UML 2 Models for ODP Engineering/Technology Viewpoints – An Experiment

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

Hiroshi Miyazaki
Fujitsu, Ltd.
miyazaki.hir-02@jp.fujitsu.com

Akira Tanaka
Hitachi, Ltd.
tanakaak@itg.hitachi.co.jp

Abstract

The advance of UML® 2.0 standardization work by OMG™ provides a good opportunity for ODP community to leverage UML 2.0 to show the value of ODP. In this paper, we examine issues in applying UML 2.0 to ODP Engineering and Technology Viewpoint Languages, and show how we may be able to use UML to represent those ODP Viewpoint specifications.

1. Introduction

With the wide acceptance of Unified Modeling Language [2] in the industry, there is a growing interest in applying UML to represent ODP [1] viewpoint specifications. This direction is beneficial to both sides, since ODP modelers will eventually be able to get ODP modeling tool based on UML tools, and UML modelers will get a robust way of organizing their UML models. This comes from ISO/IEC and ITU-T's joint project called "Use of UML for ODP system specifications," and also comes from various "Enterprise Architecture [6]" practices where rows in two-dimensional matrix are usually quite similar to ODP viewpoints. The combination of UML and ODP with OMG's MDA initiative [3][4] will create a good foundation for systems' lifecycle management. The content of this paper is a work-in-progress level, based on INTAP's technical report [5], and is a result of elaborating current Committee Draft of "Use of UML for ODP system specifications" standard by our group. In this paper, we identify and examine issues in modeling Engineering and Technology Viewpoints with UML 2.0, and propose one possible UML 2 model diagrams for those viewpoint specifications.

2. Engineering Viewpoint – Issues and discussion for UML modeling

2.1 Target Architectural Diagrams

First we need to identify kinds of architectural diagrams to be described in UML. Some candidates are found in RM-ODP Part 3, Clause 8 Figure 2 to 6. The following figures (R2 to R8) are extracts from RM-ODP Part 3 standard document. In order to avoid confusion, we will add "R" to refer to those diagrams.

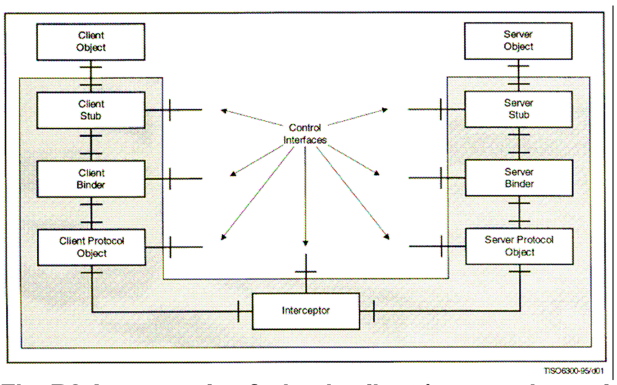

Fig. R2 An example of a basic client / server channel

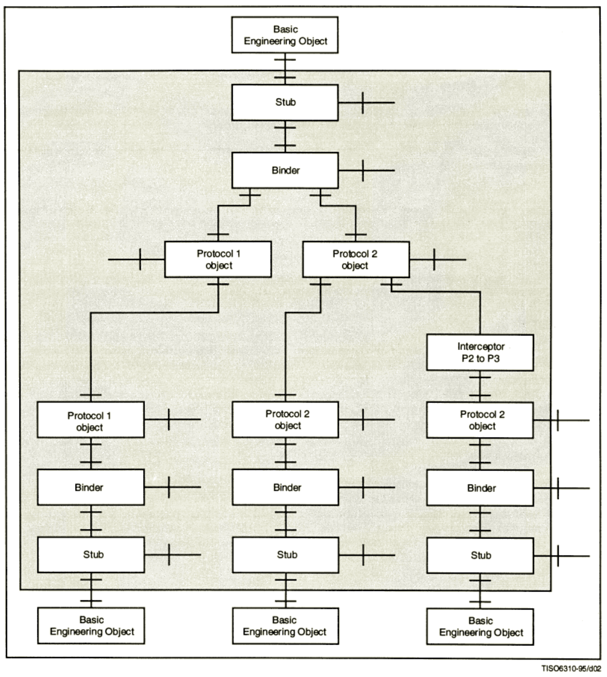

Fig. R3 An example of a multi-endpoint channel

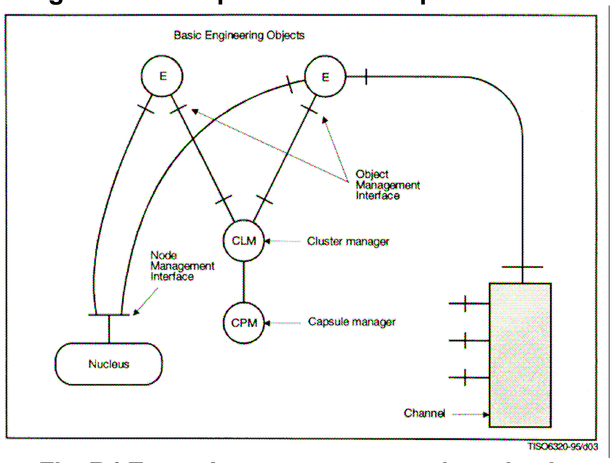
Basic Engineering Objects
E
E
Object Management Interface
CLM
Cluster manager
Node Management Interface
CPM
Capsule manager
Nucleus
Channel
TISO6320-95/d03

