Applying Model-Driven Development to Business Systems using RM-ODP and EDOC

Yoshihide Nagase
Technologic Arts Inc.
yoshi@tech-arts.co.jp

Daisuke Hashimoto
Technologic Arts Inc.
hashimoto@tech-arts.co.jp

Miwa Sato
Technologic Arts Inc.
msatoh@tech-arts.co.jp

Abstract

Improving development efficiency and maintainability for business systems requires a seamless development process, and both RM-ODP and MDA play a key role to this end. This paper shows our Model-Driven Development process in building business systems using RM-ODP and UML Profile for EDOC, with a case study of Electronic Health Record system models, and discusses several issues related to RM-ODP standard.

1. Introduction

As business systems are getting complex and large in recent years, their development efficiency and maintainability need to be improved. Especially, efforts to Total Optimization by SCM (Supply Chain Management) have a great influence on the way business systems are developed. To conduct Total Optimization, various information systems introduced to enterprise and supply chain need to collaborate with each other. In order to develop such information systems, it is required to model business processes and identify the roles of those systems in the entire business. Moreover, certain mechanism is necessary to transform the models seamlessly into implementation, which makes systems development more efficient with traceability among models.

2. MDA and RM-ODP Viewpoints

MDA, advocated by OMG (Object Management Group), is the technology that seamlessly reflects models to implementation. In MDA, models are developed from three perspectives: CIM (Computation Independent Model), PIM (Platform Independent Model), and PSM (Platform Specific Model). MDA is an abstract framework, thus concrete development processes are required for realization of MDA.

This paper shows overview of our Model-Driven Development process using RM-ODP [1] framework and notations defined by UML Profile for EDOC[2] (referring to it as EDOC from now on).

Our process uses RM-ODP Viewpoints as follows: business models are developed in Enterprise Viewpoint, information models in Information Viewpoint, and component models in Computational Viewpoint. These models are CIMs and PIMs in MDA. System architectures are defined in Engineering Viewpoint, and mapping rules are defined in Technology Viewpoint. In this development process, EDOC notations with defined semantics are used to define Enterprise, Information, and Computational Viewpoint specifications, since modeling elements, such as "process," "entity" and "component" required for describing those viewpoint specifications, were already standardized in the EDOC standard.

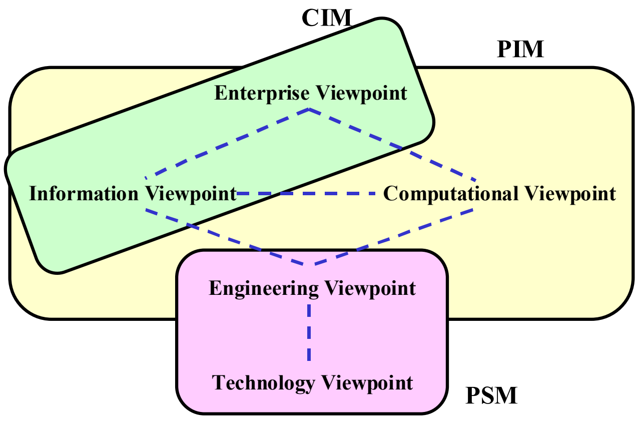

Figure 1. Relation between MDA and Viewpoints of RM-ODP

In the following sections, Electronic Health Record system models[34] are used as a case study for applying this development

[1] The standard frameworks for distributed object systems, standardized by ISO/IEC and ITU.
[2] The set of extended UML for distributed object systems, standardized by OMG. It extends notations for components and business processes.
[3] The Enterprise Viewpoint part of these models are an accomplishment of "The

process. This case study is an accomplishment of a project formed by two organizations: JAHIS, whose focus is on establishing approaches for developing standard components for Electronic Health Record systems, and INTAP, whose focus is on interoperability among enterprise systems with RM-ODP. Japanese government has been involved in and funds the project.

3. EDOC

EDOC includes a set of following profiles and those meta-models that extend standard UML for enterprise distributed object computing environments.

3.1 Component Collaboration Architecture (CCA) Profile

CCA Profile is a profile used to define computational components and interactions between those components. In this profile, Process Component is defined as a fundamental functional component. In interfaces of a Process Component, three kinds of ports (Flow, Protocol, and Operation) can be defined. With those ports, three kind of interactions can be specified: data flow interaction (simple data flow in and out via Flow Port), protocol flow interaction (two-way interactions via Protocol Port), and operational flow interaction (call/return type of interaction via Operation Port).

