A Meta-model Syntax for Structural Constraints in ODP Enterprise Language

Mohamed. Bouhdadi, El maâti. Chabbar, Hafid belhaj, Youssef bellouki

Department of Mathematics and Computer Science
University Mohammed V Faculty of Sciences
B.P1014 Rabat Morocco

.

Abstract: - The Reference Model for Open Distributed Processing (RM-ODP) provides a framework within which support of distribution, inter-working and portability can be integrated. It defines un object model; architectural concepts and an architecture for the development of ODP systems in terms of five viewpoints. However, RM-ODP is a meta-norm and several ODP standards have to be defined. Indeed the viewpoint languages are abstract in the sense that they define what concepts should be supported not how these concepts should be represented. Using the meta-modeling approach we define in this paper the syntax for a fragment of ODP organizational defined in the foundations part and in the enterprise viewpoint language. These concepts are suitable for describing and constraining ODP enterprise viewpoint specifications. This meta-modeling approach could be used to define semantics and concepts characterizing dynamic behaviour in ODP enterprise viewpoint.

Key-Words: - RM-ODP , Organizational Concepts, Enterprise Viewpoint language, Meta-modeling Syntax.

1 Preliminaries.

The rapid growth of distributed processing has led to a need for coordinating framework for the standardization of Open Distributed Processing (ODP). The Reference Model for Open Distributed Processing (RM-ODP) [1-4] provides such a framework. It creates an architecture within which support of distribution, networking and portability can be integrated.

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 including basic modeling concepts, specifications concepts, organizational concepts, and structuring 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, called enterprise, information, computational, engineering and technology which provide a basis for the specification of ODP systems.
- A viewpoint language for each viewpoint, defining concepts and rules for specifying ODP systems from the corresponding viewpoint
- Specifications of the functions required to support ODP systems
- Transparency prescriptions showing how to use the ODP functions to achieve distribution transparency;

.In other words, the three viewpoints do not take into account the distribution and heterogeneity inherent problems. This principle corresponds closely to the concepts of PIP/PSM models in the OMG MDA architecture [5].

However, RM-ODP is a meta-norm [8] and can not be directly applicable. Indeed, for example, the viewpoint languages are abstract in the sense that they define what concepts should be supported, not how these concepts should be represented. It is important to not that, RM-ODP uses the term language in its broadest sense:" a set of terms and rules for the construction of statements from the terms, «and does not propose any notation for supporting the viewpoint languages. In fact, RM-ODP only provides a framework for the definition of new ODP standards. These standards include standards for ODP functions [6-7], standards for modeling and specifying ODP systems; standards for methodology, programming, implementing, and testing ODP systems. Elsewhere the languages Z [9], SDL [10] and LOTOS [11], and Esterelle [12] are used in RM-ODP architectural semantics part [4] for the specification of ODP concepts. Elsewhere, up to now no formal method is likely to be suitable for specifying and verifying every aspect of an ODP system. The inherent characteristics of ODP systems imply the need to integrate different specification languages, and to handle non-behavioral properties of ODP systems.
There had been an amount of research for applying the UML [13] as a syntactic notation to the ODP viewpoints [14-17]. The approach taken is to give a meta-model description for the language ; it is a definition of that language in terms of itself.
This is presented 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 OCL [18]. Indeed, we used the meta-modeling approach in our work [19] in order to define the syntax of a sub-language for the ODP QoS-aware enterprise viewpoint specifications. We used OCL for specifying the context constraints of the syntax of the diagrammatical languages defined based on UML.

Elsewhere, a part of UML meta-model itself has a well defined semantics. Hence UML could be adequate for ODP systems which necessitate well formed and unambiguous languages in order to build ODP automatic tools which have to make use of semantic content. In order to give a semantics to a modelling language (which may not be directly executable), there are, essentially two approaches : an axiomatic approach, which states what sentences in the languages ca be derived from other sentences ; and a denotational approach, where expressions are mapped to the « instances » they denote.

A denotational approach [20] would be realised by a) a definition of the form of an instance of every UML language element (e.g. the objects that could be denoted by a class, the links that could be denoted by associations, etc.) and b) a set of rules which determine which instances are and are not denoted by a particular language element.
There are three main steps to a denotational, meta-modelling approach to the semantics approach:

1. Define the meta-model for the language of the model: object template, interface template, action template, type, role
2. Define the met-model for the language of the instances: objects, links, and interfaces,
3. Define the mapping or the meaning function (also within the meta-model) between these two languages.

