Papier publié dans «Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02), IEEE (Ed), Oxford, England, August 2002, pp265-270»

Towards an MDA-Oriented Methodology

Marie-Pierre Gervais
Laboratoire d'Informatique de Paris 6 (LIP6) and Université Paris X
8 rue du Capitaine Scott - F75015 Paris
Marie-Pierre.Gervais@lip6.fr

Abstract

With the introduction of the Model Driven Architecture by the OMG, modeling technology seems to take the step on the middleware technology for distributed applications development. The MDA recommends to separate the business from their technical aspects in the development of applications. Modeling techniques offer tools providing abstraction that enables the isolation of business concerns from their technical achievement. EDOC Profile is an example of such tools as it provides a modeling framework. It makes use of the RM-ODP architectural framework that provides conceptual tools. These tools, so useful are, remain insufficient in the sense that they do not provide any process to guide the software designers in the modeling step, i.e., they are not methodological tools.

We present in this paper a methodology based on the RM-ODP that falls under the MDA initiative. We describe its principles by illustrating them with an example. Then we provide the research directions enabling to get a fully compliant MDA methodology.

1. Introduction

The combined evolution of the telecommunications technology and the organizations structure leads to the emergence of complex distributed applications. To assist the software engineers when developing such applications, middleware technology emerged, offering facilities supporting distribution management. CORBA, .NET, EJB or CCM are examples of such technologies. However, the proliferation of these platforms raises new problems of interoperability, increasing the cost of migration and adaptation from one platform to a new one. A way of managing this heterogeneity is to base the distributed applications development on the separation of concerns between the business aspects and the technical aspects. In this optic, OMG issued the Model-Driven Architecture (MDA) initiative aiming at a global approach for integration and interoperability where models play a central role [9]. Indeed, the use of modeling techniques is strongly anticipated as a way of simplifying the construction of applications. In this way, the main part of the development becomes an activity upstream, dedicated to the business concerns through the elaboration of the application specification that abstracts away technical details, i.e., the so-called Platform-Independent Model (PIM). The transformation of a PIM into a Platform-Specific Model (PSM) is then achieved when introducing into the PIM the technical considerations depending on the chosen platform.

In conformance with this MDA approach, the OMG Enterprise Distributed Object Computing (EDOC) Profile has been produced aiming at simplifying the development of component based systems [10]. This is a modeling framework composed of set of UML Profile Elements [11]. Among them is the Enterprise Collaboration Architecture (ECA) that is a technology independent profile allowing the definition of PIMs. ECA is an MDA approach for specifying systems that uses as conceptual framework the Reference Model of Open Distributed Processing (RM-ODP). The RM-ODP developed by the International Standardization Organization (ISO) and the International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) provides an architectural framework that defines a set of rigorous concepts for developing distributed systems [6].

Thus the EDOC modeling framework and the RM-ODP architectural framework provide a set of very useful tools, respectively modeling and conceptual tools. These support the elaboration of sophisticated models describing at various levels of abstraction the applications to be developed. However, none of them includes precise rules or guidelines explaining how software engineers can use these tools to build their models. Together they provide a set of concepts and a notation, but no steps or process. In fact, they do not provide a methodology that is a set of concepts, the usage rules of these concepts by organizing them into various steps, the process associated with these steps and a notation.

We present in this paper a methodology named ODAC, which is based on the RM-ODP and has the potential to be an MDA-oriented methodology. Indeed, ODAC makes use of the RM-ODP concepts, defines steps, process for these steps and uses the UML notation and profiles. It then provides a set of guidelines in terms of UML Profiles to software engineers to explain how they can describe the architecture of the system he/she is building, from the most abstract level to the most concrete level, which is the implementation of the system. The methodology is not dedicated to a domain of applications. For example, we use it in the active network area in one hand and in the mobile agents systems field on the other hand [3][4].

This paper is organized as follows. Section 2 provides an overview of the ODP standards, pre-required to approach our methodology, which is introduced in Section 3. Section 4 focuses on the ODAC part devoted to the PIM elaboration. Section 5 provides the research directions enabling to get a fully compliant MDA methodology. The conclusion expresses the future developments of the methodology and draws some issues to this work.

2. The ODP Standards

The Open Distributed Processing (ODP) standards developed by ISO and ITU-T provides a Reference Model (RM-ODP) that defines an architectural framework for the construction of distributed systems and applications [8]. It provides a set of concepts and their structuring rules. A main concept is that of *viewpoint*, which enables to structure the modeling activities. A viewpoint is a subdivision of a complex system specification. It corresponds to a particular perspective, allowing the system to be “viewed” from a particular angle, focusing on specific concerns. The five viewpoints are the Enterprise, Information, Computational, Engineering and Technology viewpoints.

