Verifying ODP Computational Behavioral Specification by using B-Method

Jalal Laassiri laassiri.jalal@gmail.com
Faculty of Science/Department of Mathematic
And Informatics/ Laboratory of Mathematic
and Informatics and Applications
Mohamed V University -Agdal
Rabat/ BP 1014/ Morocco

Saïd El Hajji elhajji@fsr.ac.ma
Faculty of Science/Department of Mathematic
and Informatics/ Laboratory of Mathematic and Informatics
and Applications
Mohamed V University -Agdal
Rabat/ BP 1014/ Morocco

Mohamed Bouhdadi bouhdadi@fsr.ac.ma
Faculty of Science/Department of Mathematic
and Informatics/ Laboratory of Mathematic
Pand Informatics and Applications
Mohamed V University -Agdal
Rabat/ BP 1014/ Morocco

Abstract

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 and specified. We treat in this paper the need of formal notation and specification for behavior al concepts in the Computational language. Using the Unified Modeling 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 Computational language. We mainly focus on time, action, behavior constraints (sequentiality, non determinism and concurrency constraints), and policies (permission, obligation, prohibition). We also give a mapping of the considered concepts to Event-B. This will permit the verification of such specifications. Finally we explore the benefits provided by the new extension mechanisms of B-Method for verifying the ODP computational specifications.

Keywords: RM-ODP, Computational Language, computational specifications, Behavior Semantics, UML/OCL, B-Method.

1. 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 Computational, 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] in the sense that it defines a standard for the definition of other ODP standards. The ODP standards include Modeling languages, specification languages and verification.
In this paper we treat the need of formal notation of ODP viewpoint languages. The languages Z [6], 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 Modeling Languages UML 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 Modeling 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-Modeling 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 especially for the Computational 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-Modeling 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-Modeling syntax and semantics approaches in the context of ODP systems. We used the meta-Modeling approach to define syntax of a sub-language for the ODP QoS-aware Computational viewpoint specifications [5]. We also defined a UML/OCL meta-model semantics for structural concepts in ODP computational language [22]. In this paper we use the same approach for behavior al concepts in the foundations part and in the Computational language. We also show how the ODP considered concepts could be specified in the Event-B method.
The paper is organized as follows. In Section 2, we define a meta-model semantics of core behavior concepts (time, action, behavior, role, process). Section 3 defines a meta-model semantics for behavior concepts of RM-ODP foundations part namely, time, and behavior al constraints. We focus on sequentiality, non determinism and concurrency constraints. In Section 4 we introduce the behavior concepts defined in the Computational language. We give precise definitions for behavior al policies. In section 5 overview the correspondence of the main concepts with the B-Method method constructs. A conclusion and perspectives end the paper.

2. Meta-Modeling Core Behavior Concepts in RM-ODP Foundations Part

We consider the minimum set of modeling concepts necessary for behavior specification. There are a number of approaches for specifying the behavior of distributed systems 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 [23]. 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 [24]. 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 mainly use concepts taken from the clause8 "Basic Modeling 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 Modeling concepts and generic specification concepts are defined by RM-ODP as two independent conceptual categories [25].
The behavior definition uses two RM-ODP modeling concepts: action and constraints (RM-ODP, part 2, clause 8.6):
Behavior (of an object): "A collection of actions with a set of constraints on when they may occur".
Action: "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 > 0
Context m: model behavior inv: m.behavior->includesAll(m.Actions->union(m.constraints))
For any element b from Behavior. "if b is an Action and has at least one constraint , this constraint is a Behavior element." Similarly when b is a Constraint and has at least one action, 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 [26].
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).