3.2 Entity Profile

Entity Profile is a profile used to define entity's structure of the target domain. Entity is a specialization of Process Component. Entity Data represents an aspect of Entity's data structure. Entity Data can have primary key and foreign key like relational database.

3.3 Business Process Profile

Business Process Profile is a profile used to define business processes. This profile extends CCA Process Component to define Business Process, Compound Task, and Activity etc., which together provide necessary semantics and notations to represent enterprise viewpoint process models. An Activity is a work to be done to complete a process, and may be associated with one or more of three roles, which are Performer, Artifact and Responsible Party. Compound Task is a container of Activities, and Business Process is the outermost container of the composition.

3.4 Event Profile

Event profile is a profile used to define business processes driven by business events and the mechanism of the state transition for business entities.

3.5 Relationship Profile

Relationship Profile is a profile used to define clearer relationship between model elements.

4. Enterprise Viewpoint

In the Enterprise Viewpoint, target business models and system requirements are modeled. The most important artifacts of this Viewpoint are business processes. Organizing business processes in the entire supply chain and enterprise clarifies the role of information systems. The first thing to do in the Enterprise Viewpoint is to define a scope for the system and divide it into smaller ones for a unit called Community. It is the unit to organize the scopes with their purposes. Defining Communities can make the size of the scope appropriate. Next, procedures for accomplishing purposes of each Community are defined as business processes. With Business Process Profile and Event Profile of EDOC, event-driven and non-event-driven business processes can be described, and information and functions used in the business process as well as Responsible Parties of Activity can be defined.

Modeling language with semantics of business processes was required to define business processes. Since business process semantics was not included in standard UML, we adopted EDOC for process modeling, instead of extending UML independently by ourselves. Therefore, Business Process Profile of EDOC was used in the development of Enterprise Viewpoint specification. Should we start this activity today, we will need to carefully watch and choose a right standard from BPDM, UML for ODP, and UML2.0 etc.

In this viewpoint, we did not make much use of developed policy statements. The issue was lack of policy statement language with clear syntax, semantics, and grammar. Expected are the emergence of policy statement language, and the structuring rule for policies with other enterprise model elements in the enterprise viewpoint specification.

Development of Electronic Health Record System through Standardizing Components" as a special science research theme awarded by the Ministry of Health, Labor and Welfare in the fiscal year 2002.

[4] The scale of those models includes 12 communities and 32 business processes.

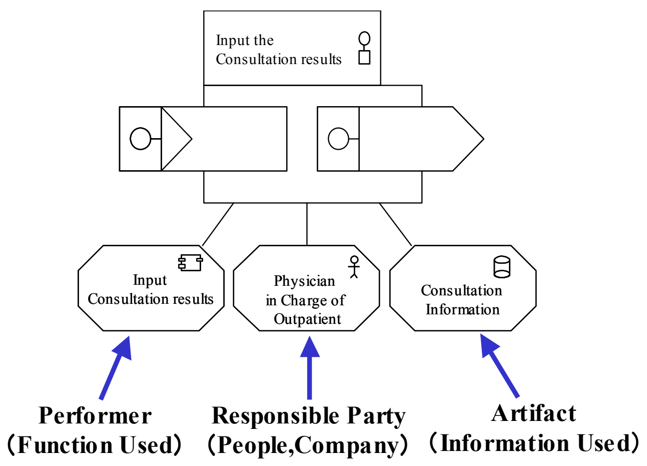
Input the Consultation results
Input Consultation results
Physician in Charge of Outpatient
Consultation Information
Performer (Function Used)
Responsible Party (People,Company)
Artifact (Information Used)

Figure 2. Example of Activity

5. Information Viewpoint

In the Information Viewpoint, information referred to and processed by the systems is modeled. Our development process defines that the details of information used in the business processes should be analyzed. The resulting information model elements of the system are described using Entity Profile in the EDOC. Later in this paper, the use of CCA Profile to describe Computational Viewpoint specification is explained. In order to integrate functional components derived from Enterprise Viewpoint specification with Entity Components from Information Viewpoint specification to complete the Computational Viewpoint specification, it is necessary to componentize Information Model with EDOC.

The use of the Entity profile enables description of primary key and foreign key of entity implemented using relational database, and define access path to the target information by creating Entity Components. Constructing information as components is effective in preventing reduction of performance, since it identifies the gate of information to control the number of remote accesses.