The whole specification of a system is obtained by establishing the five specifications from each viewpoint and their correspondences. Nevertheless, RM-ODP is an architectural framework and not a methodology, i.e. it is not prescriptive enough and does not provide tools such as a notation or a sequence between the viewpoints that would help the software designers to build the system specification. We then present in the next Section how we use this framework and supplement it in order to provide a methodology.

3. The ODAC Methodology

As mentioned previoulsy, a methodology should provide a set of *concepts*, the usage rules of these concepts by organizing them into various *steps*, the *process* associated with these steps and a *notation*. We describe below how these various elements are defined in ODAC.

3.1. The ODAC Concepts and Steps

The ODAC *concepts* are the ODP concepts, in particular the viewpoint concept. It is used to define the *ODAC steps* by identifying a parallel between the activities of analysis, design and implementation and the ODP viewpoints (Figure 1). Thus the ODAC methodology associates in an informal way the Enterprise, Information and Computational viewpoints to the analysis, the Engineering viewpoint to the design and the Technology viewpoint to the implementation. It distinguishes then in these steps intended to define the system those that describe it independently of any target environment of those that describe it according to the environment in which it will be carried out. Three categories of specifications are identified: *Behavioral Specification* of the system, *Engineering Specification* of the system and its *Operational Specification*. The relationships between them are illustrated in Figure 2.

The *Behavioral Specification* is the output of the analysis, i.e., it corresponds to the specifications established in the Enterprise, Information and Computational viewpoints. It describes the system according to its objective, its place in the company in which it is developed, information that it handles and the tasks that it carries out. Thus it is a Platform-Independent Model (PIM) as described in the MDA [9].

The *Engineering Specification* is the specification established in the Engineering viewpoint, i.e., the description of the execution environment. It can be seen as a Platform-Description Model (PDM) [1]. Thus for each kind of environment, ODAC provides the designer with a PDM Guideline that helps for describing the considered environment.

The *Operational Specification* results from the projection of the behavioral specification (i.e., a PIM) on a target environment reflecting the real execution environment. It constitutes the description from which code is generated and the implementation is carried out. It corresponds to the transformation of the PIM, which is configured according to the PDM. It is the Platform-Specific Model (PSM) of the MDA.

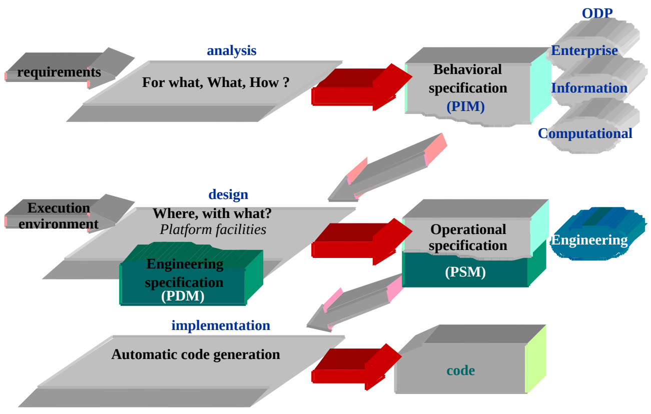

Figure 1: The ODAC Methodology Principles

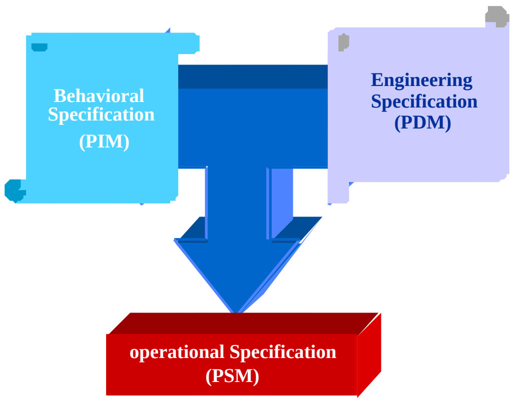
Behavioral Specification (PIM)
Engineering Specification (PDM)
operational Specification (PSM)

Figure 2: The Three Models in ODAC and their Relationships (from [1])

3.2. The ODAC Process and Notation

Associated with each step, a process is provided that prescribes how the modeler can establish the specification.

