Analysis and design of a management application using RM-ODP and OMT

Erik Colban[a] and Fabrice Dupuy[b]

[a]Norwegian Telecom c/o Bellcore, NVC-1C115, 331 Newman Springs Road, Red Bank, NJ 07701, USA. Phone: + 1 908 758 2875, e-mail: erik@tinac.com

[b]France Telecom / CNET, LAA/EIA/BSA, Technopole Anticipa, 2 avenue Pierre Marzin, 22307 Lannion Cedex, France. Phone: + 33 96 05 36 65, e-mail: dupuy@lannion.cnet.fr

Abstract

This paper studies the assimilation of OMT, an object-oriented design method not particularly well suited for distributed applications development, into RM-ODP, a framework for distributed applications lacking a method. It reports on current work in this venture in the TINA Consortium.

Keyword Codes: D.2.2; D.2.7; D.2.10
Keywords: Software Engineering, Tools and Techniques; Distribution and Maintenance; Design

1. Introduction

Adhering to the RM-ODP viewpoints along with object-orientation is not as easy as it may seem from the reading of the standard and it can lead to different interpretations as to what the objects described mainly from the information viewpoint, the computational viewpoint and the engineering viewpoint really represent.

Since the standard to be RM-ODP [1] [2] [3] does not prescribe any method along with the architectural concepts, a system designer can for example choose to follow an object-oriented analy-sis and design method, without any additional consideration of the RM-ODP viewpoints, and be convinced (somehow rightly) that his/her analysis and design models correspond respectively to the RM-ODP information and computational models. Another designer can adhere to the functional model and describe the functional modules by means of the computational viewpoint and the information exchanged between them by means of the information viewpoint. RM-ODP would perfectly suit his/her needs.

The authors of this paper, and the TINA-C core team [4] working on the software architecture of the future telecommunications information networks, adopted another interpretation or method which consists of taking over an object-oriented analysis and design method, namely OMT [5] and adding a substantial set of RM-ODP concepts to the analysis and design phases in order to take more thoughtfully account of the distribution issues.

This paper describes first the application example that provides the ground for comparison. Then the main sections of this paper aim at showing how the example is examined using the OMT method on one hand and using a mixure of RM-ODP and OMT, as proposed by the TINA-C core team, on the other hand. The conclusion highlights the benefits of the latter method.

2. Example

All along this contribution, the same example, network resource management, will be taken. The problem statement that serves as an input to the analysis and design phases of the methods is proposed to be the following: A computer-based distributed system is required to manage a telecommunication network, regardless of the size and the type of the network (ATM, SDH, POTS,...), in order to react to its faults as seamless and transparently to its users as possible, and to control the correct establishment of end-to-end connections at its various end-points. A topological map of the telecommunications resources that make the network is stored in the system in order for the network manager to be kept updated about the network configuration, to easily locate the established connections or the faults, and to identify the actions to undertake.

It should be noted that the purpose of this contribution is of course not to come up with a complete problem statement, nor to completely show how the sketchy statement given above will be analyzed and turned into a complete object-oriented design. Only parts of this problem statement will be used to exemplify the discourse and therefore will undergo the process of analysis and design outlined below. The focus is mainly on the OMT method, its benefits and its shortcomings related to distribution concerns.

3. The OMT approach to the analysis and design

OMT is only used in this contribution to exemplify the discourse and because the core team gained experience in using it during two years (In no way, this choice should be considered an assessment of the various OOA&D methods).

The OMT object-oriented analysis and design method consists of three non compulsory sequential steps: analysis, system design and object design.

- It is proposed in OMT to analyze a system according to three related but different "viewpoints": an *object model*, a *dynamic model* and a *functional model*. The *object model* represents the static, structural "data" aspects of a system. The *dynamic model* represents the temporal "control" aspect of it. The *functional model* represents the transformational aspects.
- The OMT system design consists in grouping the objects identified in the analysis phase into subsystems, each subsystem being associated later with specific computing resources (DBMS, graphical user interfaces,...)
- The OMT object design consists in refining the objects identified during the analysis phase.

