UML Profiles for the Specification and Instrumentation of QoS Management Information in Distributed Object-based Applications

Juan I. Asensio[†], Víctor A. Villagrá, Jorge E. López·de·Vergara, Julio J. Berrocal.

Department of Telematic Systems Engineering
Technical University of Madrid (DIT-UPM). Spain.
{jasensio,villagra,jlopez,berrocal}@dit.upm.es
[†] Visiting researcher from U. of Valladolid. Spain

ABSTRACT

This paper focuses on the QoS management of IT systems based on object-oriented distributed applications. It presents a way of specifying application-level QoS information during the development of object-based distributed applications: a UML profile (an extension to the UML) based on QoS concepts and principles defined by ISO/ITU-T in their works on QoS in ODP, that can be applied to the specification of QoS management information independently of those particular details related to the distributed object-based distributed computing platform architecture used in each case. It also presents a solution (another UML profile) for modeling the instrumentation of QoS monitoring mechanisms within the monitored applications. The contributions presented in this paper are intended to facilitate the development of distributed object-based applications whose QoS can be managed in an integrated way.

Keywords: QoS management, QoS specification, QoS monitoring, UML profile, distributed object-based applications.

1. INTRODUCTION

The Quality of Service (QoS) management of IT systems is one of the key aspect for their success due to the great competitiveness of the environment in which they are deployed: no matter what technology they are based on, only those systems that increment the level of satisfaction of their users, when using them, will gain acceptance. Therefore, although lots of definitions of the term Quality of Service can be found, in the context of this paper Quality of Service will be understood as the set of non-functional aspects of a system that determine the satisfaction level of the users of the functionality they provide.

Quality of Service Management refers to the set of activities devoted to the surveillance and control of the resources involved in the provision of an adequate QoS level. QoS Management is a global task in the sense that it takes into account all kind of resources that support the service(s) provided by an IT system: network resources, system resources, middleware resources, applications, etc. All of them must be co-ordinated in order to fulfil the overall goal of obtaining QoS levels that match the users' needs. That way of understanding QoS is called "end-to-end QoS". End-to-end QoS implies a top-down translation of user-level QoS requirements into lower level QoS requirements.

This paper focuses its attention into the QoS Management of one particular type of resources: applications. Furthermore, it deals with one particular kind of applications: those supported by distributed object-based processing platforms (CORBA, RMI, DCOM,...). These applications are becoming more and more important in the IT world mainly because they can be easily deployed over heterogeneous environments, because they present important advantages for their developers and because they can integrate legacy application within new, upgraded systems.

This paper proposes new techniques for facilitating the introduction of QoS aspects in this type of applications during their development. More precisely, it defines ways of creating design models of an application enhanced with all the QoS aspects of it, as well as ways of modeling how to introduce support (within the application itself, if needed) for managing that QoS. The proposals of this paper must be understood as an enhancement of traditional development process of distributed object-based applications in order to take into account QoS issues.

The proposals of this paper are based on the ITU-T QoS framework [14] (that provides basic QoS concepts and principles for distributed object-based applications), and the Unified Modeling Language, UML [8] (that is extended in order to support the mentioned QoS concepts and principles during different modeling activities of different development processes).

This paper proposes two complementary extensions to UML:

- The first one (called UML-Q) would allow, for instance, specifying (at design time) how long it should take to obtain the results of the invocation of a method of a particular object of an application.
- The second one (called UML-M) would allow specifying how it should be checked whether that requirement is satisfied or not during run time: introducing new code into the application, by monitoring external events, etc.

Adequate tool support of these extensions might facilitate the introduction of QoS management capabilities within the application, independently of the their functional scope (i.e. e-commerce, tele-education, etc.).

The paper is structured as follows: section 2 presents the ISO/ITU-T work on QoS which is the conceptual base for the content of the paper; section 3 gives an overview of the way UML has been extended for developing the results presented in this paper; section 4 presents the developed language (UML-Q) for QoS management information specification and illustrates its usage in a simple example; section 5 presents the proposed UML extensions for QoS monitoring modeling (UML-M); section 7 summarizes the paper and presents some future research topics.

2. THE ISO/ITU-T QOS FRAMEWORK AND ITS APPLICATION TO ODP

QoS Management of IT systems is a very broad and complex challenge, event when restricting its scope to the QoS Management of distributed object-based applications. Different architectures for structuring all the aspects QoS Management involves have been proposed. But when comparing these architectures [3][11], a fundamental question arises: what are the characteristics that an "ideal" QoS Management Architecture should have? This question should be answered by means of QoS frameworks that precisely define the QoS concepts involved in the QoS Management domain and that provide architectural patterns and other tools that must be used as the basis for the development of QoS Management architectures.

One of the most important and complete QoS frameworks is the one that is currently being jointly developed by ISO and ITU-T. The ITU-T X.641 Recommendation (ISO/IEC IS 13236) [14] provides a framework that defines the QoS basic principles, concepts and terminology that can be used in order to develop different contributions related to QoS. The goal of ISO/ITU-T is providing a framework whose contents are generic enough so as to be applied to basically all the aspects of IT. The main concept of the this QoS framework is that of "QoS Characteristic" which is defined as *"a quantifiable aspect of QoS, which is defined independently of the means by which it is represented or controlled"*.