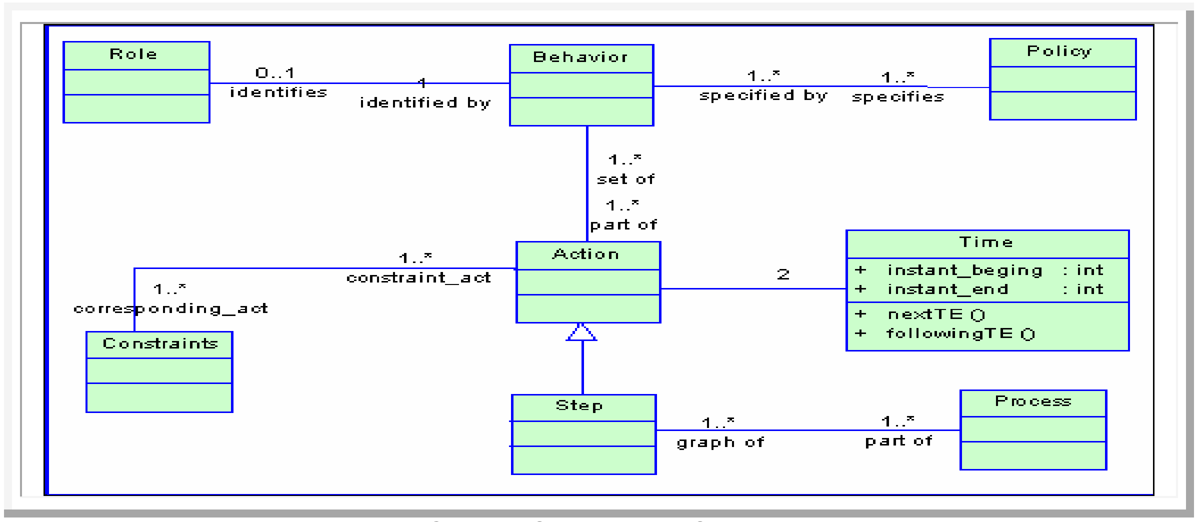


FIGURE 1: Core Behavior Concepts

3. Meta-Modeling 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.

3.1 Time

Time has two following important roles in system design [26]:

•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 fulfil 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 [27]. 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” [27]. 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 [27] 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 Modeling. 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 [26]:

•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 events [26].

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:

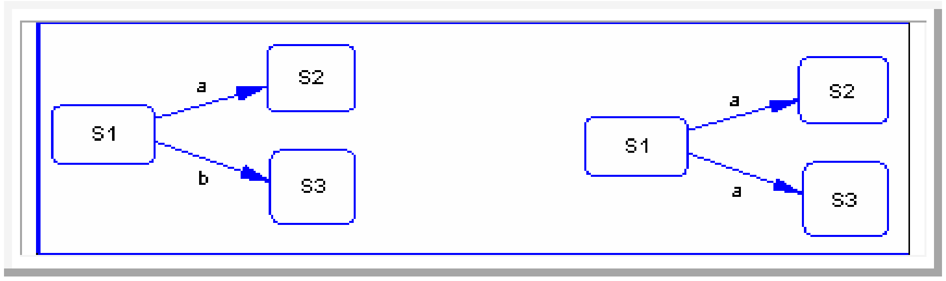
followingTE: defines all possible following time events Using followingTE we can define the following invariant that defines the transitive closure and guarantees that time event sequences do not have loops :

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.

3.2 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. The system is specified using constraints of non-determinism since state S1 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 non deterministic constraints.

3.2.1 Constraints of sequentiality

Each constraint of sequentiality should have the following properties [26]:
•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.

3.2.2 Constraints of concurrency

Figure 3 shows a system specification that has constraints of concurrency since state a1 has a simultaneous choice of two actions a2 and a3.