There are good reasons for adopting a meta-modelling approach for in the context of UML and of ODP systems. The UML meta-models provide a blueprint for the core of any CASE tool. The tools include a consistency checker that makes sure invariants defined on a model do not conflict, a consistency checker between meta-models that makes sure that different system specifications are consistent and do not conflict. Also, tools can be built which generate code from UML meta-models, and these tools can be used to bootstrap themselves every time the meta-model is changed or extended. Furthermore, for testing ODP systems [2-3], the current testing techniques [21], [22] are not widely accepted. However, a new approach for testing, namely agile programming [23], [24] or test first approach [25] is being increasingly adopted. The principle is the integration of the system model and the testing model using UML meta-modeling approach [26].

This approach is based on the executable UML [27]. In this context OCL is used to specify the properties to be tested. OCL also serves to attach constraints to UML meta-models in order to verify the coherence of meta-models and to translate the constraints into code for evaluating them on instance models.

The part of RM-ODP considered in this paper is a subset for describing ODP enterprise object structure. It consists of modeling and specifying concepts defined in the RM-ODP foundations part and concepts in the enterprise viewpoint language. We do not consider concepts for describing dynamic behaviour.

For characterizing models, it includes the essentials of class diagrams, and a significant fragment part of the OCL, a precise language based on first order –logic, used for expressing constraints on object structure which cannot be expressed by class diagrams alone. For characterizing instances of models, it includes the language of object diagrams. Because of the role of OCL in ODP information viewpoint specifications, a major component of the meta-model presented is a representation of the concepts underpinning OCL. Thus the UML/OCL meta-model developed here elaborates the conceptual core of the ODP enterprise viewpoint language for ODP enterprise specifications. It is not tied to any particular concrete syntax.

Section 2 describes the subset of concepts considered in this paper namely the object model, organizational concepts and enterprise language. Section 3 describes the the enterprise language defined in the framework part. Section 4 describes the meta-model for generic models, object, action, interaction, interface, template, type/subtype, class/subclass, basic class/derived class. A conclusion and perspectives end the paper. .

2 The RM-ODP

RM-ODP is a framework for the construction of open distributed systems. It defines a generic object model in the foundations part and an architecture which contains the specifications of the required characteristics that qualify distributed processing as open. The architecture extends and specializes the object concepts of the foundations part.

2.1 The RM-ODP Object Model (Foundations part)

In general, the term object model refers to the collection of concepts used to describe objects in an object-oriented spacification (OMG CORBA object model [2], RM-ODP object model [4]. It corresponds closely too the use of the erm data-model in the relational data model. To avoid misunderstandings, theh RM-ODP defines each of the concepts commonly encountered in objectt oriented models. It underlines a basic object model which is unified in the sense that it has successfully to serve each of the five ODP viewpoints. It defines the basic concepts concerned with existence and activity: the expression of what exists, where it is and what it does. The core concepts defined in the object model are object and action.

An object is the unit of encapsulation : a model of an entity. It is characterized by its behavior and, dually, by its states. Encapsulation means that changes in an object state can occur only as a result of interna actions or interactions.

An action is a concept for modeling something which happens. ODP actions may have a duration and may overlap in time. All actions are associated with at least one object : internal actions are associated with a single object ; interactions are actions asociated with several objects.
Objects have an identity, which means that each object is distinct from any other object. Object identity implies that there exists a reliable way to refer to objects in a model.

Depending on the RM-ODP viewpoint, the emphasis may be placed on the behavior or on the states. When the emphasis is placed on behavior an object is informally said to perform functions and offer services, theses functions are specified in terms of interfaces. It interacts with its environment at its interaction points which are its interfaces. An interface is a subset of the interactions in which an object can participate. In contrast to other object models, an ODP object can have multiple interfaces. Like objects, interfaces can be instantiated.

The other concepts defined in the object model are derived from the concepts of object and action; those are class, template, type, subtype/supertype, subclass/superclass, composition, and behavioral compatibility.

Composition of objects is a combination of two or more objects yielding a new object.

An object is behaviorally compatible with a second object with respect to a set of criteria if the first object can replace the second object without the environment being able to notice the difference in the object behavior on the basis of the set of criteria.

A type (of an $<x>) is a predicate characterizing a collection of <x>s. Objects and interfaces can be typed with any predicate, but are commonly typed on the basis of the template of which they are intances. The ODP notion of type is much more general than of most object models. Also ODP allows ODP to have several types, and to dynamically change types.