In this viewpoint, we did not consider dynamic addition or deletion of model elements. However, if dynamic change is included in Enterprise Viewpoint Model, such as dynamic creation of a Community with new roles introduced, information viewpoint model should also accommodate the corresponding dynamic changes within the model.

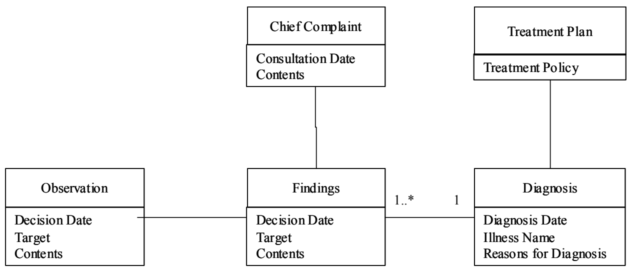

Figure 3. Example of Information Model (Consultation)

6. Computational Viewpoint

In the Computational Viewpoint, functions of systems are modeled. Our process defines that the details of functions used in the business processes (i.e. performers and artifacts) and entities should be analyzed, and the functions of systems are described as Process Components of CCA Profile in the EDOC. Therefore Computational Objects in RM-ODP are represented as Process Components of CCA.

If collaboration exists among enterprises and/or among information systems, it is described as interaction among Process Components. Note that interfaces of components are discovered based on analysis of Information Viewpoint.

In the EDOC, collaborations are described by connecting components through Ports. If the functions of systems require persistent information (database information), the Process Components are connected to the Entity Components. Operation procedure of the connected components (both process and entity) is defined as Protocol and described using State Machine diagrams of UML.

It should be noted that in RM-ODP the basic unit for encapsulation is object, and the interaction between objects are made by Signal, Operation, and Flow. Although we believe CCA Process Component can represent Computational Object in pragmatic way, it would be preferable to have necessary modeling concepts for components and interactions between them (messaging) in RM-ODP itself (a discussion on the use of object versus component is described in 9.1).

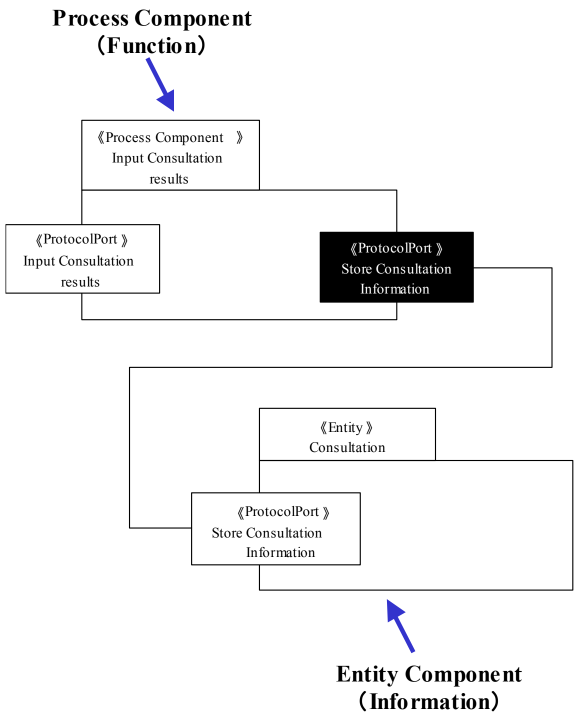
Process Component (Function)
《Process Component 》 Input Consultation results
《ProtocolPort 》 Input Consultation results
《ProtocolPort 》 Store Consultation Information
《Entity 》 Consultation
《ProtocolPort 》 Store Consultation Information
Entity Component (Information)

Figure 4. Example of Component Model

7. Engineering and Technology Viewpoints

In the Engineering Viewpoint, system architectures are designed. The design includes deployment configuration of components, transaction, and security.

It should be noted that there are choices in mapping Component Model onto platforms. One choice is on platform styles, and another choice is on configuration of nodes. Figure 5 was our choice for pilot implementation, but there were other possibilities, e.g. client node could host only GUI application and network interface components, and all other components may be hosted on one or more server nodes. Also the same component model could have been mapped onto web services platform or CORBA platform.

Another observation is that it is this viewpoint that shows portions of human-system interaction with Client node box. The model of human-system interaction should also be described in relevant viewpoints in addition to the Engineering viewpoint model.