3.1 OMT analysis

During analysis, the OMT object model permits to describe the patterns of the objects, attributes, operations and links. In the case of our example, the object model enables to give a view of the telecommunications network structure composed of layer networks, subnetworks, termination points, to state that the responsibility of managing the network can be given to different kinds of network managers, to outline the difference between link connections established between two subnetworks and subnetwork connections established within a subnetwork. The Figure 3-1. below contains neither attributes nor operations for the sake of clarity.

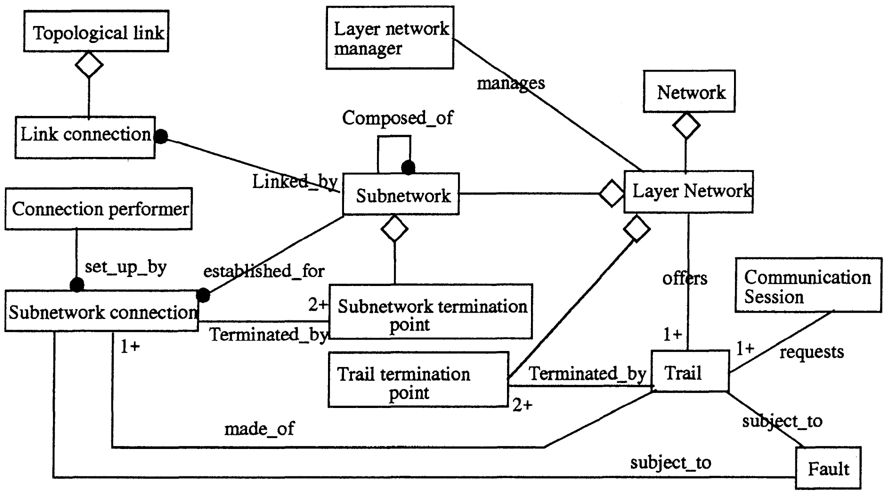


Figure 3-1. An OMT Object Model of the network resource management system

A complete OMT object model would include, as stated earlier, a description of the attributes and operations of each object. For example, the subnetwork connection is composed of attributes named *direction* (a connection can be simplex or duplex), *cast* (a connection can be unicast or multi-cast), *Quality of Service* (throughput, delay, jitter), and it offers operations to delete the connection, to reserve a connection, to start transferring the media flow through the connection, to associate a connection with subnetwork termination points (SNWTPs).

The OMT Dynamic model permits to describe the states of the objects and the events sent by objects to stimulate the others. In our case, the dynamic model enables to examine the states of the objects identified in the object model and to relate any state change to events (mainly corresponding to operation invocations). The Figure 3-3. shows a state diagram of the subnetwork connection object (inspired from [6]).

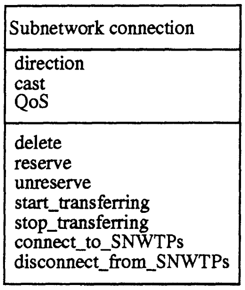


Figure 3-2. The subnetwork connection object

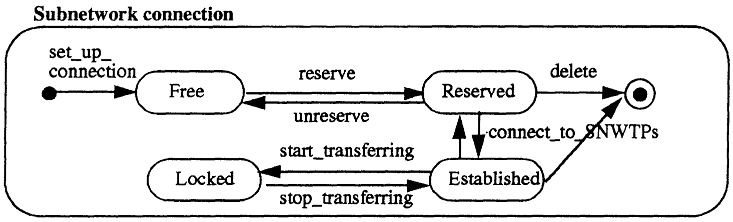


Figure 3-3. A bit of an OMT Dynamic Model of the network resource management application

The OMT Functional model specifies ***what*** happens, whereas the Dynamic model describes ***when*** it happens. The Functional model shows how output values in a computation are derived from input values. Consequently, this model describes data flows where the Dynamic model describes control flows. The Figure 3-4. shows an example with some functions associated to the object 'subnetwork connection'.

