constraints. These are part of the system behavior and are associated with actions. This can be formally defined as follows:

Context c : constraint inv:
c.constrained_act -> size > 1

Context m :modelbehavior inv :
m.behavior->includesAll(m.Actions
->union(m.constraints))

For any element b from Behavior, b is an Action and b has a at least one constraint and this constraint is a Behavior element or b is a Constraint and b has a at least one action and this action is a Behavior element.

Context b :behavior inv :
m.behavior->forall(b |(m.actions->includes(m.b)
and b.constraints->notempty) or
(m.constraints->includes(m.b) and b.actions
->notempty)

To formalize the definition, we have to consider two other modeling concepts: time and state. We can see how these concepts are related with the concept of action by looking at their definitions. Time is introduced in the following way (RM-ODP, part 2, clause 8.10):

Location in time: An interval of arbitrary size in time at which action can occur."

instant_begin : each action has one time point when it starts instant_end : each action has one time point when it finishes

State (of an object) (RM-ODP, part 2, clause 8.7): At a given instant in time, the condition of an object that determines the set of all sequences of actions in which the object can take part. Hence, the concept of state is dual with the concept of action and these modeling concepts cannot be considered separately: This definition shows that state depends on time and is defined for an object for which it is specified.

Context t :time inv :
b.actions->exists (t1,t2| t1 =action.instant_beging
->notempty and
t2 =action.instant_end ->notempty and t1<> t2)

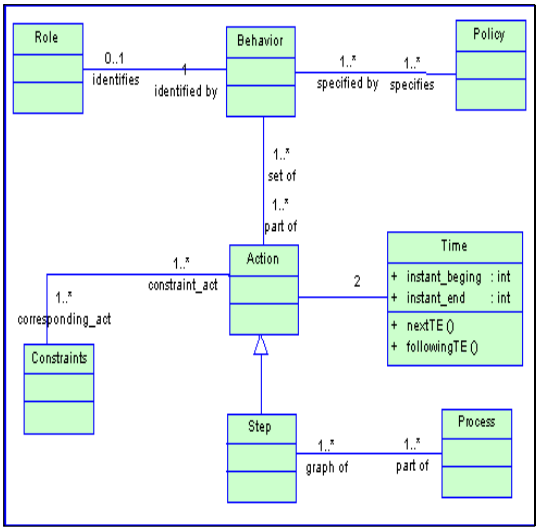


Figure 1. Core Behavior Concepts

III. META-MODELLING TIME AND BEHAVIORAL CONSTRAINTS

"Behavioral constraints may include sequentiality, non-determinism, concurrency, real time" (RM-ODP, part 2, clause 8.6). In this work we consider constraints of sequentiality, non-determinism and concurrency. The concept of constraints of sequentiality is related with the concept of time.

A. *Time*

Time has two important roles:

.•It serves for the purpose of synchronization of actions inside and between processes, the synchronization of a system with system users, the synchronization of user requirements with an actual performance of a system.

.•It defines sequences of events (action sequences)

To fulfill the first goal, we have to be able to measure time intervals. However, a precise clock that can be used for time measurement does not exist in practice but only in theory [26]. So the measurement of the time is always approximate. In this case we should not choose the most precise clocks, but ones that explain the investigated phenomena in the best way. Simultaneity of two events or their sequentiality, equality of two durations should be defined in the way that the formulation of the physical laws is the easiest" [26]. For example, for the actions synchronization, internal computer clocks can be used and, for the synchronization of user requirements, common clocks can be used that measure time in seconds, minutes and hours.

We consider the second role of time. According to [26] we can build some special kind of clock that can be used for specifying sequences of actions. RM-ODP confirms this idea by saying that "a location in space or time is defined relative to some suitable coordinate system" (RM_ODP, part 2, clause 8.10). The time coordinate system defines a clock used for system modelling. We define a time coordinate system as a set of time events. Each event can be used to specify the beginning or end of an action. A time coordinate system must have the following fundamental properties:

.•Time is always increasing. This means that time cannot have cycles.