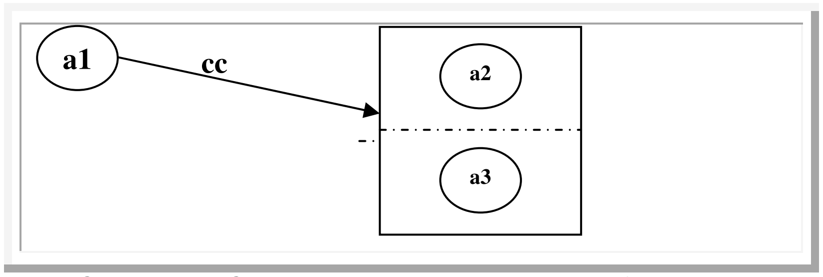


FIGURE 3: RM-ODP diagram: Example constraints of concurrency

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-> includes(a2.instant_begin) and a1.instant_end.followingTE-> includes(a3.instant_begin))

3.2.3 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).

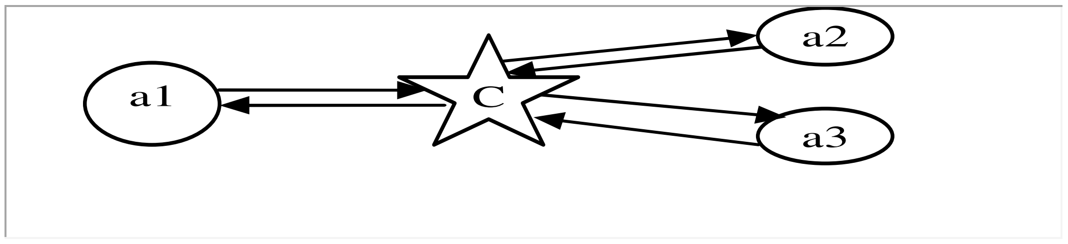


FIGURE 4: Example Constraints example of non-determinism

We define this constraint as follows:
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-> includes(a2.instant_begin) or a1.instant_end.followingTE-> includes(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 [26].

4. Modeling Behavior constraints Specifications in Event-B

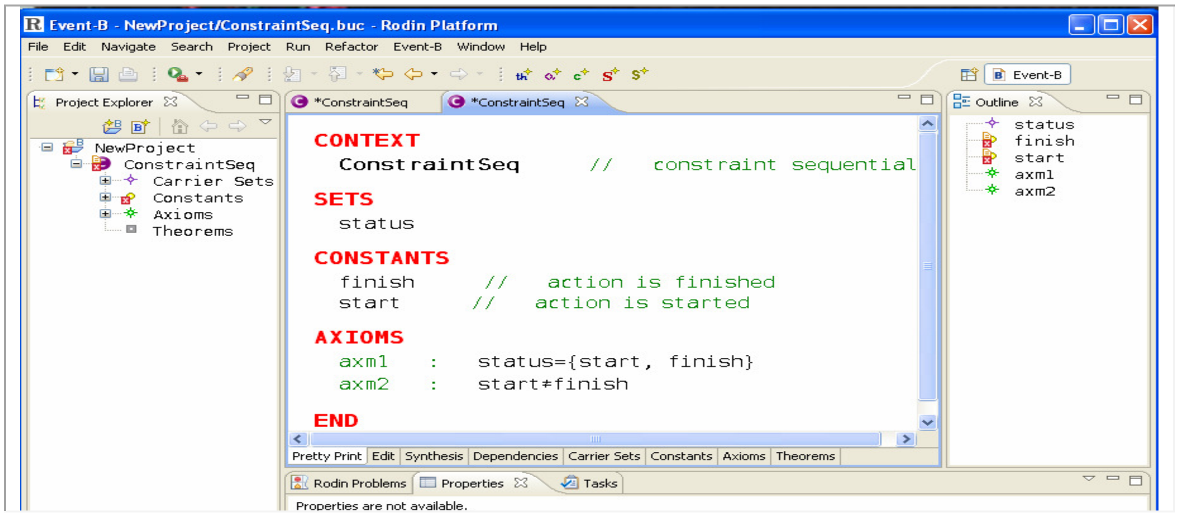
In this last section, we treat the question of verifying ODP specifications. For this we begin by defining how to use the formal method B-Method to specify the RM-ODP concepts. Event-B is a simplification as well as an extension of de B formalism [31] which has been used in number of large industrial projects. The objective of this formal method is use the refinement calculus to define and prove in the step by step fashion so that the system in question will be correct by construction. This will be very adequate in our context since each specification is a refinement of another. This will be done by using the propositional language, the predicate language, the set-theoretic language, and arithmetic language ,such they presents some mathematical justifications to proof obligation rules used in this approach.
In the previous section we specified the behavior constraints (Sequentiality, non-determinism, concurrent), here we presents how we can develop these concepts by using the Event-B and the tools of the open source RodinPlatform.
This section introduces a Event-B concepts which supports Modeling with a set of semantic constructs that correspond to those in behavior concepts, defined in enterprise language (see table 1).