Fig. R4 Example structure supporting a basic engineering object

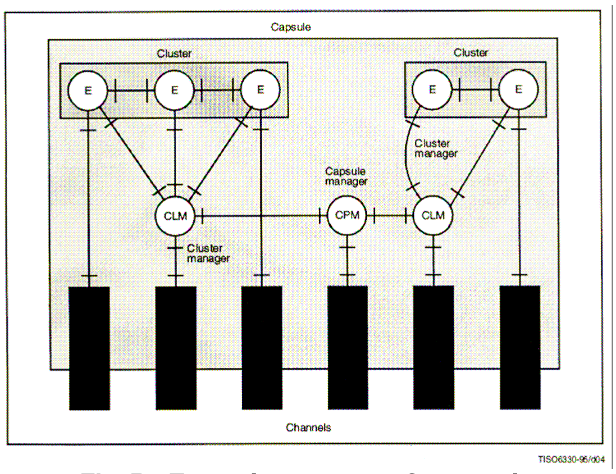

Fig. R5 Example structure of a capsule

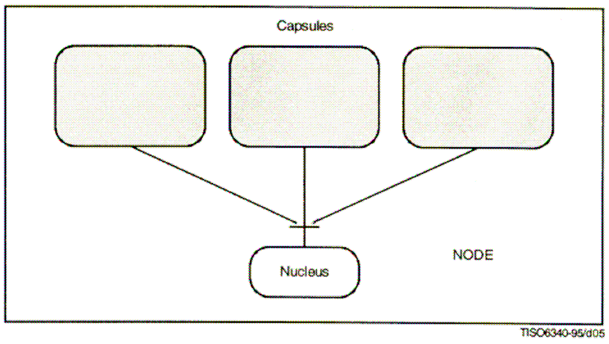

Fig. R6 Example structure of a node

From those example figures, we can identify essential target diagrams, which are Channel structure diagram (covering Figure R2 and R3) and Capsule structure diagram (covering Figure R4, R5, and R6).

In addition, to cover all the Engineering concepts, we will need UML diagrams describing domains and templates.

2.2 Representation of Engineering Concepts

Engineering Object:

The issue is the choice of UML elements representing Engineering Object, which includes at least Basic Engineering Object (we will refer it as BEO in this paper), CapsuleManager, ClusterManager, Stub, Binder, ProtocolObject, and Interceptor. It could be modeled with UML Object (InstanceSpecification of Class), UML Class, UML InstanceSpecification of Component, or UML Component. In either case, UML 2's Structured Classifier will help us describe the internal structure of Channel and Capsule.

Containers:

The next are containers such as Node, Nucleus, Capsule, and Cluster. Those elements are defined as "a configurations of (basic) engineering objects …" in RM-ODP Part 3. It may be possible to use UML Node to represent ODP Node. However, if we take this approach, only available UML diagram will be Deployment Diagram, and UML 2 only allows certain types of modeling elements (e.g. Node, Artifact) to be placed within a Node. Note that it was possible to take this approach with UML 1.4, since it was legal to place Component within a Node. The UML diagram we need has to have a capability to show the internal and logical structure of ODP Node, similar discussion regarding Class or Component in Engineering Object may also apply.

Nucleus:

Nucleus may either be treated as Engineering Object or container. It could be Engineering object, since it provides Nucleus Services to all types of Engineering Objects. It could also be considered as a container, in which Capsule and Containers may reside within.

Channel:

Channel may either be treated as a container or a structured Object or Class etc. The issue associated with Channel is that both Channel and Capsule shares the same Engineering Objects (Stub, Binder, and ProtocolObject) and we do not have overlapping diagrams even in UML 2.0.

Template:

<X> Template is defined in RM-ODP Part 2 as "The specification of the common features of a collection of <X>s in sufficient detail that an <X> can be instantiated using it. <X> can be anything that has a type." The templates in Engineering Viewpoint Language are Cluster template, Checkpoint, Cluster checkpoint. However, the definitions of those Engineering templates seem more like "snapshot" than the Part 2 definition implies. The choice of UML model element for template has impact on the choice of UML model element for Engineering Object.

