Specifying distributed object applications Using the Reference Model for Open Distributed Processing And the Unified Modeling Language

J-M. Cornily,
M.Belaunde

France Télécom
Centre National d'Etude des Télécommunications
Technopole ANTICIPA
2, Avenue Pierre Marzin
22307 Lannion
France

Tel : + 33 2 96 05 15 63
Fax : + 33 2 96 05 39 45
E-mail : jeanmichel.cornily@francetelecom.fr, mariano.belaunde@cnet.francetelecom.fr

Abstract

This paper outlines an approach for using UML as an alternative notation for modeling potentially distributed systems according to the standardized ITU-T G.851-01 specification process[1]. The G.851-01 process, which has been elaborated in the context of telecommunication management networking, is based on the RM-ODP viewpoints architecture. It proposes a top-down scheme in which the requirements of the system are firstly captured (enterprise viewpoint), then the conceptual data of the system is modeled (the information viewpoint) and then the potentially distributed processing units are defined (computational viewpoint).
The paper defines a UML profile for each viewpoint, which provides a precise guideline for applying UML notations as well as the use of well-defined modeling concepts.
The increasing acceptance of UML in telecommunication domain is promising : it will probably make easier tool support for software process engineering.

1. Introduction

Network operators manage lots of data and applications. What they call their information system is made up of databases and software applications which, due to their history and organization, is highly heterogeneous. This heterogeneity takes place at different levels and separation between application aspects and communication aspects is often not well established. In this paper, we firstly introduce the Reference Model for Open Distributed Processing which constitutes a very powerful architectural framework for specifying and modeling potentially distributed systems. Then we present the specification process, based on the RM-ODP viewpoints architecture, that has been elaborated and standardized by ITU-T in the context of telecommunication management networking [G.851-01]. When standardized, that specification process made use of ad hoc notations which, due to the growing success and the standardization of the Unified Modeling Language [UML], have now to be replaced. This paper presents how to use UML as an alternative notation for G.851-01, together with elements on how to define UML profiles for G.851-01 viewpoints and requirements for a software process engineering facility enabling G.851-01 process enactment.

[1] This work was partially sponsored by the French RNRT project PILOTE

2. The Reference Model for Open Distributed Processing

2.1 Scope

The RM-ODP is usually defined as a framework of abstractions, i.e. a set of modeling concepts relevant for the specification of ODP systems and an architecture providing a structure for these concepts. RM-ODP objective is to define modeling concepts for potentially distributed systems that are totally independent of any existing analysis / design method or notation or programming language. The modeling concepts are object based ; they are enough abstract and precisely defined to enable a possible mapping on to any of our favorite methods or languages. The architecture is essentially based on the definition of a number of viewpoints providing different perspectives on the ODP system.

2.2 Base modeling concepts

To our knowledge, the RM-ODP is one the major initiatives[2] to standardize definitions of such key and base modeling concepts as object, type, class, action, super-type / sub-type, super-class / sub-class, parent class / derived class, etc. These concepts are generally part of the meta-model of any method but may have different semantics. For example, the C++ keyword 'class' means that a type (in the sense of RM-ODP) is being declared or defined, whereas the RM-ODP class concept is not supported by the C++ language. All these modeling concepts are common to the whole set of viewpoints introduced here below and constitute the so-called foundations ; in addition, viewpoint-specific modeling concepts are also defined.

2.3 Viewpoints-based architecture

Five viewpoints compose the architecture of the RM-ODP, as illustrated in figure 1. Every viewpoint focuses on particular aspects of the ODP system and provides specific concepts to describe these aspects whereas other aspects are out of scope. The viewpoints are (intentionally listed in alphabetical order as RM-ODP gives no sequencing between viewpoints) :

- the computational viewpoint : focuses on the definition of potentially distributed functional units together with their communication interface(s). The computational viewpoint specifications are made by and for distributed application designers;
- the engineering viewpoint : focuses on the description of the platform that makes it possible for functional units identified in the computational viewpoint to actually communicate. The engineering specifications are made by and for system designers;