.•Time is always relative. Any time moment is defined in relation to other time moments (next, previous or not related). This corresponds to the partial order defined for the set of time events.

We use the UML (fig1) and OCL to define time: Time is defined as a set of time events.

nextTE: defines the closest following time events for any time event

We use the followingTE relation to define the set of the following time events or transitive closure for the time event t over the nextTE relation:

followingTE: defines all possible following time events Using followingTE we can define the following invariant that defines the transitive closure and guarantees that time event sequences do not have loops:

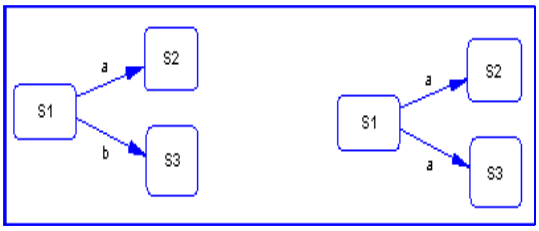
Context t :time inv :
Time->forAll(t:Time | (t.nextTE->isempty implies t.follwingTE->isempty) and (t.nextTE->notempty and t.follwingTE->isempty implies t.follwingTE =t.nextTE) and (t.nextTE->notempty and t.follwingTE->notempty implies t.follwingTE->
includes(t.nextTE.follwingTE->union(t.nextTE)) and t. follwingTE->exludes(t)).

This definition of time is used in the next section to define sequential constraints.

B. Behavioral constraints

We define the behavior like a finite state automaton (FSA). For example, figure 2 shows a specification that has constraints of sequentiality and non determinism. We can infer that the system is specified using constraints of non-determinism by looking at state S1 that has a non-deterministic choice between two actions a and b.

Based on RM-ODP, the definition of behavior must link a set of actions with the corresponding constraints. In the following we give definition of constraints of sequentiality, of concurrency and of non-determinism.



(a) (b)

Figure 2. a - Sequential deterministic constraints;
b - Sequential nondeterministic constraints.

1) Constraints of sequentiality

Each constraint of sequentiality should have the following properties [28]:

.•It is defined between two or more actions.

.•Sequentiality has to guarantee that one action is finished before the next one starts. Since RM-ODP uses the notion of time intervals it means that we have to guarantee that one time interval follows the other one:

Context sc :constraintseq inv :
Behavior.actions-> forAll(a1,a2 | a1<> a2 and a1.constraints->includes(sc)
and a2.constraints->includes(sc) and
((a1.instant_end.followingTE->includes(a2.instant_begin
)

or(a2.instant_end.followingTE->includes(a1.instant_begin)
)

For all SeqConstraints sc, there are two different actions a1, a2, sc is defined between a1 and a2 and a1 is before a2 or a2 is before a1.

2) Constraints of concurrency

Figure 3 shows a specification that has constraints of concurrency

We can infer that the system is specified using constraints of concurrency by looking at state S1 that has a simultaneous choice of two actions a2 and a3.

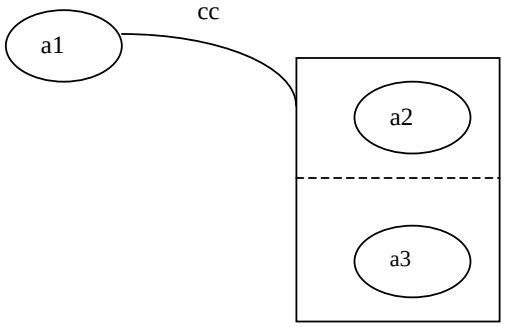


Figure 3. RM-ODP diagram: Example constraints of concurrency

For all concuConstraints cc there is a action a1, there are two different internal actions a2, a3, cc is defined between a1 and a2 and a3, a1 is before a2 and a1 is before a3