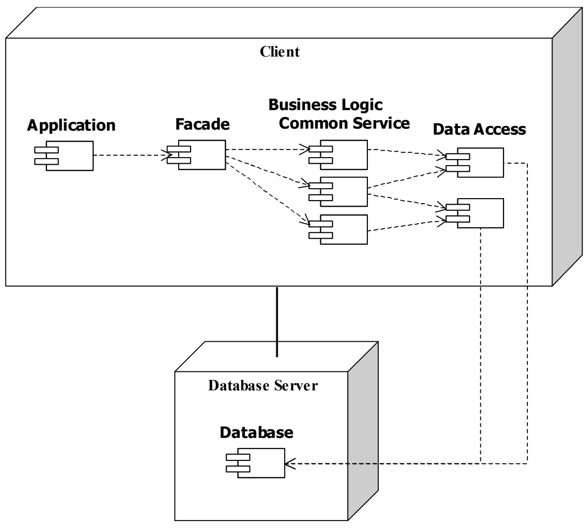

Figure 5. Example of Deployment Configuration of Components

In the Technology Viewpoint, PIM as the artifact of all the processes is mapped to a specific platform to derive PSM from it. The following example is one of mapping rules of EDOC to .NET. The mapping rules were used in the pilot development of Electronic Health Record system.

EDOC (PIM)	.NET (PSM)
Process Component 《ProcessComponent》	Mapped to .NET assembly. Select assembly type according to requirement such as deployment, performance, and security.
Protocol 《Protocol》	Mapped to interface.
Interface 《Interface》	Mapped to interface.
Protocol Port 《ProtocolPort》	Mapped to interface.
Operation Port 《OperationPort》	Initiator: mapped to method call. Responder: mapped as interface method.
Flow Port 《FlowPort》	In case of flow port included in operation port: Initiator: mapped to argument of method. Responder: mapped to returned value of method.
Entity Component 《Entity》	Mapped to .NET component using ADO.NET.
Entity Data 《EntityData》	Mapped to ADO.NET DataSet defined by XML schema.
Key 《Key》	Mapped to constraint defined by XML schema.
Foreign Key 《ForeignKey》	Mapped to constraint defined by XML schema.

Table 1. Example of Mapping Rules (EDOC to .NET)

In Table 1, interfaces provided by the Process Components and various Ports included in the components are mapped to interfaces of .NET components and their methods. The Entity Components are mapped to .NET components that use functions of ADO .NET. Remote accesses among components may be required depending on deployment configuration of components in Engineering Viewpoint. In this case, Web services and remoting functions of ASP .NET should be used to realize the remote accesses.

8. Viewpoint correspondence

To realize the seamless development and to ensure the traceability, we need to elaborate on viewpoint correspondence. In RM-ODP and Enterprise Language standard, viewpoint correspondences are defined but are not very useful in practice. In EDOC, because of its design, Business Process Profile, Entity Profile, and Event Profile are defined as specializations of CCA Profile, and thus there exists stronger correspondence between those and CCA Profile. Once the mapping from those to CCA Profile is done, and the final model is represented in CCA Profile, the resulting model will be considered as a Computational Viewpoint Model. In our development process, EDOC/CCA-based viewpoint correspondences are applied to those viewpoint specifications.

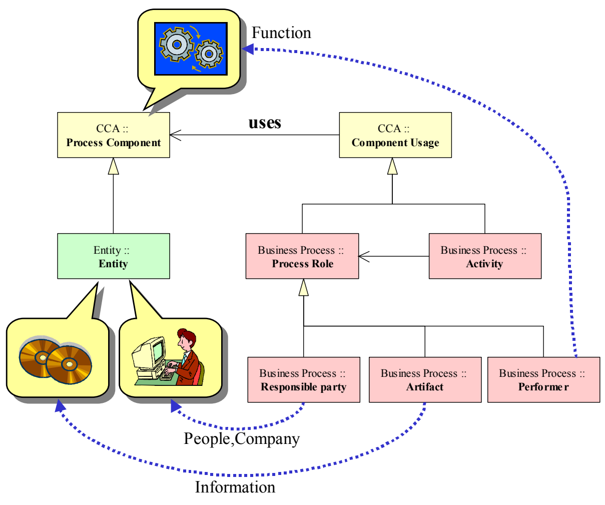
Function
uses
CCA :: Process Component
CCA :: Component Usage
Entity :: Entity
Business Process :: Process Role
Business Process :: Activity
Business Process :: Responsible party
Business Process :: Artifact
Business Process :: Performer
People,Company
Information

Figure 6. Relationship of EDOC Profiles

9. Discussion

Based on the experience of the above explained project, it is the our belief that the following issues should be considered at the workshop.