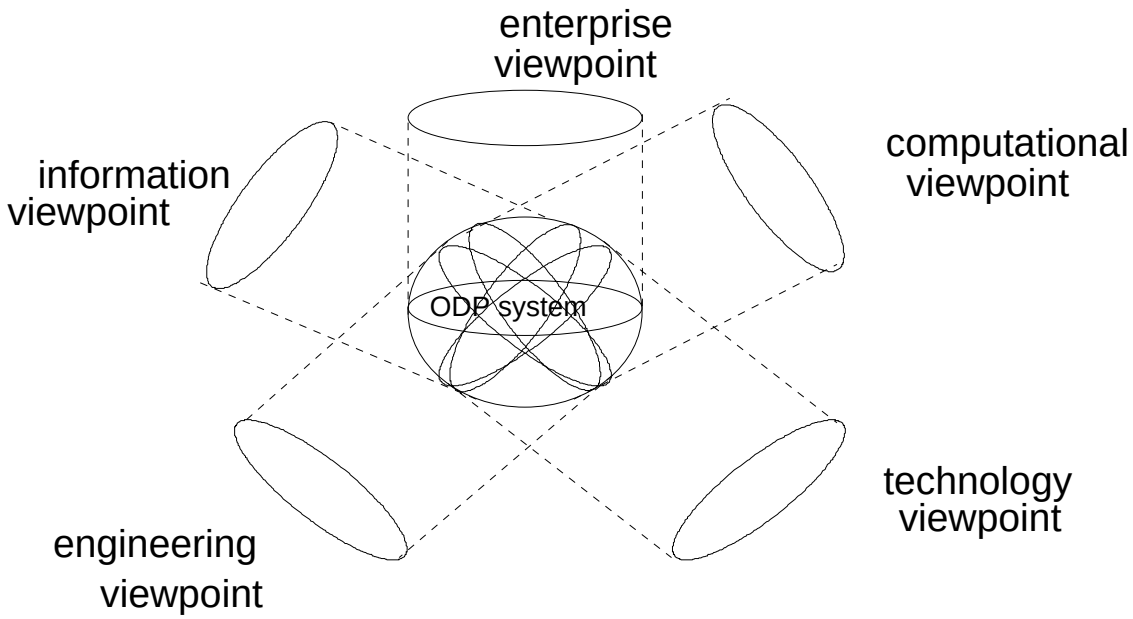

Figure 1. RM-ODP viewpoints architecture

[2] Of course, UML, from the OMG, is also a major contribution in that domain.

- the enterprise viewpoint : focuses on the describing the ODP system with regard to its functionality, how it takes place in its environment, how it interacts with it and what are the business requirements and management / usage rules that govern its design. The enterprise specification is made by and for decision-makers;
- the information viewpoint focuses on the definition of the data that is present in the ODP system, its possible states and transitions. The information viewpoint is made by and for information system designers;
- the technology viewpoint focuses on defining which actual components are being used in terms of operating systems, database management systems, programming languages, communication protocols, etc. The technology viewpoint is made by and for developers.

The five RM-ODP viewpoints enable to partition the specification of the ODP system ; depending on the people to whom one is presenting the ODP system, the appropriate viewpoint language must be selected. RM-ODP does not prescribe the use of any concrete language, i.e. syntax, though it defines the vocabulary and its associated semantics ; nor RM-ODP claims to be an object-oriented analysis and design method : viewpoints may be used separately or as a whole but, in the latter case, no process is prescribed by the RM-ODP.

2.4 From RM-ODP to G.851-01

It is commonly agreed that a specification method must include the three following ingredients :

1. A set of modeling concepts;
2. A (set of) notation(s);
3. A process.

The Reference Model for Open Distributed Processing provides a very complete and powerful set of modeling concepts. It does not prescribe a particular notation for viewpoint languages, nor it forces to use its viewpoints in a precise way.

In the context of the specification of standardized network-level management services in ITU-T, a process based on RM-ODP viewpoints architecture has been invented and standardized. Also, ad hoc notations were developed and standardized. Chapter 3 introduces the G.851-01 specification process based on the RM-ODP viewpoints.

3. The ITU-T G.851-01 distributed systems specification process

Figure 2 depicts the essentially top-down process that was developed and standardized in ITU-T for the specification of standardized network management services.

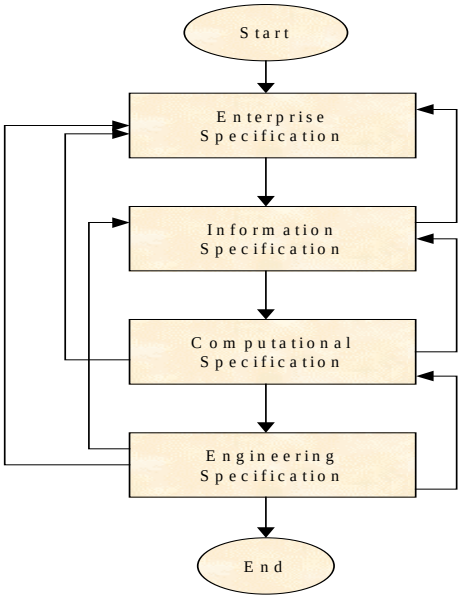

Figure 2. The G.851-01 modeling process.

This process is not specific to network management but can be applied to the specification of any open distributed system. As said earlier, the G.851-01 process is essentially top-down as it was invented to specify and standardize new network management services.

Naturally, the enterprise viewpoint is the appropriate place to capture requirements, in terms of what is expected from the service defined. The service is defined as a contract between a client and a provider. Expected behavior of both the client and the provider are defined together with the externally observable behavior. Provided they respect the contract policies, what the service does is defined. Enterprise viewpoint language concepts include objects having roles and forming a community ; policies are attached to roles.

Secondly, the information viewpoint specification defines the data that characterize the ODP system as well as data processing. Information viewpoint language concepts include schemas which can be either static, dynamic or invariant. The information viewpoint pays no attention at whether the ODP system is distributed or not ; the resulting object model is conceptual. For a given enterprise specification, a number of information specifications can be imagined.

Then the computational viewpoint specification defines the potential for distribution, i.e. it describes units that are subject to distribution and their interfaces. Computational viewpoint language concepts include interfaces which can either have a server or a client role, operations associated to the interface and the operations signature. The computational viewpoint is the first place in our process where distribution scenarios may be defined. The computational viewpoint does not deal with how communication can actually occur between distributed objects. For a given information specification, a number of computational specifications are possible.