A class (of an <x>) defines the set of all <x>s satisfying a type. An object class, in the ODP meaning, represents the collection of objects that satisfy a given type. Manyy objectt models do not clearly distinguish between a specification for an object and the set of objects that fit the specification. ODP makes the distinction template and class explicit. The class concept corresponds to the OMG extension concept, the extension of a type is the set of values that satisfy the type at a particular time. A subclass is a subset of a class. A subtype is therefore a predicate that defines a subclass. ODP subtype and subclass hierarchies are thus completely isomorphic.

A <x> template is the specification of the common features of a collection x in a sufficient detail that an x can be instantiated using it.

Types, classes, templates are needed for object, interface, and action.

2. 2 RM-ODP organizational concepts

The definition of a language for each viewpoint describes the concepts and rules for specifying ODP systems from the corresponding viewpoint. The object concepts defined in each viewpoint language are specializations of those defined in the foundations part of RM-ODP. We give here the organizational concepts.

<x> Group : a set pf objects with a particular characterizing relationship <x>. The relationship <x> characterizes either the structural relationship among objects or a expected common behaviour of the objects. Examples of specialize groups are : addressed group, fault group, communication group.

Configuration (of objects): a collection of objects able to interact at interfaces. A configuration determines the set of objects involved in each interaction. The specification of a configuration may be static or may be in terms of the operation of dynamic mechanisms which change the configuration, such as binding and unbinding.

<x> Domain : a set of objects, each of which is related by a characterizing relationship <x> to a controlling object. Each domain has a controlling object associated with it. The controlling object can determine the identities of the collection of the objects which comprises the associated domain. The controlling object may communicate with a controlled object dynamically or it may be considered to have communicated in an earlier epoch of the controlling object. Generally, the controlling object is not a member of the associated domain.
In enterprise terms, various policies can be administered by the controlling object over the domain. Domains can be disjoint or overlapping. By definition, a domain is a group, but not vice versa.

Subdomain : a domain which is a subset of a domain;

3 The RM-ODP enterprise viewpoint language

An enterprise specification is concerned with the purpose, scope and policies for theh ODP system. Bellow, we summarise the basic enterprise concepts.

Community is the key enterprise concept. It is defined as a configuration of ojects formed to meet an objective. The objective is expressed as a contrat that specifies how the objective can be meet.

A contract is a generic concept that specifies an agreement governing part of the collective behaviour of a set of objects. A contarct specifies obligations, permissions and prohibitions for objects involved. A contact specification may also include the specification of different roles engaged in the contract, the interfaces associated with the roles, quilatyy of service attributes, indications of period of validity, behaviour that invalidate the contrat, andn liveness and safety conditions.

The community specification also includes the environment contracts that state policies governing interactions of this community with itst environment.
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.

A role is a specification concept describing behaviour. A role may beb composed of several roles. A configuration of objects establishe for achieving some objective is referred to as a communiti. A role thus identifies behaviours to be fulfilled by the objects comprising the community. A 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.

A policy statement provides a additional behavioural specification. A community contract uses policy statements to separate behavioural specifications about roles. Examples of policy are the statements of permissions, prohibitions, and the obligations related to the roles or the enterprise objects.

A policy is a set of rules related to a particular purpose. A rule can be expressed as an obligation, a permission, or a prohibition. An ODP system consists of a set of enterprise objects.

An enterprise object may be a role, an activity or a policy of the system.

4 Syntactic Domain

In this section we give the meta-models describing the contex free syntax for the above concepts. Figures 1, 2 and 3 define the objet model, the organizational concepts and the enterprise viewpoint concepts.

We give here some context contraints in OCL.

Context m : Model inv :

m.Roles->includesAll(m.Roles.Source ->union(m.Roles.Target)
m.Roles->includesAll(m.ObjectTemplates.Roles)
m.Roles->includesAll(m.Interactiontemplate.roles)
m.Roles->includesAll(m.InterfaceTemplate.roles)

m.InteractionTemplates -> includesAll(m.ObjectTemplates.Interactiontemplates)
m.InteractionTemplates.Types->includesAll(m.Types)

m.ObjectTemplates.InterfaceTemplates->includesAll(m.InterfaceTemplates)

m.ObjectTemplates.InterfaceTemplates->includesAll(m.InterfaceTemplates)
includesAll(m.ObjectTemplates.Interactiontemplates)
m.Types->includesAll(m.InteractionTemplates.Types-
>union(m.InterfaceTemplates.Types)- >union(m.InteractionTemplate.Target)

Context i : Interaction template inv :
r.role.inverse = r.Interactions.Roles.Source .inverse
and r.role.source = r.Interactions.Roles.Source .source
and r.role.source.inverse = r.Interactions.Roles.Source .inverse

Context o : Object Template inv :
cot is not parent of or child of itself
not (cot.parents ->includes(cot) or cot.children->includes(cot))