Context cc :constraintconc inv :
Behavior.actions-> forAll(a1 :Action ,a2 ,a3 : internalaction | (a1 <> a2) and (a2 <> a3) and (a3 <> a1) and a1.constraints->includes(cc) and a2.constraints
->includes(cc) and a3.constraints->includes(cc) and a1.instant_end.followingTE-> a2.instant_begin and a1.instant_end.followingTE-> a3.instant_begin

3) Constraints of non-determinism

In order to define constraints of non-determinism we consider the following definition given in [24]: "A system is called non-deterministic if it is likely to have shown number of different behavior, where the choice of the behavior cannot be influenced by its environment". This means that constraints of non-determinism should be defined between a minimum of three actions. The first action should precede the two following actions and these actions should be internal (see figure 4).

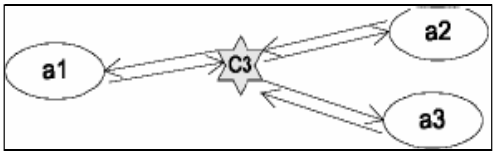


Figure 4. RM-ODP diagram: Example constraints of non-determinism

Context ndc: NonDetermConstraints inv :
Behavior.actions-> forAll(a1 :Action ,a2 ,a3 : internalaction | (a1 <> a2) and (a2 <> a3) and (a3 <> a1) and a1.constraints->includes(ndc) and a2.constraints
->includes(ndc) and a3.constraints->includes(ndc)
and a1.instant_end.followingTE-> a2.instant_begin or
a1.instant_end.followingTE-> a3.instant_begin)) .

We note that, since the choice of the behavior should not be influenced by environment, actions a2 and a3 have to be internal actions (not interactions). Otherwise the choice between actions would be the choice of environment.

IV. BEHAVIORAL POLICIES IN RM-ODP ENTERPRISE LANGUAGE

The enterprise specification is composed of specifications of the following elements : the system's communities (sets of enterprise objects), roles (identifiers of behavior), processes (sets of actions leading to an objective), policies (rules that govern the behavior and membership of communities to achieve an objective), and their relationships

The behavior of an ODP system is determined by the collection of all the possible actions in which the system (acting as an object), or any of its constituent objects, might take part, together with a set of constraints on when these actions can occur. In the enterprise language this is can be expressed in terms of roles or processes or both, policies, and the relationships between these. That is, behavior of an ODP system consists of a set of roles or a set of processes and a set of their policies. Constraints are defined for actions. Several authors have proposed different proprietary languages for expressing ODP policies, usually with formal support (e.g. Object-Z) but with no graphical syntax—hence losing one of the advantages of using UML. We propose modeling the enterprise viewpoint behavioral concepts using the standard UML diagrams and mechanisms for modeling behavior, since policies constrain the behavior of roles.

Context s :System inv :
s.behavior->(includesAll(s.Roles) or includesAll(s.Process)) ->union(s.Roles.policy))

Context o :object inv :
s.behavior-> includes(o.behavior.roles)
->-union(o.behavior.roles.policy)

We formalize in the following the concepts of policy. Policy is defined as a set of establishing , terminating and executing actions. figure 5 presents the UML meta-model for behavior and policy concepts. Establishing actions have to be defined by actions causing communications or process :
Establishing_act : set of actions which initialize a behavior
Terminating_act : set of actions which break some process
Executing_act : set of actions which execute a behavior or process

Context P : Policy inv :
P.specified_by -> size > 1

A. Obligation

To model obligations, we need to specify the actions that the system is forced to undertake as part of its intended behavior. In fact, an obligation is a prescription that a particular behaviour is required. It is fulfilled by the occurrence of the prescribed behaviour (clause :1 1 . 2 . 4). The actions must initiate by Establishing action, and to complete by the Terminating action .