Finally, the engineering viewpoint specification describes the infrastructure needed to make things defined in the computational viewpoint (e.g. interactions between computational objects) happen. Engineering viewpoint concepts include transparency objects such as stubs, protocol objects and binder objects. For a given computational specification, a number of engineering specifications may exist.

In the context of standardization, the technology viewpoint is out of scope as it is out of question to fix which techniques, tools are going to be used by different companies. A part of the value added of each company consists in realizing the best technology choices.

4. UML as the unifying notation all along the G.851-01 process

The G.851-01 specification approach, though widely accepted by most telecommunication companies, suffers a number of issues that need to be solved :

- The enterprise and computational viewpoints notations are purely invented from scratch and, thus, do not enable to promote the process in other areas having the same characteristics such as intelligent networking. In the same spirit, the information and engineering viewpoints notations, are adaptations from OSI Systems Management notations such as the General Relationship Model [GRM] and the Guidelines for the Definition of Managed Objects [GDMO];
- There are not, and will never be, off-the-shelf CASE tools that support this set of notations, hence companies have to develop their own tool.

For these reasons, the new emerging Unified Modeling Language, standardized in the OMG, was the best candidate to replace the whole set of G.851-01 notations. The following sections illustrate, on a viewpoint basis, what are the relevant modeling concepts and what are the relationships that exist between them and those from the UML meta-model.

We propose hereafter to introduce the methodology by illustrating it on a concrete example. The system being considered is typical in telecommunications as it provides means to establish and release connections across an equipment (e.g. a switch or a cross-connect). This example will serve all along the specification process. For every viewpoint, we first present an example of what kinds of UML constructs are relevant to that viewpoint. Then, considering that the UML notation either offers too many concepts or lacks other ones for our purpose, we propose to define a UML profile per viewpoint.

4.1 The profile mechanism

The OMG Process Working Group recently produced a white paper on the Profile mechanism [PROFILE]. Without going into details, one can say that four items are necessary to define a profile to UML :

- Selection of relevant concepts : consists of selecting the subset of concepts of the UML meta-model that is sufficient for our purpose. Additionally, extensions might also be necessary to introduce new modeling concepts using UML extensibility mechanisms. These extensions have to be identified. The result of that process is a superset of a subset of the UML meta-model;
- Presentation rules : prescribes how the modeling concepts of interest have to be presented at the GUI (colors, icons, etc.);
- Validation rules : consists of a list of integrity constraints that make the profile self-consistent;
- Transformation rules : consists of a set of rules stating how UML profile modeling concepts must be mapped on to specific targets, e.g. for generating CORBA IDL or Java.

In parallel, OMG domain / task forces are currently issuing RFPs to get e.g. a UML profile for CORBA, a UML profile for Enterprise Distributing Object Computing or a UML profile for Real-time applications. For our purpose, related to network management standards specification, we are envisaging to define UML profiles for G.851-01 viewpoints-based specifications. These are presented hereafter. We note that the transformation rules are not dealt with here as, though being very important, they will be elaborated later on another document (e.g. transformation between modeling concepts from different viewpoints).

4.2 Enterprise viewpoint specification

4.2.1 Example

The enterprise viewpoint specification can be captured using a *Use Case diagram*.

