Modeling ODP Policies by using event-B

Belhaj Hafid, Bouhdadi Mohamed and Elhajji Said

Department of Mathematics & Computer Science, University Mohammed V, Faculty of science
BP 1014 RP, 4. Av Ibn Batouta – Agdal, Rabat, Morocco

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: information, enterprise, computational, technology and engineering. Each viewpoint language defines concepts and rules for specifying ODP systems from the corresponding viewpoint. The enterprise viewpoint focuses on the roles and policies on the enterprise that the system is meant to support.
The use of formal methods in the design process of ODP systems is explicitly required. Formal notations provide precise and unambiguous system specifications. An important point to take into account is the incorporation of the many proofs which have to be performed in order to be sure that the final system will be indeed “correct by construction”. The Event-B method is being defined as a formal notation.
In this paper, we explore the benefits provided by using the proof construction approach to specify open distributed System in the enterprise viewpoint focusing on the specification of actions and the behavioral policies conditioning them.
Keywords: *RM-ODP, Enterprise Language, Policies, event B, RODIN platform.*

1. Introduction

The RM-ODP[1,2,3,4] framework is increasingly being used for modelling complex open distributed systems, such as those in the domains of telecommunications, finance, education and defence. While some of the ODP viewpoint languages, in particular computational and engineering are developed in sufficient details to describe programming and infrastructure artifacts of any distributed system, this is not true of the enterprise language. On the other hand, the availability of maturing distributed infrastructure platforms such as CORBA, DCOM, DCE and Java-RMI increasingly encourages the use of distributed objects for business applications. As a result, the IT community is shifting its interest from platform issues towards enterprise specifications. There is an increasing demand from industry to use enterprise specifications to improve the accuracy of the design of distributed systems, in particular those that cross various administrative and organisational boundaries [23] .
The existing ODP enterprise language, consisting of a limited number of concepts and structuring rules, needs further extensions and refinements in order to be better suited for enterprise modelling of practical open distributed systems. For example, there is a need for rigorous specification of policies governing the behaviour of complex systems and automated sub-systems. These policies need to be made explicit because their monitoring and enforcement will require actions by the system implemented, and the correctness of these actions can only be guaranteed if there is a well defined framework for the description of concepts such as ownership, right, objective, authority, delegation and policy.
The languages Z, SDL, LOTOS, and Estelle 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, we used the meta-modeling approach [5] [6] to define syntax of a sub-language for the ODP QoS-aware enterprise viewpoint specifications. We defined a meta-model semantics for structural constraints on ODP enterprise language [7] using UML and OCL. We also used the same meta-modeling and denotation approaches for behavioral concepts in the foundations part and in the enterprise language [8,9].
Furthermore, for modeling ODP systems correctly by construction, the current testing techniques [10,11] are not widely accepted. In a previous work [12,14], we specify the trading function and the protocol of negotiating QoS requirements between enterprise objects in event B.
For modeling business requirements and systems we will use the concepts provided by the RM-ODP enterprise viewpoint [15]. The enterprise viewpoint focuses on the purpose, scope and policies for the system and its environment. It describes the business requirements and how to meet them, but without having to worry about other system considerations, such as particular details of its implementation, or the technology used to implement the system.
Specifically, this paper focuses on a subset of the enterprise concepts, namely on the notions of action and policy (permissions, prohibitions and obligations), and try

to provide a more precise framework for reasoning about these fundamental enterprise concepts. It aims to allow the unambiguous specification of enterprise requirements.
we use the event-B formalism as our formal framework for developing policies in enterprise viewpoint ODP language. Event B [16] is a method with tool support for applying systems in the B method. Hence we can benefit from the useful formalism for reasoning about distributed systems given by refinement techniques and from the tool support in B. The Rodin Platform for Event-B provides effective support for refinement and mathematical proof. [17]
The structure of this document is as follows. First, Sections 2 and 3 serve as brief introductions to the RM-ODP and event B, respectively. Then, Section 4 describes our proposal to write enterprise policies specifications in event B. Finally, Section 5 draws some conclusions and describes some future research activities.