4 Conclusion and perspectives

The Reference Model for Open Distributed Processing (RM-ODP) is a meta-norm which provides a framework within which support of distribution, inter-working and portability can be integrated. It defines un object model; architectural concepts and an architecture for the development of ODP systems in terms of five viewpoints. However,. the viewpoint languages are abstract in the sense that they define what concepts should be supported not how these concepts should be represented. Using the meta-modeling approach we define in this paper the syntax for a fragment of ODP organizational defined in the foundations part and in the enterprise viewpoint language. These concepts are suitable for describing and constraining ODP enterprise viewpoint specifications. This meta-modeling approach could be used to define semantics and concepts characterizing behavioral concepts in ODP enterprise viewpoint.

References

[1] ISO/IEC, Basic Reference Model of Open Distributed Processing-Part1: Overview and Guide to Use, ISO/IEC CD 10746-1, July 1994.
[2] ISO/IEC, RM-ODP-Part2: Descriptive Model, ISO/IEC DIS 10746-2, February 1994.
[3] ISO/IEC, RM-ODP-Part3: Prescriptive Model, ISO/IEC DIS 10746-3, February 1994.
[4] ISO/IEC, RM-ODP-Part4: Architectural Semantics, ISO/IEC DIS 10746-4, July 1994.
[5] OMG, The Object Management Architecture, OMG, 1991. http://www.omg.org
[6] ISO/IEC, ODP Type Repository Function , ISO/IEC JTC1/SC7 N2057, January 1999.
[7] ISO/IEC, The ODP Trading Function, ISO/IEC JTC1/SC21, June 1995.
[8] M. Bouhdadi et al. A Methodology for the Development of Open Distributed Systems, Proc. JDIR'98, Paris France October 1998, pp. 200-208
[9] J.M. Spivey, The Z Reference manual, Prentice Hall, 1992.
[10] IUT, SDL: Specification and Description Language, IUT-T-Rec. Z.100, 1992.
[11] ISO and IUT-T, LOTOS: A Formal Description Technique Based on the Temporal Ordering of Observational Behavior, ISO/IEC 8807, August 1998.
[12] H. Bowman et al. FDTs for ODP, Computer Standards & Interfaces Journal, Elsevier Science Publishers, Vol.17, No.5-6, 1995, pp. 457-479.
[13] J. Rumbaugh et al., The Unified Modeling Language, Addison Wesley, 1999.
[14] www.edoc.org
[15] B. Rumpe, A Note on Semantics with an Emphasis on UML, Second ECOOP Workshop on Precise Behavioral Semantics, Technische Universitaty unchen publisher, 1998.
[16] A. Evans et al., Making UML precise, OOPSLA'98, October 1998,
[17] A. Evans et al. The UML as a formal notation, UML'98, France June 1998, LNCS 1618, Springer Berlin, 1999, pp. 336-348
[18] J. Warner and A. Kleppe, The Object Constraint Language: Precise Modeling with UML, Addison Wesley, 1998.

[19'] M. Bouhdadi, An UML-based Meta-language for the QoS-aware Enterprise Specification of Open Distributed Systems, IFIP TC5/WG5.5 Third Working Conference on Infrastructures for Virtual Enterprises (PRO-VE'02), May 1-3 Sesimbra Portugal, Kluwer Vol. 213 (IFIP Conference Proceeding series), 2002. Collaborative Business Ecosystems & Virtual Enterprises IFIP Series Vol. 85 Springer Boston 2002.
[20] D.A. Schmidt, Denotational semantics: A Methodology for Language Development, Allyn and Bacon, Massachusetts, 1986.
[21] Myers, G. The art of Software Testing, John Wiley &Sons, New York, 1979
[22] Binder, R. Testing Object Oriented Systems. Models. Patterns, and Tools, Addison-Wesley, 1999
[23] Cockburn, A. Agile Software Development. Addison-Wesley, 2002.
[24] Bernhard Rumpe. Agile Modeling with UML. Habilitation Thesis, Germany, 2003.
[25] Beck K Column on Test-First Approach. IEEE Software, 18(5):87-89, 2001
[26] Briand L. and Labiche Y. A UML-based Approach to System testing. In M. Gogolla and C. Kobryn (eds): " UML" – The Unified Modeling Language, 4th Intl. Conference, LNCS 2185. Springer, 2001 pp. 194-208,
[27] Bernhard Rumpe, Executable Modeling with UML. A vision or a Nightmare? In Issues & Trends of Information Technology Management in Contemporary Associations, Seattle. Idea group Publishing, Hershey, London, pp. 697-7001. 2002