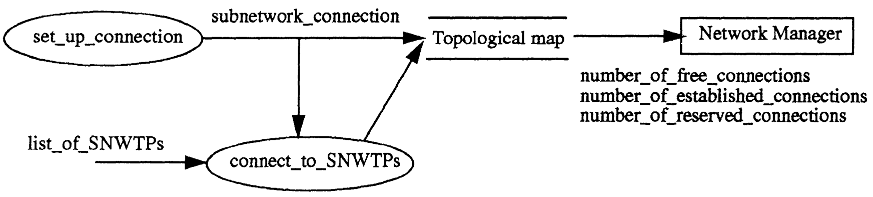


Figure 3-4. A bit of an OMT Functional Model of the network resource management application

3.2 OMT system design

During analysis, the focus is on what needs to be done. During design, the concern is on how the problem will be solved. The OMT **system design** consists in determining the overall structure of the system, a sort of high level design, in which the system is partitioned into subsystems and objects are packaged into these subsystems. The rationale behind partitioning into subsystems is to have the objects share some common properties: similar functionality, same physical location, or execution using the same computing resources.

For the network resource management example, it can be proposed to package the topological links and the link connections into a subsystem called transmission subsystem, the subnetwork, its subnetwork connections and its termination points into a switching subsystem, the trails, trail termination points and layer networks into an operations system, and the communication sessions into a telecommunication service package.

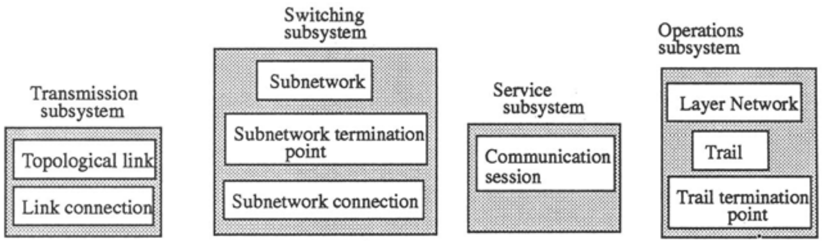


Figure 3-5. A system design of the network resource management system

One main remark is to be made concerning the system design step of the OMT method (this step consisting of packaging objects identified during the analysis phase into subsystems). Usually, when it comes to partitioning the overall system into packages, the packaged objects are executable units (or executable objects), which, by essence, constitute the basic units for structuring the computer-based system. Here, with the OMT method, the assembled objects are the objects directly derived from the analysis phase. As no intermediate step, in which the analysis objects would be turned beforehand into 'executable' objects, is explicitly proposed in OMT, it means these problem domain objects have also to be considered as executable units in OMT. Such a homomorphism between analysis and design objects cannot always hold: for example, the network manager 'human being' cannot be 'packaged' into a computer-based subsystem.

3.3 OMT object design

The OMT **object design** phase determines the full definition of classes, the implementation of the associations (with pointers to the associated objects for example), and the algorithms of the methods used to implement the operations. The objects discovered during analysis serve as a skeleton of the design, the object designer having to choose the ways to implement them, to add new objects for storing intermediate results or translating data representations,... These objects present in the analysis phase are usually carried directly to the design phase. Object design in OMT is then nothing more than a process of adding details and deciding how to implement.

For example, the object model discovered during analysis for the network resource management example is detailed during object design so that all operations and associations become implementable. The subnetwork connection object, for instance, is further designed (Figure 3-

6.) to offer access to its attributes (e.g., get_direction, get_cast, get_QoS) and to implement the relationships in which it takes place (e.g., give_termination_points). The type of the attributes is determined (e.g. *direction* of type enumeration).