2. RM-ODP enterprise language

2.1 RM-ODP

Distributed systems are inherently complex, and their complete specifications are so extensive that fully comprehending all their aspects is a difficult task. To deal with this complexity, system specifications are usually decomposed through a process of separation of concerns to produce a set of complementary specifications, each one dealing with a specific aspect of the system. Specification decomposition is a well-known concept that can be found in many architectures for distributed systems. In particular, 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] in the sense that it defines a standard for the definition of other ODP standards. The ODP standards include modelling languages, specification languages and verification[12, 13].

2.2 The Enterprise Viewpoint

RM-ODP [1-4] provides five generic and complementary viewpoints on the system and its environment: enterprise, information, computational, engineering and technology viewpoints. They enable different abstraction viewpoints, allowing participants to observe a system from different suitable perspectives [18].
The enterprise viewpoint focuses on the purpose, scope and policies for the system [15] and its environment. It describes the business requirements and how to meet them, but without having to worry about other system considerations, such as particular details of its implementation, or the technology used to implement the system. Bellow, we summarize the basic enterprise concepts.
Community is the key enterprise concept. It is defined as a configuration of enterprise objects formed to meet an objective. The objective is expressed as a contract that specifies how the objective can be meet[19].
A contract is a generic concept that specifies an agreement governing part of the collective behavior of a set of objects. A contract specifies obligations, permissions and prohibitions for objects involved.
The scope of the system is defined in terms of its intended behavior, and this is expressed in terms of roles, processes, policies, and their relationships.
Roles identify abstractions of the community behavior, and are fulfilled by enterprise objects in the community.
Processes describe the community behavior by means of (partially ordered) sets of actions, which are related to achieving some particular sub-objective within the community.
Finally, policies are the rules that constrain the behavior and membership of communities in order to achieve their objectives. A policy can be expressed as an obligation, a permission, or a prohibition.
Obligation: A prescription that a particular behaviour is required. An obligation is fulfilled by the occurrence of the prescribed behaviour (RM-ODP, part 2, clause 11.2.4) .
Permission: A prescription that a particular behaviour is allowed to occur. A permission is equivalent to there being no obligation for the behaviour not to occur (RM-ODP, part 2, clause 11.2.5) .
Prohibition: A prescription that a particular behaviour must not occur. A prohibition is equivalent to there being an obligation for the behaviour not to occur (RM-ODP, part 2, clause 11.2.6) .
In general, ODP systems are modeled in terms of objects. An object is a model of an entity; it contains information and offers services. A system is therefore composed of interacting objects. In the case of the enterprise viewpoint we talk about enterprise objects, which model the entities defined in an enterprise specification [20].

An enterprise object is an object that filles one or more roles in a community. It can also participate in more than one community at one time. An enterprise object may be a role, an activity or a policy of the system. [19]

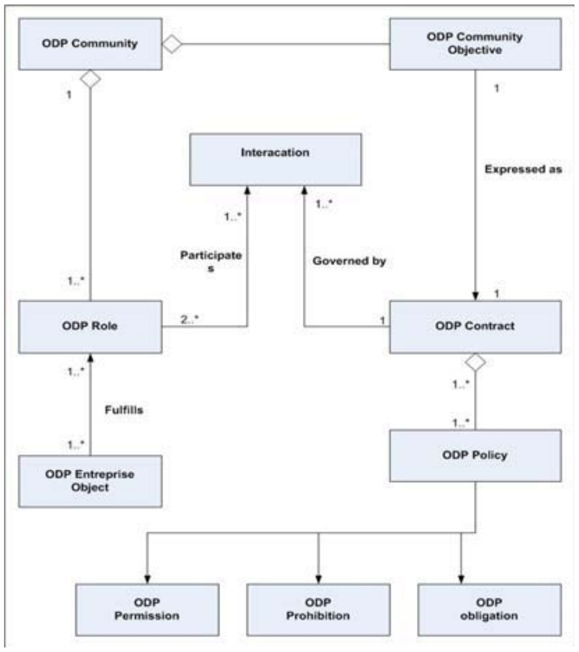

Fig. 1 Enterprise concepts [19].