9.1 Computational Object vs. Component

In this development process, Computational Object is represented as a Process Component. However, object and component are not completely the same. For instance, according to reference [11], component's characteristics are described as "A coherent package of software implementation that 1) has explicit and well-specified interfaces for the services it provides; 2) has explicit and well-specified interfaces for services it expects from others; and 3) can be composed with other components, perhaps customizing some of their properties, without modifying the components themselves. As a consequence of these properties, a component can be independently developed, delivered, and deployed as a unit."

Computational Object of RM-ODP satisfies the characteristics 1) and 3), but not the characteristics 2). In addition, some components may work only within specific platform (or framework), and that platform (framework) might require certain components as prerequisite, or might require specific interface type for components. Computational Viewpoint is a viewpoint where computational aspects of the systems are the major focus, and we believe that an ODP system comprising loosely coupled objects, which require messaging based interactions, plays an important role in today's internet world. It is, therefore, desirable to have not only object-based computation, but component-based computation. In this case, we propose to introduce a basic concept of component into Computational Language ([3] instead of [2]). Note that this proposal has close relationship with the next proposal regarding messaging.

9.2 Communication by message

Communication or interaction between objects in Computational Viewpoint is basically synchronous with RPC style. However, in the real world (e.g., in business-to-business transactions), asynchronous communication, or message-based interaction, is in fact used widely. If we only need to provide modeling capability for asynchronous communication, the discussion of property for the Binding Object will work. The difference between synchronous communication and asynchronous communication in Computational Viewpoint appears when describing a behavior of a client. While a client is blocked by issuing requests in synchronous communication, a client is allowed to continue processing in asynchronous communication even after issuing requests. The client in this case may need to monitor the status of the result in parallel to its processing in a different thread. For instance, in the case of Web Services, service provider may make two types of Web Services descriptions public: synchronous and asynchronous. And, the behavior of the client is quite different. The following is the issues for RM-ODP.

1) Computational Language provides interfaces for objects which provide services, but does not provide interfaces for object requiring services. The behavior of the client is dependent on this "interface for object requiring services." and it is hard to specify the client's behavior without having this concept.

2) There should be a standard way of describing structure of activities at the user side of interfaces. If activity structure section of [2] is assumed for this purpose, appropriate statement

should be added in Computational Language.
3) At this interface description level, there may be a necessity to introduce message management interface such as handling messages in a queue.

9.3 Concept for composite interaction

Three kinds of interactions are defined in Computational Language: Signal, Operational, and Flow. However, the concept of composite interaction, combining these different types of interactions, is not defined clearly. This concept will be useful if the interaction by two parties, like two companies in a typical business-to-business business process scenario, is represented as interactions between two Computational Objects, since the interaction may contain all the interaction patterns. It might be possible to consider Flow as the concept for this purpose, if you interpret the definition of Flow in [3] literally. If this is the case, the standard should be clear enough to state this. If this is not the case, the concept for composite interactions should be added in the Computational Language. Note that EDOC/CCA's model element called Protocol manages the composite interactions, and UML2 has Protocol State Machine as a similar model element.

9.4 Dynamic evolutions of models

The dynamic evolutions of models means that model elements may be added or changed or deleted at some point to evolve into the model in the next phase. It may be viewed as a part of the lifecycle of models or versioning of models. For example, in the Enterprise Viewpoint Language, dynamic creation and deletion of communities are described. Other possibility includes dynamic addition of Enterprise Objects, Roles, Processes, and so on. If Enterprise Viewpoint Model evolves in such a way, corresponding viewpoint models should also evolve. In Information Viewpoint, Information Object might be added or deleted, and defined schemata may also evolve. In Computational Viewpoint, the whole model might change. It will certainly impact the engineering and technology viewpoint models. We will need to discuss:
1) how these evolutions or changes (transition) can be described in each viewpoint, and
2) what concepts in which viewpoint is required to achieve this.
If we were successful, we may apply the result to "as-is model" and "to-be model" in Enterprise Architecture today.