Context po :policyobligation inv :
b.policy->includes(po) implies (Behavior.actions
-> (includes(self.Establishing_act) and
(Behavior.actions-> includes(self.Terminating_act)
and (Behavior.actions-> includes(self.Executin_act)

B. Permission

Permission is a prescription that a particular behavior is allowed to occur. A permission is equivalent to there being no obligation for the behavior not to occur (clause 1 1 . 2 . 5).

Permissions allow state transitions. Therefore, permission is expressed by a action Establishing_act which determine the scenario of the permitted action(s) and their participants, while its Terminating_act diagram describes the effects of such action(s).

Context pp :policypermission inv :
b.policy->includes(pp) implies (Behavior.actions)
-> (includes(self.Establishing_act) or (Behavior.actions
-> includes(self.Terminating_act)

C. *Prohibition*

A prohibition is prescription that a particular behaviour must not occur. A prohibition is equivalent to there being an obligation for the behaviour not to occur (clause1 1 . 2 . 6.) Prohibitions can be treated in two different ways, depending on their natures. The first way is to express them as conditional statements, explicitly banning action Establishing_act. In this way, the system will automatically prevent the prohibited action to happen. The second way to deal with prohibitions is by using watchdog rules again, which detect the occurrence of the prohibited action and execute the action Terminating_act , if possible.

Context ppr :policy Prohibition inv :
b.policy->includes(ppr) implies (Behavior.actions)
-> (excludes(self.Establishing_act)
and (Behavior.actions->excludes(self.Executing_act)
and includes(self.Terminating_act))

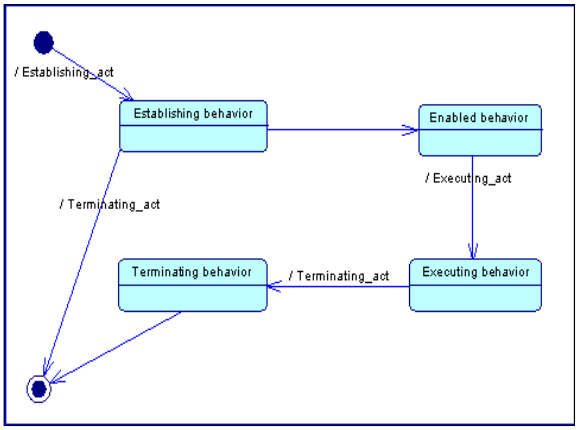


Figure 5. A meta-model for behavior and policy concepts

V. CONCLUSION

We address in this paper the need of formal ODP viewpoint languages. Using the meta-modeling semantics, we define a UML/OCL based semantics for a fragment of behavior concepts defined in the foundations part (time, sequentiality, non determinism and concurrency) and in the enterprise viewpoint language (behavioral policies). These concepts are suitable for describing and constraining the behavior of open distributed processing enterprise specifications. We are applying the same approach for other ODP enterprise behavior concepts (real time) and for behavior concepts in the computational language.