3. Event B modeling approach

The Event-B [21] [22] is formal techniques consist of describing rigorously the problem, introduce solutions or details in the refinement steps to obtain more concrete specifications and verifying that proposed solutions are correct. The system is modeled in terms of an abstract state space using variables with set theoretic types and the events that modify state variables. Event-B, a variant of B, was designed for developing distributed systems. In Event-B, the events consist of guarded actions occurring spontaneously rather than being invoked. The invariants state properties that must be satisfied by the variables and maintained by the activation of the events.
The mathematical foundations for development of event based system in B is discussed in [13]. An abstract machine consists of sets, constants and variables clause modelled as set theoretic constructs. The invariants and properties are defined as first order predicates. The event system is defined by its state and contain number strained by the conditions defined in the properties and invariant clause known as invariant properties of the system. Each event in the abstract model is composed of a guard and an action. A typical abstract machine may be outlined as below.

```
    MACHINE              M
    SETS                 S1,S2,S3...
    CONSTANTS            C
    PROPERTIES           P
    VARIABLES            v1,v2,v3...
    INVARIANTS           I
    INITIALISATION       init
    EVENTS
          E1 = WHEN   G1 THEN   S1    END;
END.
```

4. Specifying ODP policies by using event B

4.1 Abstract and concrete levels on enterprise concepts

The interaction of people with IT systems generate various restriction needs to guarantee that each system user benefits of its advantages without trespassing on another user's rights. These needs vary according to the activity field required.
It could be regarding: Confidentiality (Non disclosure of sensitive information to non authorised persons), Integrity (Non alteration of sensitive information), Availability (Supply of information to users according to their rights of access these information), Auditability (The ability to trace and determine the actions carried out in the system).
Such requirements usually result in expressing policies, defining for each user his permissions, prohibitions and obligations. Users (or objects type) are active entities operating on enterprise objects (passive entities) of the system.
Summing up, an enterprise specification is composed of specifications of the elements previously mentioned, i.e. 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 [15].
A contract specifies obligations, permissions and prohibitions for objects comprising in a communities.
Just as for the objects, the actions are also gathered in processes, this implies that there are two levels of abstraction in ODP enterprise viewpoint:

– Abstract level: roles, processes and enterprise viewpoint of the system on which various permissions, prohibitions and obligations are expressed.

– Concrete level: object type (client, server, policy maker, policy administrator), actions (create, delete) and enterprise objects of the system.

Object type, actions and enterprise objects are respectively assigned to roles, processes and enterprise viewpoint by relations defined over these entities(see figure 2). We detail relations in the next sub-section.

Play, Use and belong.

Assignment of Objects type to roles: Objects type are assigned to one or more roles in order to define their privileges. Objects type play their roles in communities, which implies that these objects are assigned to roles through a ternary relation including the community:

play(com, Ot, r): means that the Object type Ot plays the role r in the community com.

Assignment of actions to processes: As for roles and Objects type, processes are an abstraction of various actions authorized in the system. The relation binding actions to processes is also a ternary relation including the community:

belong(com, a, p): means that the action a is considered as a process p in the community com.

Assignment of enterprise objects to enterprise viewpoint: The relation binding the enterprise objects to the enterprise viewpoint to which they belong is also a ternary relation including the community:

use(com, o, v): means that the community com uses the object o in the enterprise viewpoint v.