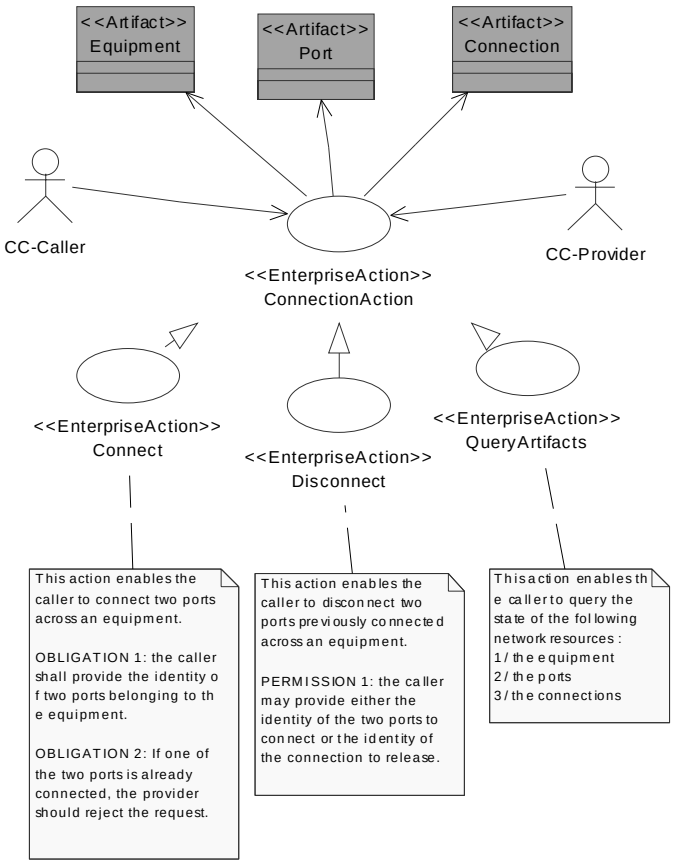

Figure 3. Example enterprise viewpoint specification using UML

As shown by figures 3 and 4, the enterprise community is modeled as a *use case model* inside which roles, actions and policies are respectively represented by *actors* / *classes*, *use cases* and *notes* (*comments*). As an illustration, RM-ODP agent roles such as the *CC-Caller* and *CC-Provider* are modeled using an UML actor ; as, by definition, they are active roles, the interaction is oriented from them towards the use case. Artifact roles such as *Equipment*, *Port* and *Connection* are represented by UML classes stereotyped <<*Artifact*>> ; Being passive roles, the interactions are oriented from the use case to the artifacts. Enterprise actions such as *Connect*, *Disconnect* and *QueryArtifact* are modeled using UML use cases stereotyped <<*EnterpriseAction*>>. For sake of clarity, all these use cases are represented as specializations of the abstract use case *ConnectionAction*. Attached to every entity, a note can capture policies ; for example, a structured documentation is attached to the use case *Connect* capturing both the purpose of the action and the policies stating behavioral constraints of actors with regard to the action *Connect*. For sake of readability, we choose to put them in a single note. Of course, each policy may alternatively be specified as a separated UML constraint (a note with brackets around the text).

4.2.2 UML Profile for the enterprise viewpoint specification

4.2.2.1 Selection of relevant concepts

The following UML meta-model classes are necessary : *Package*, *UseCase*, *Association*, *Actor*.

The following extensions are needed :

- *EnterpriseAction*, defined as a stereotype of *UseCase*,
- *Artifact*, defined as a stereotype of *Class*.

4.2.2.2 Presentation rules

Some presentation rules might be prescribed, like :

1. Create a UML package to encompass the whole enterprise community specification
2. Define inside that package the three kinds of entities : use cases, artifacts and actors
3. Use a special fill color for display of enterprise artifacts.

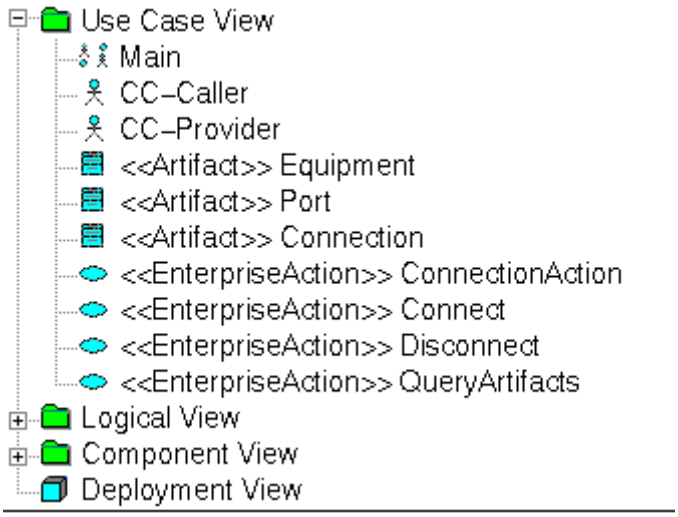

Figure 4. Enterprise viewpoint specification presentation.

4.2.2.3 Validation rules

None.

4.3 Information viewpoint specification

4.3.1 Example

The information viewpoint specification defines the information that represents the system. In UML, we use both class and state diagrams.

In the class diagram shown in figure 5, all object classes are stereotyped <<*InformationObjectType*>> to indicate clearly that they are special kinds of object classes , i.e. they take place only at the information viewpoint. Clearly, the information object types that are defined here are *Equipment*, *Port* and *Connection* which are the result of the object-oriented modeling of artifacts (respectively *Equipment*, *Port* and *Connection*) identified in the enterprise viewpoint specification. To represent this tracing relationship, a dependency link stereotyped <<*trace*>> is drawn between every information object type and its corresponding artifact(s). Information object types are defined by both attributes and operations. At this stage, operations are only named stimuli, with no parameters, associated to state transitions.