Behavior Concepts	***Event-B Construct***
Behavior	***Machine***

State	***State static (constant with axioms) or State dynamic(variable with invariants)***
Action	***Event with guards(necessary conditions for event to occur)***
Constraint	***Invariants + guards***

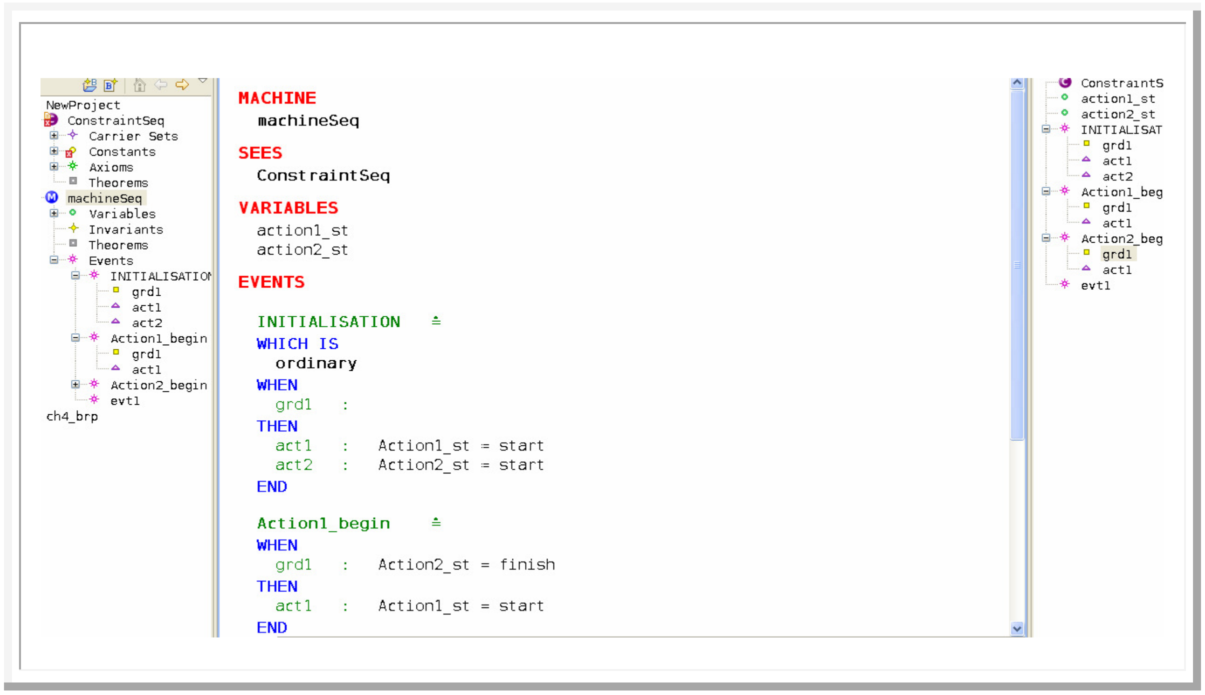
Table 1: T Sample table

We develop the initial model of the sequential constraint by both essentials construct of Event-B: machine and context.