9.5 Human-System interactions

In the Viewpoint Languages ([3] and [4]) today, there seems to be no place for describing Human-System Interaction. However, [2] defines "Perceptual Reference Point" under "Class of Reference Points" (see [2], section15.3.2). We believe it important to clarify the position of the standard. The discussion we would like to have is whether we would like to introduce new concepts around human-system interactions. Even if the result of it leads to take no action, there is a need to provide a guideline or an official statement on this point. If we could have concept(s) for this purpose in viewpoint languages, the following may be a possible candidate where the concept(s) may contribute to.
1) Specification of Human-System interactions for the system providing similar services through multi-channels (Web, telephone, fax, e-mail, and letter)
2) Screen design and screen flow for a Web-based application
Note that there is a similar issue regarding "Interchange Reference Point."

9.6 Policy language

The current policy related concepts are those of meta-model level concepts, and may be used as policy categories, but those are not detailed or concrete enough to apply in the real world specifications. The work done in network management area may be of interest to us. The followings are several modeling elements (minimum) to be considered.

The policy will be initially described in natural languages. Therefore, the following elements may be required, as a minimum, to create a complete policy statement with existing policy concepts: subject, verb, object, and condition. If we tried to associate with Enterprise Viewpoint Language concepts, the following may apply to some cases.
1) subject <-Enterprise Object | Role | Community
2) verb <- Action including Obligation and so on| Process
3) object <- Enterprise Object | Role | Community
4) condition <- Predicate | Guard Condition
Policy statement example:
"If an emergency patient is brought into the hospital (Condition), a doctor on duty (subject: Role Doctor fulfilled by Enterprise Object Person) has an obligation (Obligation) to make a diagnosis of the patient (verb: Action 1, object: Role Patient fulfilled by Enterprise Object Person) or find a suitable doctor to request a diagnosis for the patient (verb: Action 2, object: Role Doctor fulfilled by Enterprise Object Person)."

10. Summary

10.1 Model-Driven Development

Our concrete process worked well to develop business systems by Model-Driven Development, resulting in successfully applying our process to an Electronic Health Record system. It turned out that applying development frameworks of RM-ODP enabled seamless development from business models to program codes. Especially in developing business systems that aim for Total Optimization, this framework is certainly the best since the scope of collaboration can be considered as distributed objects. The use of EDOC also enables to describe various aspects of business systems in detail. As EDOC has great affinity for RM-ODP, it is the most appropriate modeling language in building CIMs and PIMs. Since this case study of Electronic Health Record system is implemented as the pilot system, our next goals are to use our process to implement an Electronic Health Record system of practical scale, and to apply our process to systems with other domains for its refinement.

10.2 Possible revisions to the standard

We have identified several issues for RM-ODP standard, and included some of our suggestions in this paper. It is our hope that those issues be considered, discussed, and resolved at the workshop, or passed to ISO/IEC and ITU-T RM-ODP group as a part of issues list for RM-ODP revision work.

Reference

[1] ISO/IEC IS 10746-1, *Open Distributed Processing -Reference Model: Overview* 1998.

[2] ISO/IEC IS 10746-2, *Open Distributed Processing -Reference Model: Foundations*, 1996.

[3] ISO/IEC IS 10746-3, *Open Distributed Processing -Reference Model: Architecture* 1996.

[4] ISO/IEC 15414, *Open distributed processing - Reference model -Enterprise language*, 2002

[5] UML Profile for EDOC Part I, Document number: ptc/2004-02-01

[6] UML Profile for EDOC Part II, Document Number: ad/01-08-20

[7] Model Driven Architecture, Document number: ormsc/01-07-01

[8] MDA Guide, Document number: omg/03-06-01

[9] JAHIS Web Site, http://www.jahis.jp/english/english.html

[10] INTAP Web Site, http://www.net.intap.or.jp/e/

[11] D.F. D'Souza and A.C. Wills, *Objects, Components, and Frameworks with UML (The Catalysis Approach)*, Addison-Wesley, 1998

[12] Moore, B., Ellesson, E., Strassner, J. and A. Westerinen, *"Policy Core Information Model - Version 1 Specification"*, RFC 3060, February 2001.

[13] Moore, B., Ed., *"Policy Core Information Model Extensions"*, RFC 3460, January 2003.

[14] Distributed Management Task Force, Inc., *"DMTF Technologies: CIM Standards CIM Schema: Version 2.8"*

[15] Distributed Management Task Force, Inc., *"Common Information Model (CIM) Specification: Version 2.2"*, June 14, 1999