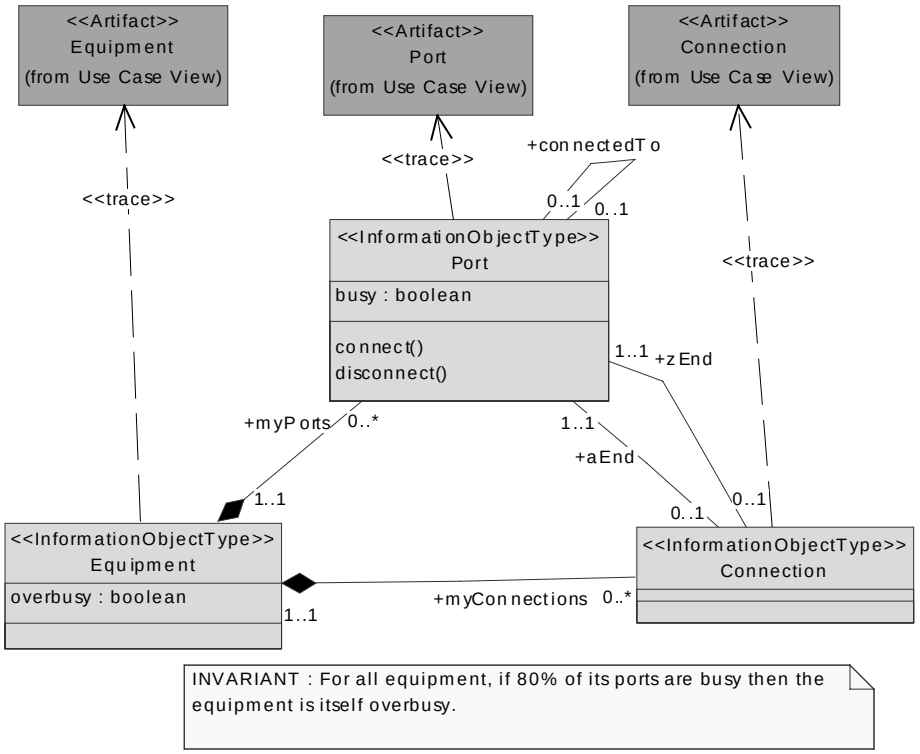

Figure 5. Example information viewpoint specification using UML – class diagram

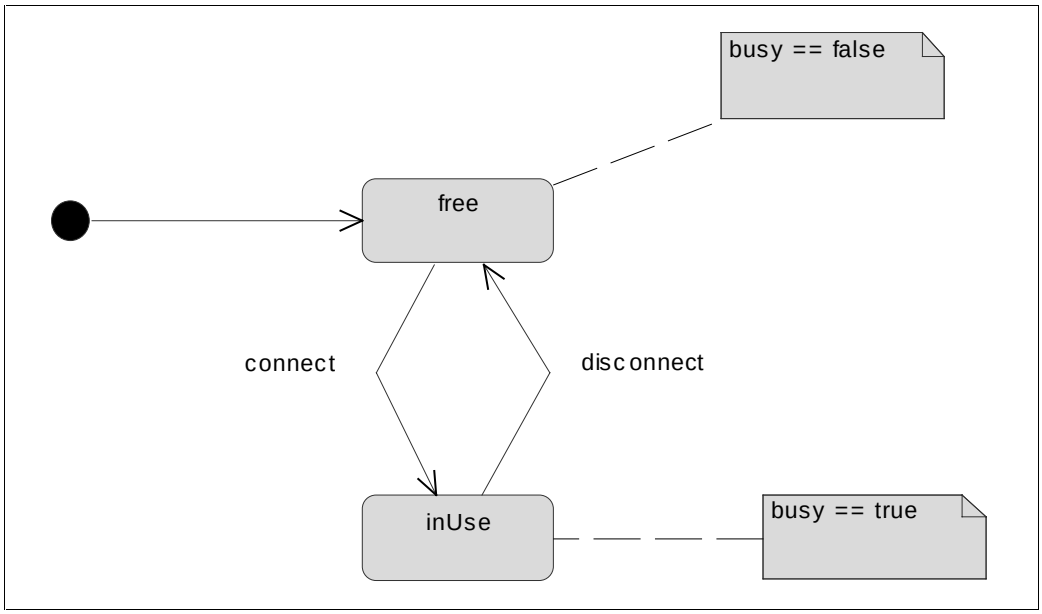

Figure 6. Example information viewpoint specification using UML – state diagram of the class Port

Figure 6 shows the state machine attached to the class Port. States are defined by a label (e.g. *free* or *inUse*) and the relationship between the state and the attributes of the information object type is captured through a note. For example, the state « free » corresponds to the case where the value of the attribute « *busy* » is false, though it could imply several attributes.

4.3.2 UML Profile for the information viewpoint specification

4.3.2.1 Selection of relevant concepts

The following UML meta-model concepts are selected :
Package, *Association*, *StateMachine*, *State*, *Attribute*, *Operation*, *Trace Dependency* stereotype.

Additional concepts are needed such as :

- *InformationObjectType*, defined as a stereotype of the UML meta-class *Class*

4.3.2.2 Presentation rules

Information viewpoint modeling concepts such as *InformationObjectType* and *State* are displayed using a fill color which is different from the other viewpoints ones. Also, a package is defined to gather the whole information specification.

4.3.2.3 Validation rules

Every information object type must be associated via a dependency link to one (or more) enterprise artifact(s). This is a means to ensure that no information is introduced by the designer, which would not be justified by a business requirement.

4.4 Computational viewpoint specification

4.4.1 Example

The computational specification defines functional units that are potential for distribution. More precisely, it consists of the definition of a number of computational object types together with their various interfaces. Every interface is described by the list of the operations that may be invoked through this interface (see figure 7). Every operation is defined by its name, its parameters defined each by their name and type and its return type.