The ODAC analysis process prescribes to consider first the Enterprise viewpoint, then Information and Computational viewpoints. The system description according to one of these viewpoints gives place to a specification in this viewpoint. Consequently, for each viewpoint, ODAC identifies a set of steps to write the corresponding specification, the concepts involved at each step and the associated notation. All the obtained specifications constitute the PIM of the system. Regarding to the notation, ODAC makes use of the UML notation and its extension mechanism. Thus to help the modeler for writing such a PIM, we provide the “ODAC Guideline for PIM” as a UML profile (cf. Section 4).

In the same way, the ODAC design process consists of providing a set of rules with the associated notation to enable the designer to describe the considered environment in order to obtain the PDM. Once again, the UML profile approach is used. Then each guideline for each kind of environment mentioned above, namely the MASIF-DESIGN Guideline for describing mobile agent platforms and the ODAC*for*ANTS Guideline for describing active network platforms [4][3].

We focus hereafter on the ODAC Guideline for PIM we propose. To illustrate how the modeler can apply the principles mentioned in this guideline, we use a case study briefly introduced hereafter.

3.3. The Reliable Active Multicast Service

The Reliable Active Multicast (RAM) service enables the emission of data by a source to a set of receivers that request for them by subscription. It is reliable since lost data are retransmitted [12].

A station that wants to receive data first subscribes to a source. The set of subscribers and the source constitute a group. In the group, there is only one source. Links between source and receivers are not necessarily direct: “intermediate nodes” can exist. The source broadcasts data to all the receivers of its group, through the “intermediate nodes”. A receiver has the ability to detect and to notify the loss of data to the source. In this case, the source retransmits data.

4. The ODAC Guideline for PIM

The PIM deals with the functional aspects of the system under development, describing the behavior of the system and its business logic, with no concerns of its technical aspects relating to the execution environment. We provide a PIM modeler with a UML profile that is a composition of three profiles (one for each viewpoint involved in a PIM elaboration, namely the Enterprise, Information and Computational viewpoints) and their correspondences.

As each profile enables the modeler to describe the same system according to a specific concern, the three specifications must be consistent. Consequently the guideline includes correspondence rules that apply on the specifications. The set of the three specifications together with their correspondences constitutes the PIM of the system.

For reasons of brevity, we limit here the presentation to a part of the profile for the Enterprise specification dedicated to the specification steps. These are a set of rules that prescribes how the modeler must apply the profile to obtain the specification of a viewpoint.

4.1. The Process for Developing an Enterprise Specification

An Enterprise specification describes the system behavior in the environment with which it interacts. It is an abstraction of the system and the environment in which this system exists. It describes the relevant aspects of what the system is supposed to do in the context of the objective, the scope and the policies of its environment. ODAC defines the process for elaborating an Enterprise specification as composed of several steps that we identify from the Enterprise viewpoint concepts as described in [7]. To supplement this process with its steps and their associated concepts, the methodology provides a notation in order that the software engineer expresses the result of each step. So we have mapped the ODP concepts onto the UML. We describe hereafter each of these steps and provide an illustration with the RAM service example. The figures are realized with Rational Rose©.

Step 1: defining the objective

The `objective`[1] of the service is to provide a RAM service that guarantees the reception of all the sending messages from a source to a set of receivers that subscribe the services (Figure 3). It is represented by a Use Case.

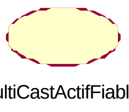

Figure 3: Objective of the RAM Service

Step 2: Role Types

Enumerating all the `role types` of the system and their associated `behavior` enabling to perform this objective. For this, refine the objective several times, stop the refinement when elementary objectives are identified, i.e., the decomposition is no longer possible and assign a role type to each elementary objective. To represent a type of role and its associated behavior, we use a class stereotype «RoleType». Since a role is an identifier of a behavior made up of `actions`, list these actions and make distinction between `interactions` and internal actions (underline names of internal actions). Express the constraints on the occurrence of an action as UML notes, if there are some;

We identify in the RAM service the following role types (Figure 4):

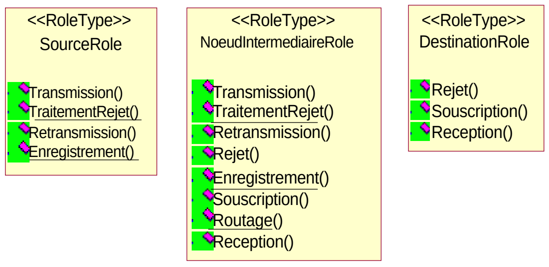

Figure 4: The RAM Service Roles Types

- The SourceRole: produces and sends data;
- The DestinationRole: receives data for which they subscribe;
- The IntermediateNodeRole: transmits data to receivers.