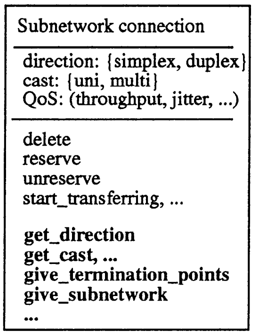
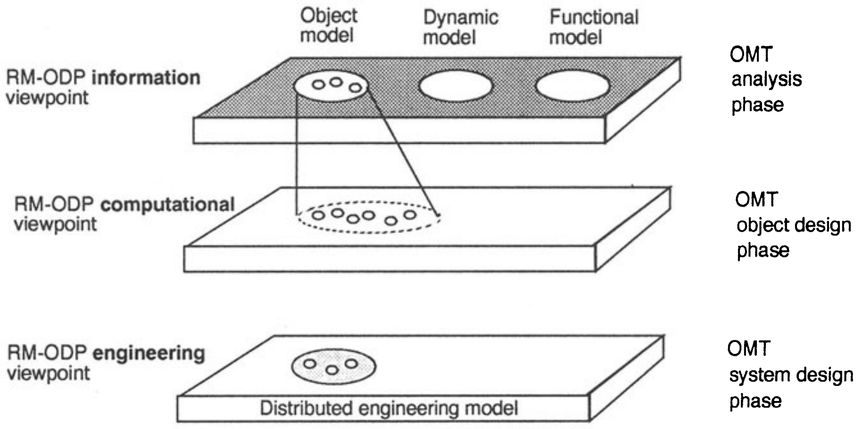


Figure 3-6. The subnetwork connection object further refined during object design

3.4 The shortcomings of the OMT approach

In this approach, the OMT analysis stage seems to correspond to the RM-ODP information viewpoint, the OMT object design stage to the RM-ODP computational viewpoint: (the designer is more concerned, at this stage, by how the system works and what to do to optimize it), and the OMT system design phase to the RM-ODP engineering viewpoint (the system is broken down into subsystems or system components that share common properties).


Object model
Dynamic model
Functional model
RM-ODP information viewpoint
OMT analysis phase
RM-ODP computational viewpoint
OMT object design phase
RM-ODP engineering viewpoint
Distributed engineering model
OMT system design phase

The shortcomings of this method can be essentially summarized as a lack of a framework for distribution:

- There is a suggested one-to-one mapping in OMT between the objects discovered during analysis and the objects eventually designed. It is deemed by the authors that this is not realistic, and too constraining. As RM-ODP does not prescribe such a mapping between information objects and computational objects, it should provide a more flexible framework.
- During the OMT system design, no distinction is suggested to be made between objects with interfaces for local use and objects with interfaces for possibly remote use. Therefore, the distributed processing infrastructure supporting the OMT object-oriented computational model (the OMT object/system design) has to handle all references to objects in the same manner, i.e. as if they were all distributable or none was at all. With the concept of "clusters", within which the communication mechanisms are left to the cluster manager, and between which communication relies on the processing environment kernel facilities (a sort of object request broker), the reference model of ODP provides a noticeable distinction that helps to reduce the distributed environment workload.
- During the OMT object design, all the burden implied by the distribution issues is left to the designer, who supposedly has to know how to deal with the different transaction semantics (which transaction model to take), with security (how to trade off between security and performance), with remote access (how to encode/decode the invocation parameters), and with quality of service procurement. It is clear that, on the subject, the RM-ODP computational model constitutes a substantial improvement of OMT.
- Also, this approach does not take parallelism or concurrent access into account. Besides an information model explaining by means of objects what the system does, further design decisions need to be taken in order to avoid too many accesses to the same object interface, or to allow concurrent processing.

4. The TINA-C approach of using OMT and RM-ODP

As the previous chapter suggests, OMT lacks concepts and principles, such as the ones included in the reference model of ODP, that can help a distributed system designer in his/her task, whereas it constitutes an object-oriented analysis and design method somehow efficient and promising. TINA-C aimed at using both RM-ODP and OMT to provide a complete and tool-supported framework. There are of course several ways in bringing these two together, not all as good. During 1993, TINA-C attempted to clarify these matters for three of the RM-ODP viewpoints: the information, computational, and engineering viewpoints.