REFERENCES

[1] ISO/IEC, ‘’Basic Reference Model of Open Distributed Processing-Part1: Overview and Guide to Use, ‘’ISO/IEC CD 10746-1, 1994
[2] ISO/IEC, ‘’RM-ODP-Part2: Descriptive Model, ‘’ ISO/IEC DIS 10746-2, 1994.
[3] ISO/IEC, ‘’RM-ODP-Part3: Prescriptive Model, ‘’ ISO/IEC DIS 10746-3, 1994.
[4] ISO/IEC, ‘’RM-ODP-Part4: Architectural Semantics, ‘’ ISO/IEC DIS 10746-4, July 1994.
[5] M. Bouhdadi et al., ‘’An Informational Object Model for ODP Applications, ‘ Malaysian Journal of Computer Science, Vol. 13, N 2, (2000) 21-32.
[6] J. Rumbaugh et al., The Unified Modeling Language, Addison Wesley, 1999.
[7] B. Rumpe, ‘’A Note on Semantics with an Emphasis on UML, ‘’ Second ECOOP Workshop on Precise Behavioral Semantics, LNCS 1543, Springer, (1998) 167-188.
[8] A. Evans et al., ‘’Making UML precise, ‘’ Object Oriented Programming, Systems languages and Applications, (OOPSLA'98), Vancouver, Canada, ACM Press (1998)
[9] A. Evans et al. The UML as a Formal Modeling Notation, ‘’ UML, LNCS 1618, Springer, (1999) 349-274
[10] J. Warmer and A. Kleppe, The Object Constraint Language: Precise Modeling with UML, Addison Wesley, (1998).
[11] S. Kent, et al. ‘’A meta-model semantics for structural constraints in UML,, In H. Kilov, B. Rumpe, and I. Simmonds, editors, Behavioral specifications for businesses and systems, Kluwer , (1999). chapter 9
[12] E. Evans et al., Meta-Modeling Semantics of UML, In H. Kilov, B. Rumpe, and I. Simmonds, eds, Behavioral specifications for businesses and systems, Kluwer , (1999). ch. 4.
[13] D.A. Schmidt, ‘’Denotational semantics: A Methodology for anguage Development, ‘’ Allyn and Bacon, Massachusetts, (1986)
[14] G. Myers, ‘’The art of Software Testing, ‘’, John Wiley &Sons, (1979)
[15] Binder, R. ‘’ Testing Object Oriented Systems. Models. Patterns, and Tools, ‘’ Addison-Wesley, (1999)
[16] A. Cockburn, ‘’Agile Software Development. ‘’Addison-Wesley, (2002).
[17] B. Rumpe, ‘’ Agile Modeling with UML, ‘’ LNCS vol. 2941, Springer, (2004) 297-309.
[18] Beck K. Column on Test-First Approach. IEEE Software, Vol. 18, No. 5, (2001) 87-89
[19] L. Briand , ‘’A UML-based Approach to System testing, ‘’ LNCS Vol. 2185. Springer, (2001) 194-208,
[20] B. Rumpe, ‘’ Model-Based Testing of Object-Oriented Systems; ‘’ LNCS Vol.. 2852, Springer; (2003) 380-402.
[21] B. Rumpe, Executable Modeling UML. A Vision or a Nightmare?, In: Issues and Trends of Information technology management in Contemporary Associations, Seattle, Idea Group, London, (2002) pp. 697-701.
[22] M. Bouhdadi et al, ‘’ An UML-based Meta-language for the QoS-aware Enterprise Specification of Open Distributed Systems, ‘’ Collaborative Business EcoSystems and Virtual Enterprises, IFIP Series, vol. 85, Springer , (2002) pp. 255-264
[23] M. Bouhdadi, Y. Balouki, E. Chabbar. ‘’ Meta-Modeling Syntax and Semantics of Structural Concepts for Open Networked Enterprises”, ICCSA 2007, Kuala Lumpor, 26-29 August, LNCS Vol. 4707 45-54 2007.
[24] Broy, M., “Formal treatment of concurrency and time,‘’ in Software Engineer's Reference Book,J. McDermid, Editor, Oxford: Butterworth-Heinemann, (1991),
[25] Wegmann, A. et al. ‘’ Conceptual Modeling of Complex Systems Using RMODP Based Ontology‘’ . in 5th IEEE International Enterprise Distributed Object Computing Conference -EDOC (2001). September 4-7 USA. IEEE Computer Society pp. 200-211
[26] Henri Poincaré, The value of science, Moscow «Science», 1983
[27] Harel, D. and E. Gery, “Executable object modeling with statecharts“, IEEE Computer.30(7) pp. 31-42 (1997)
[28] P. Balabko, A. Wegmann, “From RM-ODP to the formal behavior representation” Proceedings of Tenth OOPSLA Workshop on Behavioral Semantics ¨Back to Basics¨, Tampa, Florida, USA , pp. 11-23 (2001).W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135