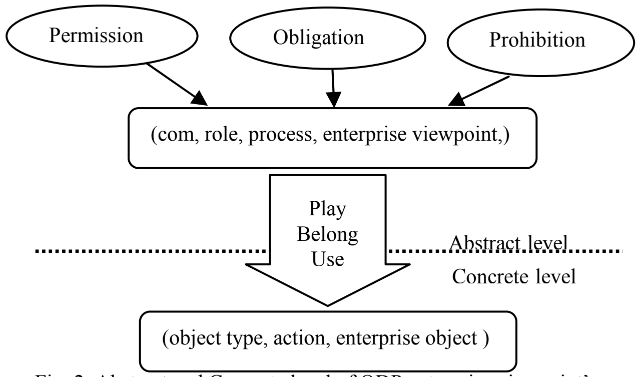
Permission
Obligation
Prohibition
(com, role, process, enterprise viewpoint,)
Play
Belong
Use
Abstract level
Concrete level
(object type, action, enterprise object)

Fig. 2 Abstract and Concrete level of ODP enterprise viewpoint's concepts.

Modeling a Policy. When objects type, actions, and enterprise objects are respectively assigned to roles, processes and enterprise viewpoint, it is now possible to describe the policy. It consists of defining different permissions, prohibitions and obligations:

– permission(com, r, p, v): means that the community com grants to the role r the permission to carry out the process p on the enterprise viewpoint v.

– prohibition(com, r, p, v): means that the community com prohibits the role r to carry out the process p on the enterprise viewpoint v.

– obligation(com, r, p, v): means that the community com require the role r to carry out the process p on the enterprise viewpoint v.

Hierarchy

In situations when two or more groups of objects, under control of different autorities, engage in cooperation to meet a mutual objective, they form a specifal kind of community called a federation.

The hierarchies allow the inheritance of the privileges (permissions prohibitions or obligations), if for example r2 is a sub-role of r1, for a community com, a process p and an enterprise viewpoint v:

– When permission(com, r1, p, v) holds then permission(com, r2, p, v) holds.

– When prohibition(com, r1, p, v) holds then prohibition(com, r2, p, v) holds.

– When obligation(com, r1, p, v) holds then obligation(com, r2, p, v) holds.

In the same way for the communities, if com2 is a sub-community of com1 then, for a role r a process p and an enterprise viewpoint v:

– When permission(com1, r, p, v) holds then permission(com2, r, p, v) holds.

– When prohibition(com1, r, p, v) holds then prohibition(com2, r, p, v) holds.

– When obligation(com1, r, p, v) holds then obligation(com2, r, p, v) holds..

4.2 Event B Models for ODP policies

The expression of the policy in event B includes several successive stages. A first B model is built and then other successive refinements are made as shown by figure 3. The first refinement validates the link between the abstract level (role, ...) and the concrete level (object type, ...).

The approach is based on refinement and each model or refinement model is enriched either by constraints required by the system specification. Each constraint is attached to an invariant. The invariant becomes stronger through the refinement steps.

4.2.1 Abstract Model with policies

As presented in the paragraph 4.1, the enterprise specification has two levels of abstraction (see figure 2). The first step consists of an event B model modeling the abstract part of the policy, i.e. initially, only concepts of community, role, enterprise viewpoint, process are considered. In the first model, permissions, obligations and prohibitions should be described.

– The clause SETS in the event B model contains basic sets such as community, roles, processes, enterprise viewpoint: COMS, ROLES, PROCESSES, ENT VP.

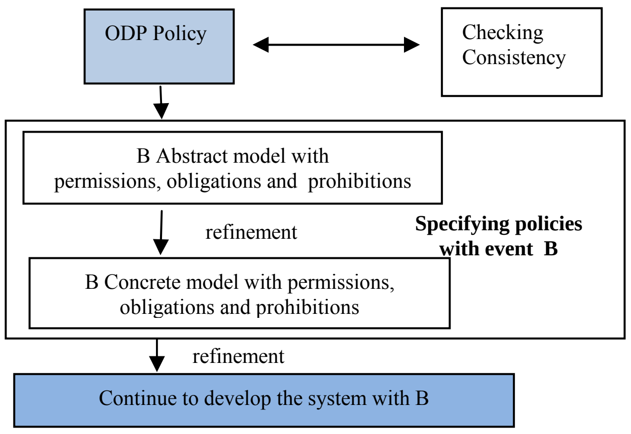
ODP Policy
Checking Consistency
B Abstract model with permissions, obligations and prohibitions
refinement
Specifying policies with event B
B Concrete model with permissions, obligations and prohibitions
refinement
Continue to develop the system with B