4.1 Information Specifications with OMT

In 1994, TINA-C started to use OMT for the information specifications. Certain care must be taken in order to make an information specification design independent. Even if one restricts the exercise to what is traditionally seen as object-oriented analysis, computational aspects are often introduced mistakenly into the specification. Consider for instance, the ***Subnetwork Connection*** object type in Figure 4-5. A *connect_to_SNWTPs* operation is specified which associates *Subnetwork Termination Points* to a *Subnetwork Connection.* The operation does not pertain only to a ***Subnetwork Connection***, but also to the ***Subnetwork Termination Points***. One may therefore question why this operation has been assigned to the ***Subnetwork Connection*** object type and not to the***Subnetwork Termination Point***, or to the ***Subnetwork***. As a matter of fact, assigning the operation to ***Subnetwork Connection*** is a design choice that belongs to the computational viewpoint. The *connect_to_SNWTPs* operation provides the means to add a *Subnetwork Termination Point* to the *Subnetwork Connect*. In the information viewpoint, the analyst should only be interested in stating that *Subnetwork Termination Points* can be associated with*Subnetwork Connections*. In Figure 4-1, this is accounted for by the multiplicity 2+ on the *Terminated_by* association.

For the information specifications, TINA-C suggests to only use the Object Model part of the OMT analysis phase. The reason for this is that it is difficult to reason about any event flows when no computational model is in mind. However, in specifying the information viewpoint, TINA-C goes far beyond simply providing a set of object diagrams.

Objects, in the information viewpoint as well, have a state. The state is defined by the values of the attributes of the object. Therefore, every object type is assigned a set of attributes. The state may change, but only according to a set of specified actions. Note that these actions only pertain to the object to which they belong, unlike the *connect_to_SNWTPs* operation mentioned earlier. If an operation can take place only under certain conditions, these are specified as preconditions to the operation. Any property that has to hold after the state change is specified as a post-condition. If there are any conditions that imply a change of state, they are specified as triggering conditions. Note that by specifying pre-, post, and triggering conditions, some of the expressive power of dynamic modelling is covered. Properties that hold regardless of the state of the objects are specified as invariants. In specifying all these conditions the analyst may refer to associated objects; the next paragraph gives more insight.

In order to account for dependencies between objects, we use the associations of OMT (referred to as relationships in TINA-C). These associations may have a state, in which case it is specified in the same manner as for object types. Also important, when specifying an association, are the constraints on the instances of the association or the associated objects. Multiplicities are one kind of such constraints, but there may be others. In specifying these constraints, we may refer to the class (in the sense of ODP) of the association and the associated objects.

In TINA-C an OMT tool has been customized in order to facilitate writing information specifications. With the customizing the tool allows the user to enter in a textual form any conditions that belong to the specifications. It also generates a report in a notation that has been chosen by the Core Team (the notation used is an adaptation of GDMO).

Figure 4-1 shows a small fragment of the TINA-C Network Resource Information Model. Note that the NRIM is under development and that it may change by the time the reader discovers it. Note also that currently the NRIM only includes information objects representing managed resources: it lacks information objects playing manager roles (as tentatively proposed in Figure 3-1.).

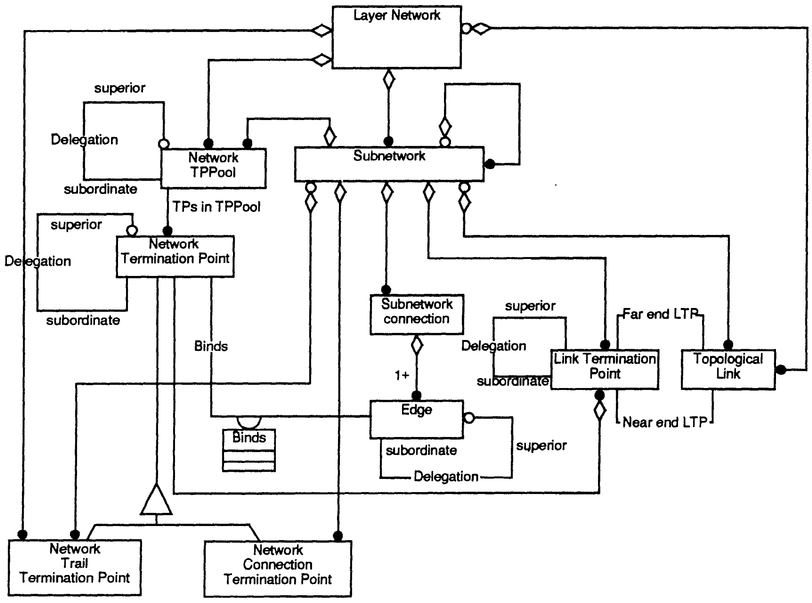


Figure 4-1. A Fragment of the TINA-C Network Resource Information Model

4.2 Computational Specifications with OMT.

Actually, OMT already includes a computational model: as mentioned in Section 3, all object-oriented methods assume a computational model; furthermore, as stated in Section 4, the OMT object design phase partly corresponds to the RM-ODP computational viewpoint. However, enhancements of this computational model are necessary relatively to the following points.