2.3 Interactions between Engineering Objects

In Computational Viewpoint, RM-ODP Part 3 provides various concepts to deal with interactions with interface and signature. In Engineering Viewpoint, however, we do not have specific concepts for this purpose, and that makes it difficult to model interactions between Engineering Objects. One possibility is to consider "recursive application of viewpoints." If we can apply Computational Viewpoint Language to interactions between BEOs, we will get a capability to specify this. In that case, the modeling element will represent Engineering Object with Computational aspects. The issue becomes how to do this in UML. For instance, do we want to allow UML Class or Component or InstanceSpecification stereotyped as "«NV_BEO» «CV_Object»?"

In addition, if we are to allow "recursive application of viewpoints" as described here, we would need to consider the alignment of base classes for corresponding viewpoint profile.

2.4 Engineering Functions

In Engineering Viewpoint, there are functionalities defined as a part of its language. Those are Checkpointing, Deactivation, Cloning, Recovery, Reactivation, and Migration. Those are different from other concepts, since they are representing functions possibly including behaviors. One possibility is to apply "recursive application of viewpoints" again, and make use of Enterprise Viewpoint's Objective and Process etc. concepts to model those functions as «EV_Objective» with behaviors as «EV_Process» expressed with Activity Diagram.

2.5 ODP functions

In RM-ODP Part 3, many ODP functions are described. When defining ODP viewpoint specifications, those common functions may need to be explicitly included. An example of ODP function is ODP Trader, where the ODP Trader standard defines Enterprise, Information, and Computational Viewpoint specifications of itself. The issues are how we can identify, reference, and include those functions in Engineering Viewpoint specifications.

2.6 Domains

Domain concept in RM-ODP is defined as follows. <X> Domain is "A set of objects, each of which is related by a characterizing relationship <X> to a controlling object. Every domain has a controlling object associated with it." It is a set of objects, rather than a configuration of objects. The issue is which UML element is suitable for representing this "a set of objects" concept.

Also, since the same Engineering Object may be a member of several different kinds of domains at the same time (e.g. NamingDomain_1, SecurityDomain_2, and PolicyDomain_3 etc.), UML element representing domain should allow sharing of objects. If we choose Class, parts may be shared. If we choose Component, it may become issues. If we choose Package, we will need to use «import»/«access» etc. to have access to those elements contained in other Packages.

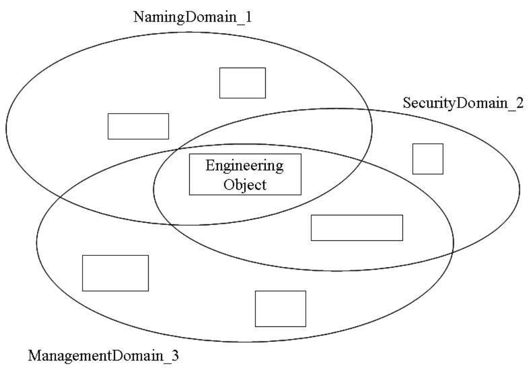

Figure 1 Engineering objects and domains

2.7 Selective Transparencies

Computational Viewpoint specifications are defined in distribution transparent manner, and the degree of distribution transparency may be specified in a "transparency schema" associated with "a specification that uses specific ODP functions and engineering structures to provide the required form of masking." This may be considered as a mapping specification from Computational Viewpoint to Engineering Viewpoint. And, this implies that given one Computational Viewpoint specification, there will be a variety of Engineering Viewpoint specifications, each corresponding to different transparency schema. Although this issue may be mainly related with Computational Viewpoint, it has an impact on Engineering Viewpoint specification and Engineering Viewpoint's correspondence to Computational Viewpoint specification.

2.8 Architectural Styles (or Engineering Templates or Patterns)

Since the day RM-ODP Part 3 became International Standard, five or more years have passed, and now there are various types of commercial and open source middleware which in fact implements most of the Engineering Viewpoint concerns, e.g. CORBA ORBs, J2EE application servers. .NET, Web Services, SOA, and Message Oriented Middleware, etc. Even more, there are new types of middleware emerging for Grid and Utility Computing, Wireless Networking, and Collaboration Environment etc. Also, various best practices and architectural styles (such as MVC) were developed and in use. Now may be a good time to define architectural styles as Engineering Templates and publish them for use in mapping Computational Viewpoint specifications to Engineering Viewpoint specifications. The issue is how to define those architectural styles with UML.

3. Engineering Viewpoint – Possible UML 2 Models

3.1 Profile definition

The first step is to decide which UML element should be used to represent Engineering Viewpoint Language, especially Engineering Object. We also need to consider about related templates. The choice we made in this paper is use of UML Component for Engineering Objects. A Component is closer to real world software component than Class, and UML 2 also provides Component with structuring capability. The following is a Class diagram showing partial UML 2 Profile definition. The stereotypes have prefix NV_, which is a rule for defining stereotype names for Engineering Viewpoint in Use of UML for ODP system specifications standard. Most (not all) stereotypes extend metaclass Component.