Fig. 3. Steps of conception of event- B based model of ODP policies

– The clauses CONSTANTS and PROPERTIES contain the constants like permission, obligation and prohibition that will contain privileges of the ODP system description. Two new constants sub_role and sub_com are introduced to take into account respectively the role and community hierarchy. It is enough to specify which roles and which communities are concerned with inheritances, and the permissions, obligations and prohibitions corresponding to inheritances are deductively generated.

```
SETS
  COMS; ROLES;
  ROCESSES;ENT VP;
```

```
CONSTANTS
   permission, prohibition,
obligation, sub_com, sub_role
```

```
PROPERTIES
permission □ COMS × ROLES × PROCESSES × ENT VP
prohibition □ COMS × ROLES × PROCESSES × ENT VP
obligation □ COMS × ROLES × PROCESSES × ENT VP
sub_org □ ORGS × ORGS
sub_role □ ROLES × ROLES
/ * Organization hierarchies * /
□(com1, com2, r, p, v).
 ((com1 □ COMS □ com2 □ COMS□ r □ ROLES □
 p □ PROCESSES□ v □ ENT VP □ (com1 ↦com2) □
sub_com      □ (com2 ↦ r ↦ p ↦ v) □ permission)
             □    (com1 ↦ r ↦ p ↦ v) □ permission)
/ * Role hierarchies * /
□(com, r1, r2, p, v).  ((r1 □ ROLES □ r2 □ ROLES□ com
□ COMS □ p □ PROCESSES  □ v □ ENT VP □ (r1↦ r2)
□ sub_role □ (com↦r2↦p↦v)□permission)
              □ (com ↦ r1↦ p ↦ v) □ permission)
/ * Same properties for prohibitions and obligation * /
```

For a given particular case, it is enough to initialize sets in the clause SETS by entities, communities, roles, enterprise viewpoint, processes. Properties of constants, like permission, prohibition, obligation, sub_role and sub_com, should also be set in the clause PROPERTIES. Consequently, permissions, prohibitions and obligations cannot be modified, since they are defined as constants.

Introducing State Variables. An event B model expresses properties over state and state variables. Variables are used to model the status of the system with respect to permissions, prohibitions and obligations:
– The clause VARIABLES contains the state variable hist_abst that contains the history of system processes and satisfy the following properties added to the invariant:
INVARIANT

hist_abst □ COMS × ROLES × PROCESSES × ENT VP

hist_abst □ permission

The initial values of the variable is set as follows: hist_abst := ∅
As the policy is supposed to be consistent, we should be able to prove in the clause ASSERTIONS :
ASSERTIONS

permission ∩ prohibition= ∅ permission ∩ obligation = ∅

hist_abst ∩ prohibition = ∅ hist_abst ∩ obligation = ∅

– The clause EVENTS contains the following event :

- The event action models when an authorization request for the access of an object type to an enterprise object of the system occurs.

```
action ≡
  any com, r, v, p where
  com □ COMS □ r □ ROLES □ v □ ENT VP
  □ p □ PROCESSES
  (org ↦ r ↦ p ↦ v) □ permission
  then
    hist_abst := hist_abst □ {(com ↦ r ↦ p ↦ v)}
end
```

The invariant should be preserved and it means that any process in the system is controlled by the policy through the variable hist_abst.