4.2.1 Correspondence between analysis and design objects

In object orientation, "real world" concepts are modelled as object types and phenomena as objects. The purpose is to get a "homomorphism" between the real world and the model or, in other words, to reflect the structure of the real world in the model. This homomorphism should be carried through to implementation. The value of doing this is to produce systems that are easier to maintain and to extend.

These principles of object orientation guide the information modelling and should, as far as possible, also guide the computational modelling. Therefore, a good starting point for the computational specification is to naturally map each information object type onto one computational object type (this is precisely the correspondence embedded in OMT) and, whenever the one-to-one mapping between information objects and computational objects can not be maintained, the designer has to remember that the homomorphism, and therefore one advantage of object orientation, are lost. In this homomorphous scenario, relationship types (corresponding to associations in OMT) may either be mapped onto separate computational object types or one may let the related objects maintain on their own the given relationship.

However, for several reasons given below, the one-to-one mapping is difficult to maintain and may lead to a non realistic distribution-driven computational object design:

- An information object may have to be split and distributed on several nodes thus providing interaction points at several locations. For example, the information object 'subnetwork' is split up into as many computational objects (also called hereafter subnetworks) as there are levels in the network structure recursiveness.
- Information objects that appear in great number can be stored in a data object container and retrieved through a single interaction point. For example, an image of all created subnetwork connections may be stored in that manner. The database manager, called hereafter topological map repository, is seen as one computational object whose interfaces are determined by the possible requests to the database.
- In order to perform certain operations that involve collection of objects, computational objects that have interfaces that offer such operations may be specified. These objects do not necessarily correspond to any information object.
- In an open application it is important to be able to set the limits on how open the application should be. One may need to prevent users from getting direct access to a piece of information to prevent users to tamper on it. The solution is to offer to the users computational interfaces that clearly defines and delimits the possible interactions that may occur.
- Trading is a costly operation that should be minimized. For this reason one might group operations, that can be related, into one interface instead of spreading them on several computational interfaces.
- A transaction is defined as an operation that spans more than one computational object. A transaction is also a costly operation and one should try to minimize the number of them in the same way as trading is minimized. This can be achieved by mapping several information objects into one computational object. For example, grouping all termination points related to a subnetwork within the same computational object 'subnetwork' allow to have the transactional operation *Connect_to_SNWTPs* not span any other computational object.

Due to these differences, separate object models need to be developed for each of the information and the computational viewpoints. Not only are the object diagrams different, but object types will also be specified differently in the computational object model than in the information object model. Object types in the computational object model will for instance be specified with other operations and attributes than the corresponding object types in the information object model. For instance, attributes that serve a object reference holders in the computational object model could correspond to associations in the information object model. See also previous comments about the operation *connect_to_SNWTPs* which is a perfectly valid operation in the computational object model, as shown in the figure below (inspired from [6] and [7]).