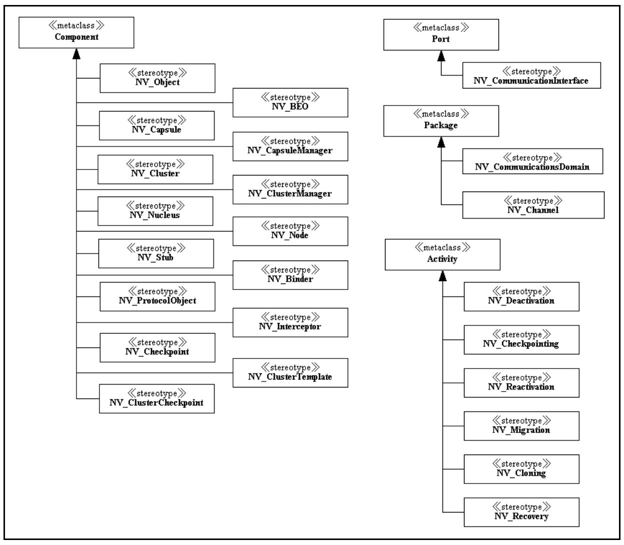
«metaclass» Component
«stereotype» NV_Object
«stereotype» NV_BEO
«stereotype» NV_Capsule
«stereotype» NV_CapsuleManager
«stereotype» NV_Cluster
«stereotype» NV_ClusterManager
«stereotype» NV_Nucleus
«stereotype» NV_Node
«stereotype» NV_Stub
«stereotype» NV_Binder
«stereotype» NV_ProtocolObject
«stereotype» NV_Interceptor
«stereotype» NV_Checkpoint
«stereotype» NV_ClusterTemplate
«stereotype» NV_ClusterCheckpoint
«metaclass» Port
«stereotype» NV_CommunicationInterface
«metaclass» Package
«stereotype» NV_CommunicationsDomain
«stereotype» NV_Channel
«metaclass» Activity
«stereotype» NV_Deactivation
«stereotype» NV_Checkpointing
«stereotype» NV_Reactivation
«stereotype» NV_Migration
«stereotype» NV_Cloning
«stereotype» NV_Recovery

Figure 2 UML 2 Profile for Engineering Viewpoint Language

3.2 Node configuration

The computational to engineering correspondence, or model transformation, is not within the scope of this paper. However, to achieve this, we will at least need a transformation pattern and mechanism. Assuming that there is one, the output N-tier distributed system's node configuration may look like the following. In this case, there are a Node for user interaction, a Node for front-end, a Node responsible for business logic, and two Nodes for persistent data. In essence, BEOs derived form computational objects and engineering objects for providing specified transparency scheme will be deployed on those nodes.

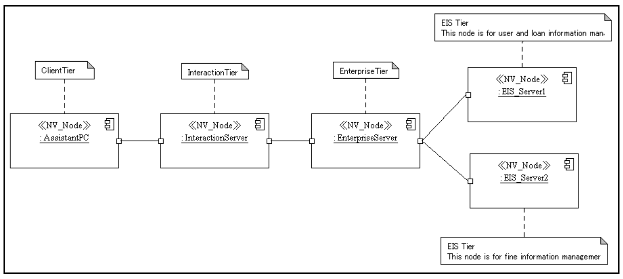
EIS Tier
This node is for user and loan information man
ClientTier
InteractionTier
EnterpriseTier
《NV_Node》 :AssistantPC
《NV_Node》 :InteractionServer
《NV_Node》 :EnterpriseServer
《NV_Node》 :EIS_Server1
《NV_Node》 :EIS_Server2
EIS Tier
This node is for fine information managemer

Figure 3 Example Node Configurations

3.3 Node Structure Diagrams

As discussed before, we need to be able to model structure of each Node. Sample structural UML Diagram is shown in Figure 4, which is a UML Component Diagram. We have a Node containing two Capsules, one of which has two Clusters containing two BEOs. There are BEOs (e.g. BEO_1AA) which have access to the services provided by Cluster Manager, which in turn has access to services provided by Capsule Managers. Other BEOs (e.g. BEO_1AB) have access to a Stub for communication, and the Stub interacts with a Binder, and the Binder interacts with a Protocol Object.