4.2.2 First Refinement: Concrete Model with Policies

We defined two levels of abstraction and the current model is refined into a concrete model. The refinement introduces object types, actions and enterprise objects: sets OBJTYPES, ACTIONS and ENTOBJ contain respectively object types, actions and enterprise objects of the system under development. The clause CONSTANTS includes the following constants: play (assignment of objects types to roles), use (assignment of objects to enterprise viewpoint) and belong (assignment of actions to processes). Properties of constants are stated as follows:
PROPERTIES

play ⊆ COMS × ROLES × OBJTYPES

use ⊆ COMS × ENT VP × ENTOBJ

belong ⊆ COMS × PROCESSES × ACTIONS

Concrete Variables. A new variable hist_conc models the control of the system according to the policy; it contains the history of the actions performed by an object type on a given enterprise object. The context in which the action occurred is also stored in this variable.

The relation between hist_conc and the variable hist_abst of the abstract model is expressed in the gluing invariant; the first part of the invariant states properties satisfied by variables with respect to permissions.

```
INVARIANT
□(ot, a, o).(
   (ot □ OBJTYPES □ a □ ACTIONS□
    o □ ENTOBJ □ (ot ↦ a ↦ o ) □ hist_conc)
           □
(□(com, r, p, v).(com □ COMS □ r □ ROLES□
 p □ PROCESSES □ v □ ENTVP□
(r ↦ ot) □ play□ (v ↦ o) □ use□ (p ↦ a) □ belong
□ (com ↦ r ↦ p ↦ v) □ hist_abst)))
```

The invariant states that each action performed by the system satisfies the policy. For the prohibitions, when a subject s wants to carry out an action a on an object o in an organization org, it is necessary to check that no prohibition exists for that action. The second part of the invariant states properties satisfied by variables with respect to prohibitions and obligations:

```
INVARIANT
∀(ot, a, o).(
  (ot ∈ OBJTYPES ∧ a ∈ ACTIONS∧ o ∈ ENTOBJ
   ∧ (ot ↦ a ↦ o) ∈ hist_conc)
   ⇒
  (∀(com, r, p, v).(com ∈ COMS ∧ r ∈ ROLES∧
   p ∈ PROCESSES ∧ v ∈ ENTVP∧
   (r ↦ ot) ∈ play∧ (v ↦ o) ∈ use ∧ (p ↦ a) ∈ belong)
    ⇒ (com ↦ r ↦ p ↦ v) ∉ prohibition) ∧
   (com ↦ r ↦ p ↦ v) ∉ obligation))
```

```
action ≡
any ot, a, o, com, r, v, p where ot ∈ OBJTYPES ∧ a ∈ ACTIONS
∧ o ∈ ENTOBJ∧ com ∈ COMS ∧ r ∈ ROLES∧ p ∈ PROCESSES
∧ v ∈ ENTVP∧ (r ↦ ot) ∈ play ∧ (v ↦ o) ∈ use∧ (p ↦ a) ∈
belong∧
/ ∗ permission ∗ /
(com ↦ r ↦ p ↦ v) ∈ permission∧
/ ∗ prohibition and obligation∗ /
(∀(comi, ri, pi, vi).((comi ∈ ORGS ∧ ri ∈ ROLES∧ pi ∈
PROCESSES
∧ vi ∈ ENTVP ∧ (ri ↦ ot) ∈ play∧ (vi ↦ o) ∈ use∧ (pi ↦ a) ∈
belong)
⇒((comi ↦ ri ↦ pi ↦ vi) ∉ prohibition)
∧(comi ↦ ri ↦ pi ↦ vi) ∉ obligation))
Then hist_conc := hist_conc ∪ {(ot ↦ a ↦ o)}   end
```

The Events. The abstract model should consider the permissions, the prohibitions and the obligations for an object type ot that asks to perform an action a on an enterprise object o.

5. Conclusions

The use of formal methods in the design process of ODP systems is explicitly required. An important point to take into account is the incorporation of the many proofs which have to be performed in order to be sure that the final system will be indeed «correct by construction».

In this article We presented our approach for developing distributed system in Event B. we used event B for modeling policies in ODP enterprise viewpoint.

The work was carried out on the Rodin platform. In order to verify our models, the abstract and refinement model of ODP policies are developed by using Event-B, Each model is analyzed and proved to be correct.

Our experience strengthens our believe that abstraction and refinement are valuable technique for modeling complex distributed system.