The objects that fulfill a role instantiated from the role type “IntermediateNodeRole” can interact together.

Step 3: Identifying the S-Community

An Enterprise specification includes at least the description of a `community` in which the ODP system is seen as a single `Enterprise Object` interacting with its `environment`. This community is referred to as the `S-Community`. It is represented as a Use Case diagram in which UML actors represent the environment roles and Use Cases represent the objective of the system.

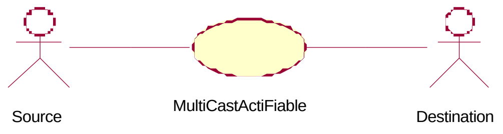

Figure 5: the RAM Service S-Community

Among the roles enumerated above, we identify that the SourceRole and DestinationRole represent the environment of the system whereas the

[1] When mentioned for the first time, the terms corresponding to ODP concepts will be written in this font

IntermediateNodeRole is intrinsic to the RAM system (Figure 5).

Step 4: Identifying the Interface roles

An Enterprise specification can include the description of more than one community. It can thus be structured in a set of communities that interact, each of these communities being seen as a composite object (called the C-object). The particular case of the C-object leads to the definition of the so-called `interface roles`. This role represents some services proposed by a community (as a C-object) within a larger community.

There is only one community in the RAM service that models the multicast group. It includes all the objects that fulfill the roles that are instances of the identified role types. Thus in this example, there is no interface role.

For each community:

Step 5: Identifying the Enterprise Objects

`Enterprise Object` is the basic concept of the Enterprise viewpoint. It is the model of an entity, this being a part of the universe of discourse (for example, human beings, data processing systems, resources etc.). It is represented by a UML object. However, let us note that in general, an Enterprise specification does not deal with instances, but rather with types or templates. The notation of anonymous object offers this facility and can be used.

For the RAM service, we use this notation for objects fulfilling the roles instantiated from the SourceRole and DestinationRole role types and we choose two objects O_i and O_j that fulfil a role instantiated from the IntermediateNodeRole role type (Figure 6). This enables us to introduce interactions between objects fulfilling a role of the same type [7].

Step 6: Describing the Community behavior

A major structuring concept of an Enterprise specification is that of **community**. This is a configuration of Enterprise objects formed to fill an **objective**. The community behavior expresses the temporal progress of the interactions between the Enterprise Objects. The term "interaction" is used here with ODP semantics, i.e. it is not synonymous of method invocation. Therefore, the arrows used in Figure 6 reflect no meaning of this type. The behavior of the RAM service is represented in Figure 6.

Step 7: Describing the Policies

The `policies` are used to express rules, which apply to the Enterprise objects, to the roles and to the communities. A policy can be expressed as an obligation, an authorization, a permission or a prohibition. They are represented as UML notes, which contain OCL (Object Constraint Language) expressions. The expressions are presented in terms of invariants, pre or post conditions and guards.

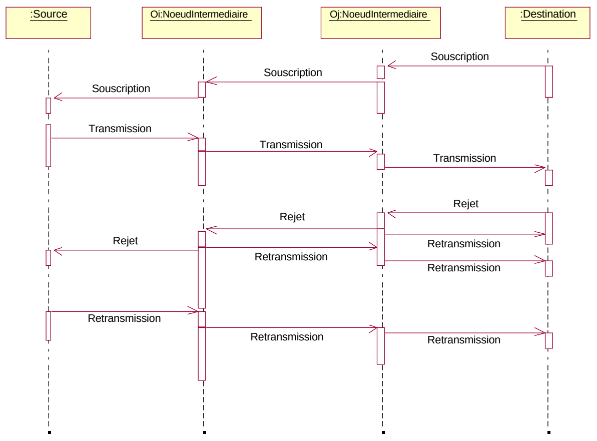

Figure 6: The Behavior of the Community

An example of the policies of the RAM service is the policy applied on the subscription action of the DestinationRole: "A subscription request is sent to one source (and only one) of a multicast group", which corresponds to an OCL invariant (Figure 7).

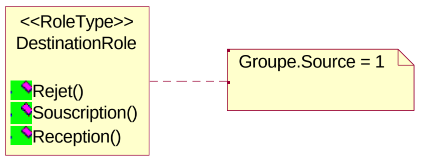

Figure 7: A Policy for the RAM Service

4.2. Conclusion

In the same way, we provide an Information Profile and a Computational Profile, with their correspondence rules [5]. Together, the three profiles and their correspondence rules constitute a guideline enabling the software modelers to define the PIM of the application being developed. This ODAC Guideline for PIM is supplemented with PDM guidelines enabling the software modelers to describe the platform providing the technical facilities for the applications execution. In this sense, ODAC is a methodology compliant with the MDA principles. We give in the next Section the issues to be solved in order to get an MDA-oriented methodology.