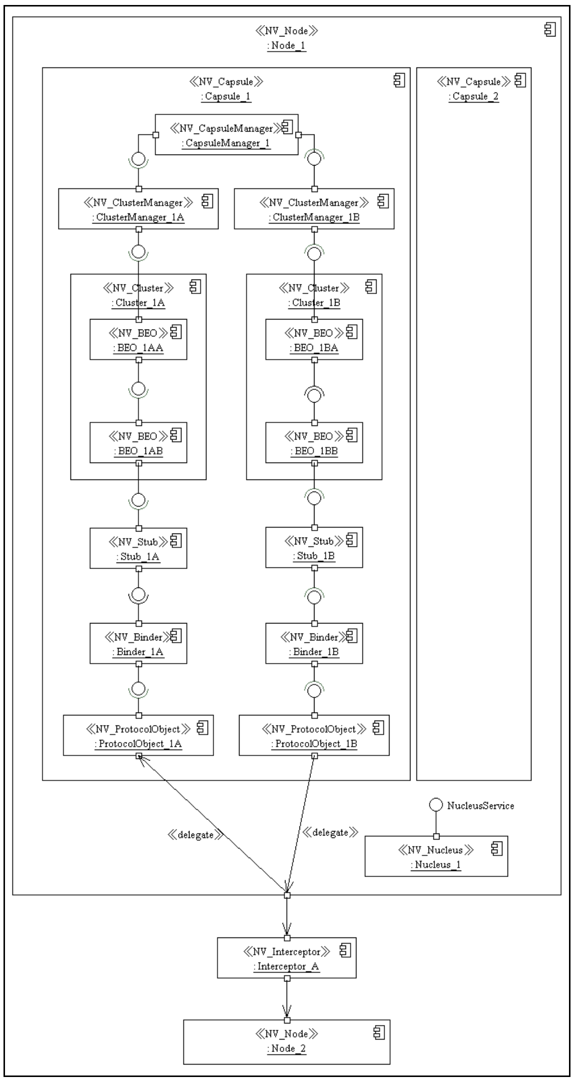

Figure 4 Example UML 2 model for Node, Capsule, Cluster, and BEO

Interesting observation is whether Capsule and Cluster should have interfaces or not, since Capsule is a unit of resource assignment, Cluster is a unit of activations/deactivations, and they have associated Managers. If they have interfaces, then what kind of interface should they be?

3.3 Channel Diagrams

A Channel may be represented with the following UML 2 diagram. The same Engineering Objects appear in Figure 4 and Figure 5 (i.e., Stub, Binder, Protocol Object and Interceptor). Figure 4 shows the structure, interfaces with ports, and services in the hierarchy. Figure 5 shows membership of the Channel Package. Those two diagrams complement with each other to provide different views to the same set of Engineering Objects that make up a part of Node and a Channel. In Figure 5, a channel is defined as a package containing engineering objects, which are the components necessary for enabling communication between Nodes. There are three BEOs involved (not shown): a BEO interacting with Stub_1, a BEO interacting with Stub_2, and a BEO interacting with Stub_3. This channel is defined to serve for those BEOs interacting with each other. The structural aspect, e.g. Stub_1 interacts with Binder_1, is described in structural diagram like Figure 4, and therefore this channel diagram just defines the member engineering objects.

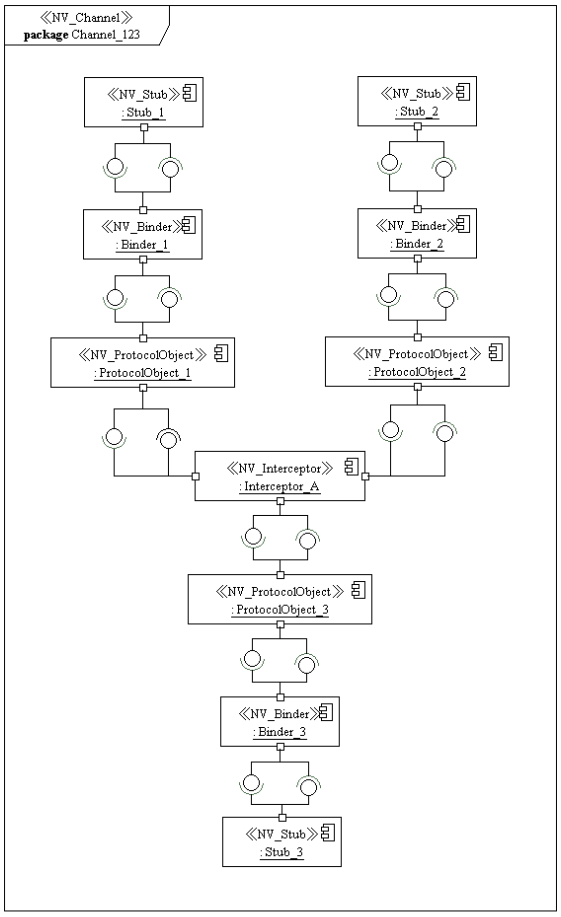

Figure 5 Example UML 2 model for Channel

Figure 4 and 5 together cover most of the figures (R2 to R8) from RM-ODP Part 3, except for control interfaces, which are just one kind of interfaces we can add to engineering objects.