Event-B - NewProject/ConstraintSeq.buc - Rodin Platform
File Edit Navigate Search Project Run Refactor Event-B Window Help
Event-B
Project Explorer
NewProject
ConstraintSeq
Carrier Sets
Constants
Axioms
Theorems
*ConstraintSeq
*ConstraintSeq
Outline
status
finish
start
axm1
axm2
Pretty Print Edit Synthesis Dependencies Carrier Sets Constants Axioms Theorems
Rodin Problems Properties Tasks
Properties are not available.

```
CONTEXT
  ConstraintSeq      //   constraint sequential
SETS
  status
CONSTANTS
  finish      //   action is finished
  start      //   action is started
AXIOMS
  axm1   :   status={start, finish}
  axm2   :   start≠finish
END
```

FIGURE 5: A context of sequential constraint



```
MACHINE
  machineSeq
SEES
  ConstraintSeq
VARIABLES
  action1_st
  action2_st
EVENTS

  INITIALISATION   ≙
  WHICH IS
    ordinary
  WHEN
    grd1   :
  THEN
    act1   :   Action1_st ≔ start
    act2   :   Action2_st ≔ start
  END

  Action1_begin    ≙
  WHEN
    grd1   :   Action2_st = finish
  THEN
    act1   :   Action1_st = start
  END
```

FIGURE 6: A machine of sequential constraint

5. CONSLUSION & FUTURE WORK

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 Computational viewpoint language (behavioral policies). These concepts are suitable for describing and constraining the behavior of open distributed processing Computational specifications.
The initial model of sequential constraint is developed by using Event-B, Each model will be analyzed and proved to be correct. The next step is the refinement of this model. We are applying the same approach for other ODP Computational behavior concepts (real time).

6. 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., ''A UML-Based Meta-language for the QoS-aware Enterprise Specification of Open Distributed Systems'' IFIP Series, Vol 85, Springer, 255-264 (2002).
6. Abhishek Dixit and al. "Applying UML and Z to Extended Basic Interoperability Data Model", International Journal of computer science and security (IJCSS), June 2007.
7. B. Rumpe, ''A Note on Semantics with an Emphasis on UML, '' Second ECOOP Workshop on Precise Behavioral Semantics, LNCS 1543, Springer, 167-188 (1998).
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, 349-274 (1999)
10. J. Warmer and A. Kleppe, the Object Constraint Language: Precise Modeling with UML, Addison Wesley, (1998).

11. S. Kent, and 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 and 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 Language 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, 297-309 (2004).
18. Beck K. Column on Test-First Approach. IEEE Software, Vol. 18, No. 5, 87-89 (2001)
19. L. Briand, ''A UML-based Approach to System testing, '' LNCS Vol. 2185. Springer, 194-208 (2001).
20. B. Rumpe, '' Model-Based Testing of Object-Oriented Systems; '' LNCS Vol.. 2852, Springer; 380-402 (2003).
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, 697-701 (2002).
22. 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 4707, Springer, 45-54 (2007)
23. Lamport, L. and N.A. Lynch, Distributed Computing: Models and Methods, in Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics. 1990, Elsevier and MIT Press.
24. Broy, M., "Formal treatment of concurrency and time,'' in Software Engineer's Reference Book,J. McDermid, Editor, Oxford: Butterworth-Heinemann pp 23, (1991).
25. Wegmann, A. and 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. 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).
27. Henri Poincaré, The value of science, Moscow «Science», 1983
28. Harel, D. and E. Gery, "Executable object modeling with statecharts", IEEE Computer.30(7) pp. 31-42 (1997)
29. Jean-Raymond Abrial: A System Development Process with Event-B and the Rodin Platform. ICFEM (2007) 1-3.
30. A.R.M Nordin and al. Managing Software Change Request Process: Temporal Data Approach,. International Journal of Computer Science and Security, (IJCSS) Volume (3):January 01, 2009.