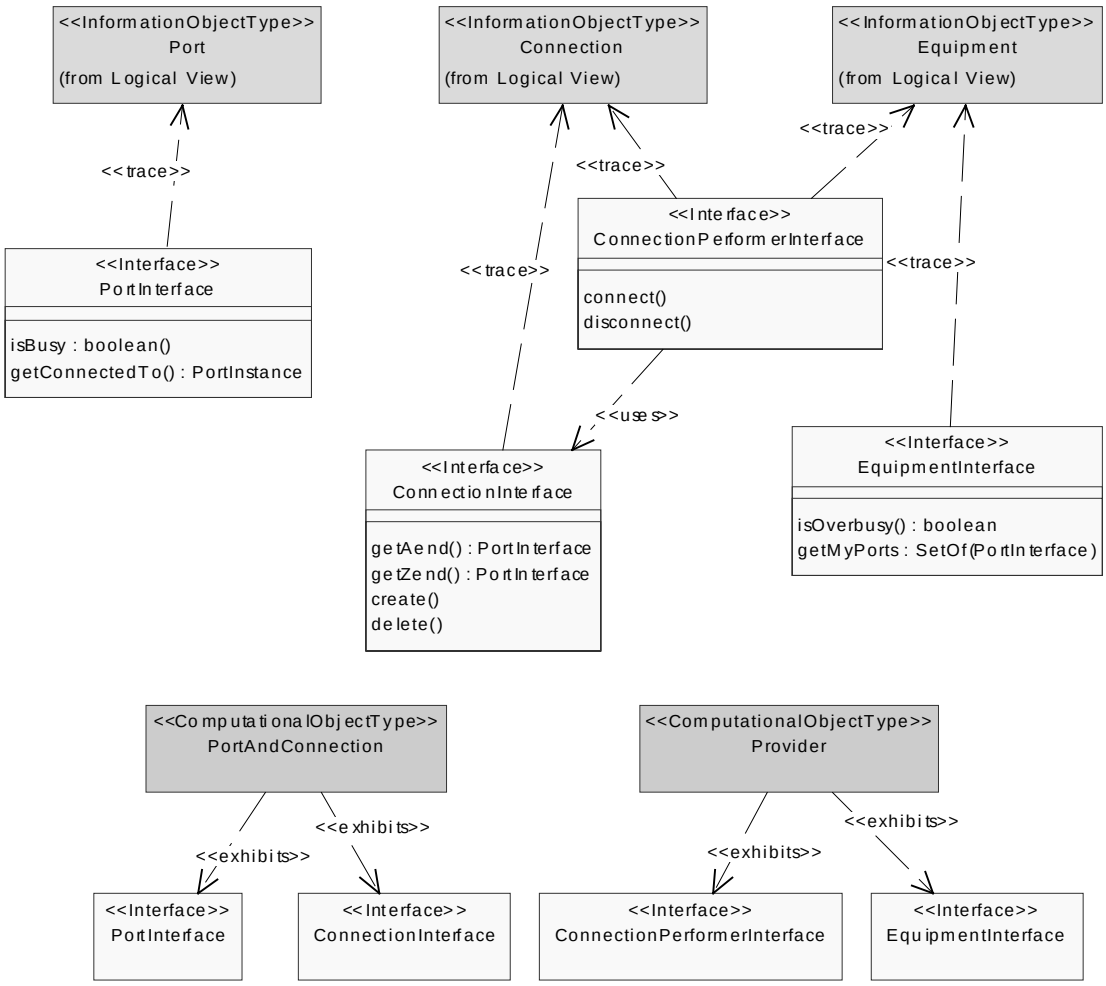

Figure 7. Example computational viewpoint specification using UML

Whereas interfaces are the units of designation in RM-ODP, computational objects are units of distribution. The relationship between interfaces and the information that they encapsulate must be captured. This is done through the dependency relationship stereotyped *<<trace>>*. As an illustration, one can see on figure 7 that the operational interface type *EquipmentInterface* encapsulates and thus provides for access to the information object type *Equipment.* More complex, the operational interface type *ConnectionPerformerInterface exhibited* by the provider computation object type, encapsulates and gives indirect access to both the information object types *Connection* and *Equipment*. That interface has operations which behavior specification implies the creation and deletion of connections. As the computation object type *Connection* is not part of the same distribution unit, the operational interface type *ConnectionPerformerInterface* has to interact with the interface *ConnectionInterface* to request for creation / deletion of relevant computational objects.
A unit of distribution (a computational object) provides support for one or more interfaces. This is specified using a dependency stereotyped *<<exhibits>>*.

4.4.2 UML Profile for the computational viewpoint specification

4.4.2.1 Selection of relevant concepts

The following UML meta-model concepts are selected for the computational viewpoint specification : *Package*, *Interface*, *Usage and Trace Dependency* stereotype.
The following modeling concepts are introduced :

- *ComputationalObjectType*, defined as a stereotype of *Class*,

- *exhibits*, defined as a stereotype of *Dependency*.

4.4.2.2 Presentation rules

Computational viewpoint modeling concepts such as *ComputationalObjectType* and Interface are displayed using a fill color which is different from the other viewpoints ones. Also, a package is defined to gather the whole computational specification

4.4.2.3 Validation rules

None

4.5 Engineering process

The resulting computational viewpoint specification makes no assumption on how the basic engineering objects that are the implementation of computational objects are going to be grouped together into clusters, capsules, nodes. For sake of simplicity, we will not detail this in the paper. Also, this part of the work, dealing with the study of what UML profile is needed for capturing engineering concerns, is still in progress. However, figures 8 and 9 provide for an illustration of how a given computational specification can be mapped on to different architectures.