From other perspective, a channel diagram is a package importing necessary member engineering objects from Node structure packages for communication.

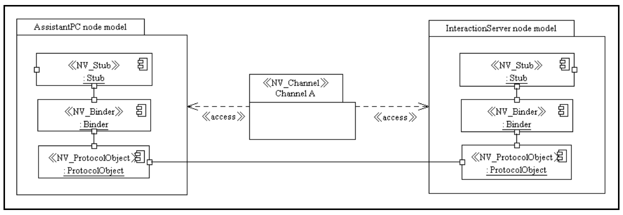

Figure 6 Example Channel

3.4 BEO configuration

Like in computational viewpoint, it is sometimes necessary to specify interactions between BEOs. One of the issues is that the diagram will become complex if BEO configuration is placed in Node Structure diagram (Figure 4). One possibility is to isolate Clusters from the Node Structure and use component diagram (in our case) to describe the interaction. We may need to use e.g. double stereotypes to represent computational aspects of engineering object, e.g. for interfaces, signatures, and interactions for engineering objects. In Figure 7, a Cluster contains two BEOs providing services through the Cluster's Port and Interface.

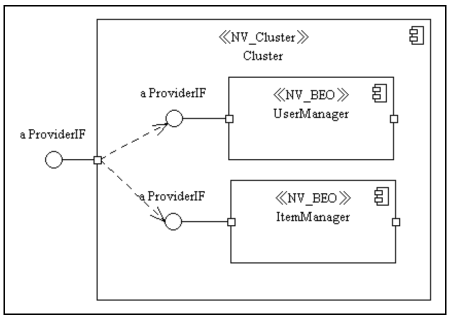

Figure 7 Example basic BEO Configuration

Figure 8 shows a case where two BEOs in a same Cluster have dependency, or interaction. If we need to specify this interaction, interface, and signatures, we may need to borrow stereotype elements from already defined Computational Profile.

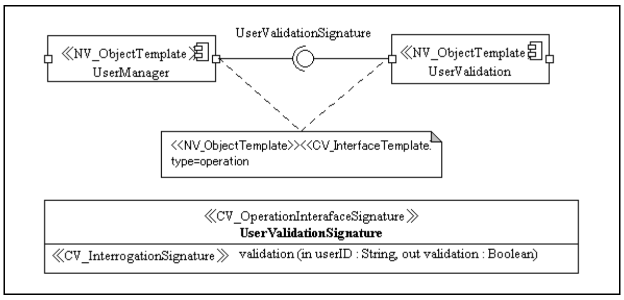
UserValidationSignature
《NV_ObjectTemplate》
UserManager
《NV_ObjectTemplate》
UserValidation
<<NV_ObjectTemplate>><<CV_InterfaceTemplate>>
type=operation
《CV_OperationInterafaceSignature》
UserValidationSignature
《CV_InterrogationSignature》 validation (in userID : String, out validation : Boolean)

Figure 8 Example BEO configuration including computational aspects

3.4 Domains

Domain may be represented with the following UML 2 diagrams (Packages). One missing element is a Policy that Controlling Object is enforcing. Since Policy concept, at least the base class for it, should be aligned with other viewpoint (i.e. Enterprise Viewpoint), and that is not concluded yet, we did not include it in this diagram.

In Figure 9, CommunicationDomain_A is defined as a package containing a CommunicationControllingObject, ProtocolObject_1 to 3, and an Interceptor_A. Those objects share the same communication protocol, and are able to communication with each other. Note that not all ProtocolObjects included in this package may be instantiated at certain location in time.

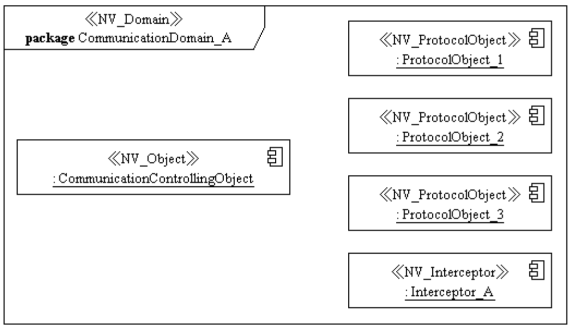

Figure 9 Example Communication Domain

Also from different perspective, a CommunicationDomain can be described as the package importing necessary member engineering objects.
The following diagram shows CommunicationDomain package containing only one engineering object (in this case it is a controlling object) accessing other member engineering objects from various Node structure packages.