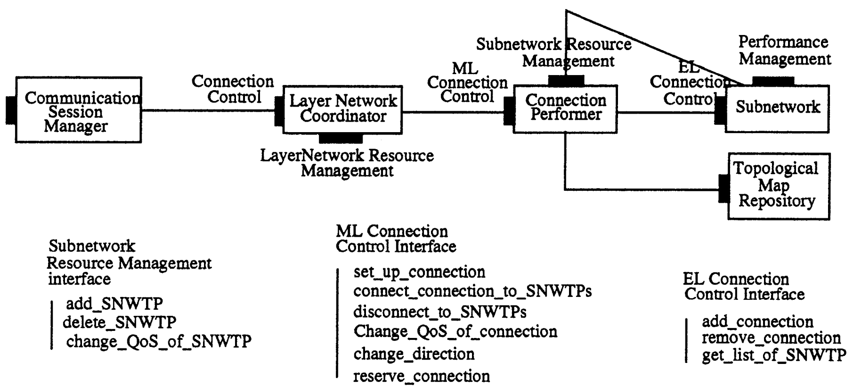


Figure 5-3. A possible Network Resource Management Computational Model

4.2.2 Extensions of the OMT computational model

In addition, the computational concepts in OMT are not fully complete to allow a good design of a distributed application. OMT lacks concepts to express, for instance:

- objects with multiple interfaces and dynamic instantiation of interfaces,
- stream interfaces through which continuous media flows are permitted (for a lower-level subnetwork called a matrix and conveying user data for instance),
- quality of service like transactional QoS, real-time QoS, availability,
- concurrency control.

In TINA-C, an Object Description Language (ODL) has been defined to enable the designer to specify these aspects of a computational object (ODL is an extension og OMG IDL). We believe that OMT should be enhanced with these concepts although the TINA-C core team has not yet totally carried out any experiments that validate the appropriateness of the language.

4.3 Engineering Specifications with OMT

TINA-C is currently working on the definition of a so-called Life-Cycle Service (as close to the OMG object life cycle as it can be) and on the deployment of computational objects into RM-ODP clusters. LCS and cluster management are used to master the computational object life cycle (creation, activation, deactivation, deletion) respectively at a fine grain (application level) and at a coarse grain (system level). Cluster management constitutes a major part of an engineering specification.

The TINA-C core team believes that the OMT system design is insufficient with respect to engineering modelling. In a one or two year time frame, it should be able to propose a language, aligned with the RM-ODP engineering concepts, that enhances OMT on this matter.

5. Conclusion

OMT is not a method that is targeted towards distributed software development. In order to use OMT in such a context, it is necessary to enhance it with concepts that belong to the area of distributed computing. TINA-C is currently pursuing this route by working on a software architecture intended to include the most interesting concepts and principles of both RM-ODP and OMT.

Further more, the core team is currently customizing an OMT tool so that the user can express various aspects or viewpoints of distributed computing as defined in RM-ODP. So far, the work has been done for the information viewpoint, but similar customizing is planed for the computational and engineering viewpoints. This customized OMT tool may constitute in the future a skeleton of a distributed software engineering tool that would enable to user-friendly and more easily follow the RM-ODP recommendations.

Acknowledgments

The authors would like to thank the core team of the TINA Consortium who contributed to the specifications of the TINA Architecture, Magnus Lengdell (Telia, Sweden)who tested the usefulness of the OMT tool customization and Valère Robin (France Telecom / CNET) who provided valuable comments on this contribution.

References

[1] ISO/IEC 10746-2.2 / ITU Recommendation X.901, *Basic Reference Model of Open Distributed Processing - Part 1: Overview and Guide to Use,* International Organization for Standardization and International Electrotechnical Committee, June 1993.

[2] ISO/IEC 10746-2.2 / ITU Recommendation X.902, *Basic Reference Model of Open Distributed Processing - Part 2,* International Organization for Standardization and International Electrotechnical Committee, June 1993.

[3] ISO/IEC 10746-3 / ITU Recommendation X.903, *Basic Reference Model of Open Distributed Processing - Part 3 - Prescriptive Model,* International Organization for Standardization and International Electrotechnical Committee, November 1992.

[4] Fabrice Dupuy, Gunnar Nilsson, *The TINA Consortium: Towards Telecommunications Information Networking Services,* to be published in the ISS'95 proceedings, Berlin, April 1995.

[5] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, *Object-Oriented Modeling and Design,* Prentice Hall, Englewood Cliffs, N.J.:, 1991.

[6] ITU SG15 Q.30/15 D.272, *Information Specification of the subnetwork connection services,* May 1994.

[7] Bloem, J., Pavon, J., Oshigiri, H., Schenk, M., «*TINA-C Connection Management Architecture,* TINA'95 Proceedings, Melbourne, February 1995.