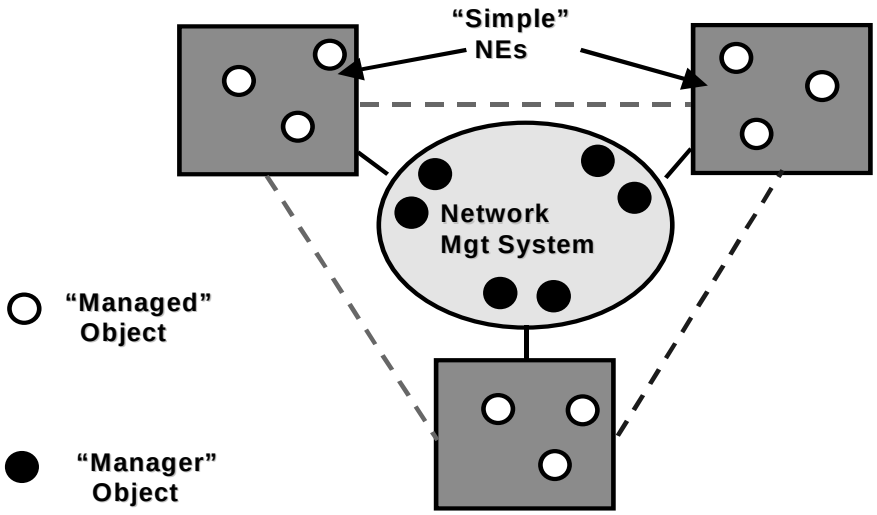

Figure 8. A computational model superimposed over a centralized physical architecture.

This is shown on a concrete example in the context of the Telecommunication Management Network (TMN). A management application specified using the G.851-01 methodology leads to the definition of distribution-independent computational objects which can be mapped, according the deployment scenarios, on to different implementation solutions. The first scenario (cf. figure 8) illustrates the case where a centralized management system manages the whole set of network elements ; in that case, communication mechanisms must be provided to enable the actual communication between the « manager objects » and the « managed objects ». The second scenario (cf. figure 9) shows a completely distributed architecture where no more centralized management system remains ; conversely, « management objects » are widely distributed over network elements ; of course, a communication protocol between distributed objects has to be specified. This is part of the engineering viewpoint specification.

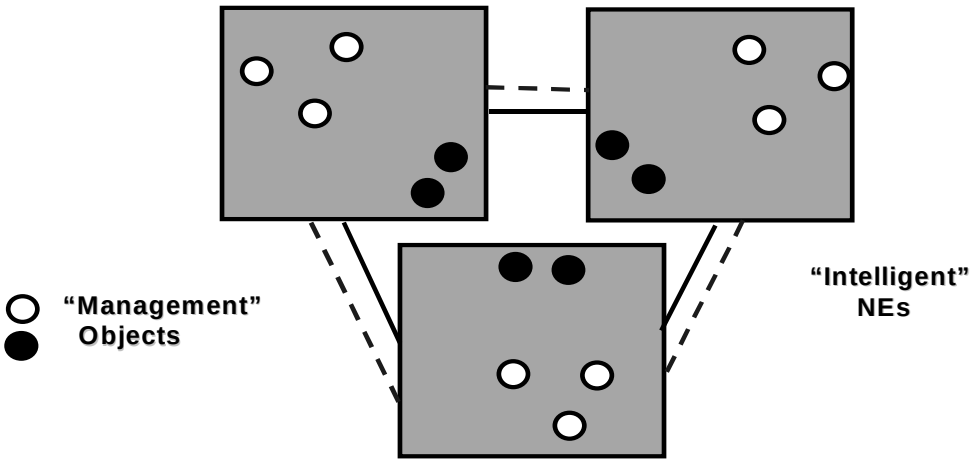

Figure 9. Same computational model superimposed over a distributed physical architecture.

5. Relationship with OMG work on process

Whereas the models that have been presented earlier on in this paper take place in the layer M1 of the OMG layered modeling architecture, work is on progress to reify these specifications. A draft result is presented by figure 10 below, in the case of the computational viewpoint.

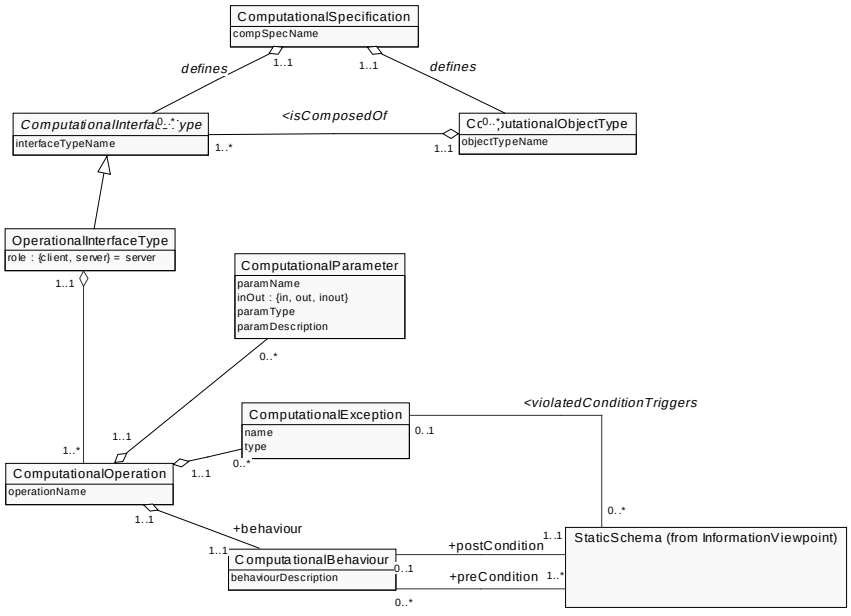

Figure 10. The computational viewpoint specification meta-model.