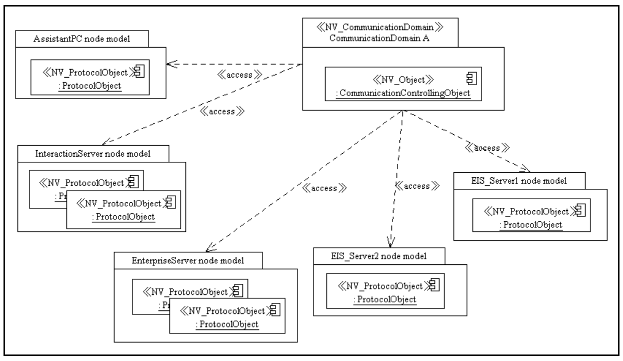

Figure 10 Example Communication Domain (2)

3.5 Correspondence

From Engineering Viewpoint to Computational Viewpoint

Our assumption is that each BEO has one to one relationship to corresponding Computational Object (from Engineering to Computational, not vice versa). This could be expressed as dependency from BEO sub-Package in Engineering Viewpoint Package to Computational Objects in Computational Viewpoint Package in UML. Note that the distribution support mechanism part of Engineering Viewpoint Language model elements may have one to one relationship with specified transparencies in transparency schema.

From Engineering Viewpoint to Technology Viewpoint

This correspondence is closely related to OMG's MDA initiative where Platform Independent Model is transformed to Platform Specific Model. In this case, Platform means real software and hardware platforms. It is our expectation that when

MDA standards get mature, they will become an enabling technology for specifying Viewpoint Correspondence.

Each Engineering Objects may be implemented with multiple Technology Objects, and multiple Engineering Objects may be implemented with one Technology Object. This could be expressed also as dependency between two Viewpoint Packages.

4. Technology Viewpoint – Issues and discussion for UML modeling

4.1 Hardware, Software and Network

In UML Deployment Diagram, main modeling elements are Node and Artifact. Typical representation of hardware elements such as CPU and memory is by the use of tagged values to Node, and we follow this approach. For operating systems and middleware, UML 2 introduced new metaclass ExecutionEnvironment to Node. We can add, if necessary, ODP semantics to this model element. Software implementing BEOs can be modeled as Artifact. CORBA Middleware and POSIX-compliant Operating System, for instance, may be modeled as or based on ExecutionEnvironment. Network like the Internet and LAN can be modeled as a Node, although hardware/software aspects of network can be treated differently.

The discussion may be regarding the extent for defining UML Profile for Technology Viewpoint, since UML 2 provides a similar set of modeling elements. Our approach is to define minimal extensions and to see what is missing based on users' experience.

4.2 Representation of Technology Concepts

Technology Object

Technology Objects, covering software, hardware and network, may be represented with UML Artifact or Node (including Node within Node).

Implementable standard may be considered as a part of specification Technology Objects implements. It could be modeled as UML Class or Component.

Implementation is defined as "a process of instantiation whose validity can be subject to test." A process may be represented with UML Activity Diagram. Since "Process" concept has been refined in Enterprise Language, we may need to refer to related modeling elements in Engineering Viewpoint.

IXIT (Implementation eXtra Information for Testing) may be represented with UML Comments as annotations to Technology Viewpoint specifications. However, there may be a case where an IXIT contains a lot of information and could not be described within a UML Comment, we may need to introduce a mechanism to refer external documents.

5. Technology Viewpoint – Possible UML 2 Models

5.1 Profile definition

The following is a diagram showing a partial UML 2 Profile definition. In order to represent hardware, software, and network as Technology Object, two base classes are used for Technology Object. The stereotypes have prefix TV_, which is a rule for defining stereotype names for Engineering Viewpoint in Use of UML for ODP system specifications standard.

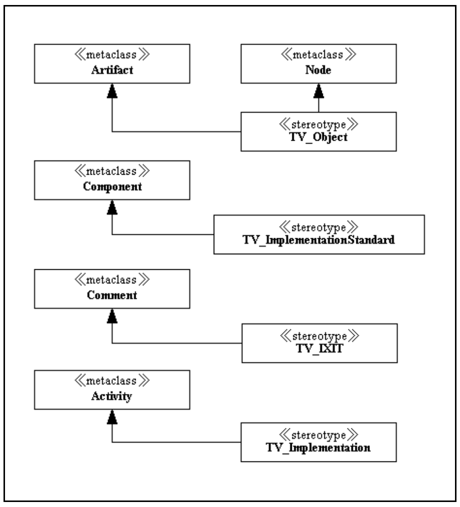

Figure 11 UML 2 Profile for Technology Viewpoint Language

5.2 Node Configuration

In technology viewpoint, we can show network components and hardware components as well as software components. This is the difference with Node Configuration of engineering viewpoint.

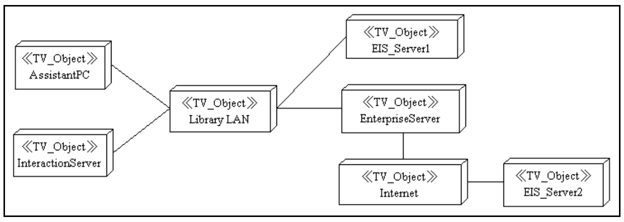

Figure 12 Example Node Configuration

5.3 Node Structure Diagram

The following is an example diagram showing physical structure of a node with hardware, software, and network elements. In this diagram, there is a Node which is connected with the Internet, and which hosts POSIX compliant operating system and J2EE compliant middleware, and two applications are introduced as Artifacts to run under those execution environments.