5. ODAC : An MDA-Oriented Methodology

As mentioned in Section 4, ODAC defines its own UML profiles in the Guideline for PIM. In order to be aligned with the MDA recommendations, ongoing work in ODAC is the use of the EDOC Profiles Elements, especially those of the Enterprise Collaboration Architecture (ECA). The EDOC standard already proposes a mapping between RM-ODP basic concepts and the EDOC Profile Elements. However, it is insufficient since many RM-ODP concepts are not considered in this proposal and thereby it needs to be supplemented. We evaluate that this work does not present any major technical difficulty as it corresponds to the adaptation of the ODAC notation.

Another issue is the transformation of a PIM according to a PDM in order to get a PSM (see Figure 2). Let us recall that the PIM describes the functional aspects of the application, i.e., its business logic, with no technical consideration of the execution environment that will support it. The PDM describes the technical aspects of this execution environment, i.e., the facilities it provides to applications. The PSM is a configuration of a PIM for a given PDM. This transformation is a key issue for which the MDA identifies three basic ways, namely manual, semi-automatic and fully automatic. The latter is most interesting and implies the ability of models' storage and models' transformation. Regarding the models' storage, we already provide a MOF/XMI tool that is a meta-models repository [2]. Thus models are XMI files on which XSLT stylesheets can be applied in order to transform them. All these techniques can then be useful in the PIM transformation into a PSM. However, the difficulty is the identification of the transformation rules, which must express how the PIM and the PDM can be merged to constitute the PSM. This relates to the introduction of the non-functional properties described in the PDM in the PIM that deals with the functional concerns. For this, works from Aspect-Oriented Programming that deal with the weaving of aspects could help and should be investigated in order to study if the solutions existing at the code level would be adaptable at the model level. In this case, the transformation rules would establish the weaving of the PIM profile and the PDM profile. This issue is for further works.

6. Conclusion

We have presented in this paper the ODAC methodology that is originally based on the RM-ODP and that is currently adapted to become an MDA-oriented methodology. The availability of such a methodology would benefit to software designers because it would help them to construct the models of applications they are developing by using modeling tools as described in OMG Profiles such as EDOC Profiles or platforms-oriented Profiles (e.g., CORBA Profile or EFB Profile). Thus the methodology supplements the works on the MDA currently achieved at the OMG.

Future works are concerned with the key issue of the automatic transformation of the PIM into the PSM according to a PDM. Research direction is to investigate the Aspect-Oriented Programming in order to apply it at the model level.

7. References

[1] J. Bézivin and N. Ploquin, *Combining the Power of Meta-Programming and Meta-Modeling in the OMG/MDA Framework*, OMG's 2nd Workshop on UML· for Enterprise Applications, San Francisco, USA, December 2001

[2] X. Blanc, M.P. Gervais and R. Le Delliou, *On the Construction of Distributed RM-ODP Specifications*, In "New Developments in Distributed Applications and Interoperable Systems", Kluwer Academic Publishers, pp99-111

[3] S. Bouzitouna et al., *Création de services actifs dans ANTS*, 4ème Colloque francophone GRES, Marrakech, décembre 2001

[4] M.P. Gervais and F. Muscutariu, *Towards an ADL for Designing Agent-Based Systems*, in Proc. of AOSE'01, LNCS n°2222, Springer Verlag (Ed)

[5] M.P. Gervais, *Méthodologie ODAC : le guide de spécification comportementale*, LIP6 2001 / 024, nov. 2001

[6] ISO IS 10746-x, *ODP Reference Model Part x*, 1995

[7] ISO/IEC CD 15414, *ODP Reference Model: Enterprise Viewpoint*, January 2000

[8] J.R. Putman, *Architecting with RM-ODP*, Prentice Hall PTR, 2001

[9] OMG, *Model Driven Architecture, A Technical Perspective*, Document ab/21001-02-05, Februray 2001, http://www.omg.org

[10] OMG, *UML Profile for Enterprise Distributed Object Computing*, Document ptc/2001-12-04, December 2001

[11] OMG, *Unified Modeling Language Specification v. 1.4*, TC. Document ad/01-02-13, Februray 2001

[12] P. Spathis, K.-L. Thai, *Spécifications du multicast fiable*, livrable RNRT Amarrage-SP2-D2.3, Dec 2000 (in French)