This model (M2) defines the structure of a computational viewpoint specification which is a work product produced by the application of the G.851-01 process. We expect that the set made up of the model of the G.851-01 process, the models of the work products (enterprise, information and computational viewpoint specifications), and the UML profiles definition that we presented for every viewpoint will serve as inputs to a Software Process Engineering Facility under study in the Process Working Group of the OMG. The ultimate goal is to make possible that any UML CASE tool might be driven by this facility. Of course, this technique may be applied to any modeling process, other than G.851-01. This leads to the definition of the functional architecture depicted in figure 11. Such a facility would enable to :

- define UML profiles ("Profile Modeling") and store them in a repository ("Profile Repository");
- provide means to define, compose, specialize, etc. processes ("Process Modeling") and store them ("Process Repository")

- retrieve, from a Workflow product, any process model from the Process Repository (to which is associated a (set of) profile(s) corresponding to the work products associated to the process) and activate any UML CASE tool.

Standardizing the interfaces between these functional blocks would be of interest for the future.

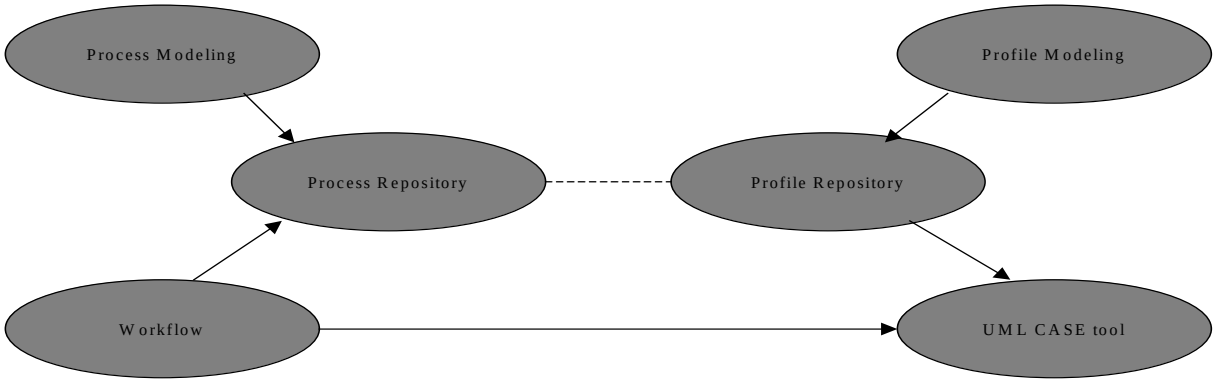

Figure 11. Software process engineering facility functional architecture.

6. Conclusion

In this paper, we have presented an effective marriage between RM-ODP, G.851-01 and UML, each one providing the necessary ingredients to build a methodology (respectively a set of modeling concepts, a process and a notation). We have shown how the main benefits of the ITU-T G.851-01 modeling approach is being dealt with UML : separation of concerns with viewpoints, traceability and transparency with regard to distribution. For every G.851-01 viewpoint, we have defined a UML profile. Finally, as no methodology can exist without supporting tools, we have presented what information a software process engineering facility must consider as input for driving the modeling activity done with a CASE tool ; this is often referred to as process enactment.

7. References

[G.851-01] ITU-T Recommendation G.851-01 : Management of Transport Networks - Application of the RM-ODP framework

[GDMO] ITU-T Recommendation X.722 : Open System Interconnection - Information Technology – Guidelines for the Definition of Managed Objects

[GRM] ITU-T Recommendation X.725 : Open System Interconnection - Information Technology – General Relationship Model

[RM-ODP] ITU-T Recommendation X.901 : Open System Interconnection - Information Technology - Open Distributed processing - Overview and Guide to Use

[RM-ODP-2] ITU-T Recommendation X.902 : Open System Interconnection - Information Technology - Open Distributed processing – Foundations

[RM-ODP-3] ITU-T Recommendation X.903 : Open System Interconnection - Information Technology - Open Distributed processing – Architecture

[RM-ODP-4] ITU-T Recommendation X.904 : Open System Interconnection - Information Technology - Open Distributed processing - Architectural Semantics

[UML] UML 1.3 : UML Specification– Object Management Group

[PROFILE] White Paper on the Profile mechanism – Version 1.0 – Object Management Group – Analysis and Design Task Force – OMG Document ad/99-04-07, April 1999

[CORNILY-1] « An RM-ODP based approach to modeling distributed systems and associated tools » – M. Belaunde, J-M. Cornily, E. Debeau – International Conference on Software Engineering and its Applications – Paris, Dec. 1998

[CORNILY-2] « Application of ODP Modelling Techniques to SDH Network Management » - J-M. Cornily, D. Doherty, J. Ellson, P. Mullan, C. Pageot-Millet, T. Stéphant – IEEE Globecom – Houston, 1993

[CORNILY-3] « Application des techniques des systèmes répartis ouverts à la gestion des réseaux SDH » - J-M. Cornily, T. Stéphant – Echo des Recherches N° 161 - 1995