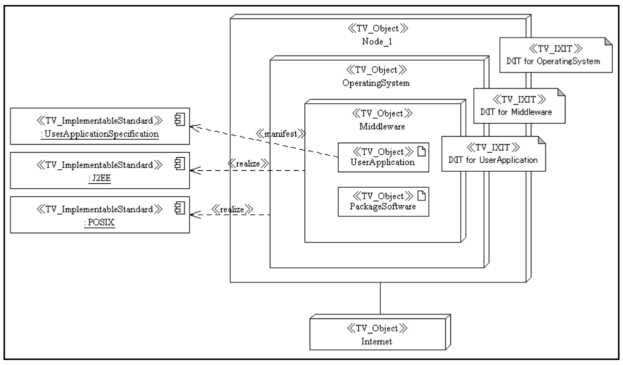
《TV_Object》 Node_1
《TV_IXIT》 IXIT for OperatingSystem
《TV_Object》 OperatingSystem
《TV_IXIT》 IXIT for Middleware
《TV_ImplementableStandard》 :UserApplicationSpecification
《manifest》
《TV_Object》 Middleware
《TV_Object》 UserApplication
《TV_IXIT》 IXIT for UserApplication
《TV_ImplementableStandard》 :J2EE
《realize》
《TV_Object》 PackageSoftware
《TV_ImplementableStandard》 :POSIX
《realize》
《TV_Object》 Internet

Figure 13 Example Node Structure

5.4 IXIT

IXIT stands for Implementation eXtra Information for Testing, and is one of technology viewpoint concepts. Since this is associated with technology objects, and since we do not have formal way of modeling “extra information,” UML Comment is used.

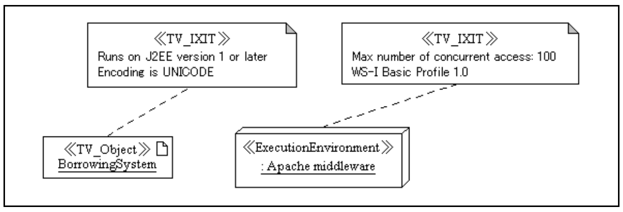

Figure 14 Example IXIT

5.5 Correspondence

<u>From Technology Viewpoint to Engineering Viewpoint</u>

Each Technology Object has one to one relationship to corresponding Engineering Object. This could be expressed as dependency from Technology Objects in Technology Viewpoint Package to Engineering Objects in Engineering Viewpoint Package.

6. Conclusions

There may be multiple ways to represent ODP viewpoint specifications with UML 2.0. Regarding approaches, we believe we have choices e.g. on class-based modeling vs. component-based modeling and on class/component based modeling vs. instance (object/component instance) based modeling. In this paper we took component-based modeling approach. And, if you compare the diagrams presented in this paper with the diagrams in RM-ODP Part 3 (i.e. Figure R2 to R8 in 2.1), we believe we have successfully demonstrated most of the Engineering Viewpoint Specifications with UML 2.0 and its Profiles. We also have demonstrated a possible UML Profile for Technology Viewpoint, which also works. However, we need to develop consistent UML Profiles for all the ODP viewpoints to allow describe different viewpoint’s concern in certain viewpoint’s specification. Another important point is that the UML mapping should be practical and the results should be accessible and usable. We hope that UML Profile for ODP to be standardized soon and the profile data be developed, published, and become openly available to interested parties, that is, UML modelers and ODP modelers.

Acknowledgements

The work described here was mostly done within INTAP Open Distributed Processing Committee, and includes some ideas and discussions from International and Japanese Committee for SC7/WG19. The authors would also like to thank Mr. David Frankel for his contribution in [5] to this UML and ODP work.

References

[1] RM-ODP, ITU-T Recommendation X.901 to X.904 and ISO/IEC 10746-1 to 4
http://isotc.iso.ch/livelink/livelink/fetch/2000/2489/Ittf_Home/ITTF.htm??Redirect=1

[2] UML 2.0 Superstructure, OMG,
http://www.omg.org/cgi-bin/doc?ptc/2004-10-02

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

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

[5] Applying EDOC and MDA to the RM-ODP Engineering and Technology Viewpoints, INTAP Technical Report, David Frankel Consulting, http://www.net.intap.or.jp/e/odp/odp-techguide.pdf

[6] Federal Enterprise Architecture Reference Model, http://www.whitehouse.gov/omb/egov/a-2-EAModelsNEW2.html