As for future work, we are going to generalize our approach to other concepts in ODP systems. This will be our basis for further investigation of using event-B in the design process of ODP systems. Moreover, case studies should be developed using these models.

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 and al., ‘’A UML-Based Meta-language for the QoS-aware Enterprise Specification of Open Distributed Systems’’ IFIP Series, Vol 85, Springer, (2002) 255-264.

[6] Mohamed Bouhdadi and al. ‘A Semantics of Behavioural Concepts for Open Virtual Enterprises’. Series: Lecture Notes in Electrical Engineering, , Vol. 27 .Springer, 2009. p.275-286.

[7] Belhaj H and al. Event B for ODP Enterprise Behavioral Concepts Specification, Proceedings of the World Congress on Engineering 2009 Vol I, WCE '09, July 1 - 3, 2009, London, U.K., Lecture Notes in Engineering and Computer Science, pp. 784-788, Newswood Limited, 2009

[8] Mohamed Bouhdadi and al., ‘Using BPEL for Behavioural Concepts in ODP Enterprise Language’, Virtual Enterprises and Collaborative Networks, IFIP, Vol. 283, pp. 221-232, Springer, 2008

[9] Mohamed Bouhdadi and al., ‘Meta-modelling Syntax and Semantics of Structural Concepts for Open Networked Enterprises’, Lecture Notes in Computer Science, Vol. 4707, pp. 45-54, Springer, 2007.

[10] Myers, G. The art of Software Testing, John Wiley &Sons, New York, 1979

[11] Binder, R. Testing Object Oriented Systems. Models. Patterns, and Tools, Addison-Wesley, 1999

[12] Belhaj Hafid and al.: Using Event B to specify QoS in ODP Enterprise language. PRO-VE'10 11th IFIP Working Conference on VIRTUAL ENTERPRISES, Saint-Etienne, France, 11-13 October 2010.
[13] J.-R. Abrial. The B-Book: Assigning programs to meanings. Cambridge University Press, 1996.
[14] Belhaj Hafid, Bouhdadi Mohamed, El hajji Said : Verifying ODP trader function by using Event B. IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 9, July 2010.
[15]ISO/IEC. RM-ODP Enterprise Language. Draft International Standard ISO/IEC 15414, ITU-T X.911, ISO, 2001.
[16] http://www.event-b.org/
[17] RODIN. Development Environment for Complex Systems (Rodin). 2009. http://rodin.cs.ncl.ac.uk/.
[18] P. Linington. RM-ODP: The architecture. In K. Milosevic and L. Armstrong, editors, Open Distributed Processing II, pages 15–33. Chapman & Hall, Feb. 1995.
[19] Mohamed Bouhdadi and al. " A UML/OCL Meta-model Syntax for Structural Constraints in ODP Enterprise Language" Journal WSEAS Transactions on Computers, Vol 6, Issue 1, WSEAS Press, pp:31-36, 2007.
[20] Francisco Duran, Javier Herrador, and Antonio Vallecillo. "Using UML and Maude for Writing and Reasoning about ODP Policies". Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks (POLICY'03) , Lake Como (Italy). pp. 15-25, IEEE Computer Society Press, June 2003.
[21] Joochim, T., Snook, C., Poppleton, M. and Gravell, A. (2010) TIMING DIAGRAMS REQUIREMENTS MODELING USING EVENT-B FORMAL METHODS. In: IASTED International Conference on Software Engineering (SE2010), February 16 – 18, 2010, Innsbruck, Austria.
[22] C.Snook & M.Butler, UML-B and Event-B: an integration of languages and tools. Proc. IASTED International Conf. on Software Engineering (SE2008), Innsbruck, Austria, 2008.
[23] P.F. Linington, Z Milosevic, and K. Raymond, Policies in communities: Extending the enterprise viewpoint. In Proc. 2nd International Workshop on Enterprise Distributed Object Computing (EDOC'98), San Diego, USA, page 11, November 1998.