This generic framework has been refined in order to be used in two particular areas: communications based on architectures compliant with the OSI reference model and distributed object-based applications compliant with the RM-ODP standards [13][12]. This last particularization has been adopted as the conceptual base for the scope of this paper. It defines Specific ODP QoS concepts such as "QoS Relation" (mutual obligations of an object and its environment) which is the basic component of "QoS requirements", "QoS capabilities", "QoS offers" and "QoS contracts". It also gives some ideas of how QoS aspects can be introduced into the five viewpoints prescribed by RM-ODP, it points out other important QoS management functions and, finally, it identifies the main requirements on notations for expressing QoS (representation of all the potential types of QoS characteristics, support for QoS aspects that depend on measurements coming from different places, etc.).

All these aspects have motivated the main two subjects covered by this paper:

- Definition of a language capable of specifying quality of service information that is compliant with the concepts defined in the work of ISO/ITU-T on QoS in ODP: UML-Q described in section 4.
- Definition of a language capable of modeling QoS monitoring mechanisms (one of the most important mechanisms that support the QoS management functions identified in the described frameworks) within distributed object-based applications: UML-M described in section 5.

Both languages are examples of the so-called "UML profiles". The following section describes this concept.

3. UML PROFILES

Although UML is a general purpose modeling language, it contains extensibility mechanisms that can be used to tailor it to specific domains (QoS information specification, for instance). These extensibility mechanisms can be understood as indirect modification, at the model level, of the UML meta-model [8]. The standard extensibility mechanisms of UML are *stereotypes, tagged values and constraints*. These extensibility mechanisms are called "lightweight extensibility mechanisms" [1] in contrast to the direct manipulation of the UML "meta-model" that can be interpreted as "heavyweight extensibility mechanisms" (addition of new meta-classes, meta-associations, etc.).

In order to give support to the gradual adoption of "standard" UML extensions, OMG has introduced the concept of "UML profile" which, in spite of the lack of a normative definition, has already been used in several OMG technical groups. A "profile" [9] might be defined as an "specification that specializes one or several standard meta-models, called "reference meta-models". In the context of OMG, all those reference meta-models must be compliant with the meta-meta-model prescribed by MOF (Meta-Object Facility) [7] of OMG.

This paper defines two UML profiles in order to use this modeling language for the specification of QoS information related to distributed object-based applications and for the modeling of mechanisms for monitoring the specified QoS information.

Some problems have been identified when using the above concept of profile and UML (mainly related to the reduced set of data types that can be used at the meta-model level) and it is not clear how to represent the UML meta-model extensions introduced by the profile. Nevertheless, this paper has adopted the following approach so as to define the proposed UML extensions:

- New stereotypes and tagged-values are represented by means of elements of a new "virtual meta-model". Stereotypes are represented by new meta-classes that maintain a generalisation relationship with the corresponding base meta-classes. Tagged-values are represented by means of meta-attributes.
- In the new virtual meta-model some new meta-associations are added. This addition might be considered as a heavyweight extension to the UML. Nevertheless, this new meta-associations can be considered as a way of representing meta-attributes whose type is the meta-class that is placed on the opposite end of the meta-association. This consideration implies that the tagged-values (that would represent those meta-attributes at the model level) should be allowed to use values whose types might correspond to elements of the meta-model.

All these ideas are used in the following sections for describing UML-Q and UML-M.

4. UML-Q: A UML PROFILE FOR QOS MANAGEMENT INFORMATION SPECIFICATION IN DISTRIBUTED OBJECT-BASED APPLICATIONS

One of the items identified in the work developed by ISO/ITU-T regarding QoS in ODP is the need for a QoS language capable

of representing all the QoS information related to a system during all its life cycle. UML-Q is the proposal of this paper for solving this issue.

UML-Q (UML for the QoS information specification) is a UML profile for the specification of QoS information in distributed object-based applications. UML-Q is based on the concepts coming from RM-ODP [13] and from the ISO/ITU-T framework on QoS and its refinement for ODP [14][12]. As UML-Q is based on RM-ODP concepts, it is independent from particular distributed object-oriented processing platforms (CORBA, RMI, etc.). UML-Q does not imply the use of any particular application development methodology. It only needs that the software engineering methods used in the chosen methodology include models with contents similar to those prescribed by the information and computational ODP viewpoints, and that those models are expressed using UML

UML-Q is defined in terms of a new "virtual meta-model" based on the UML meta-model, as it was described in the previous section. The new "virtual meta-model" is described in terms of:

- What types of QoS information can be specified
- How QoS information can be combined
- How QoS information is related to application models

For instance, Figure 1 shows the part of the new "virtual meta-model" of UML-Q that defines the modeling constructs needed to specify different types of QoS information.

UML-Q relies heavily on a contract specification language called OCL (*Object Constraint Language*) [8]. OCL is related to UML-Q in two senses:

- It is used for expressing QoS requirements
- It is used for detailing the well-formedness rules (the static semantics) of UML-Q itself (in the same way as it is used for defining the well-formedness rules of UML [8]).

In order to illustrate the usage of UML-Q, Figure 2 shows a UML model (a simplified computational viewpoint) of what might be a currency trading system (this example is extracted from [4]).

Figure 3 shows how UML-Q can be used to define QoS information (numberOfFailures, delay, etc.) that is grouped into categories (reliability, performance, etc.). This QoS information (based on the QOSCharacteristicType modeling element) is defined independently of the application it is going to be applied to (these definition therefore might be reused).

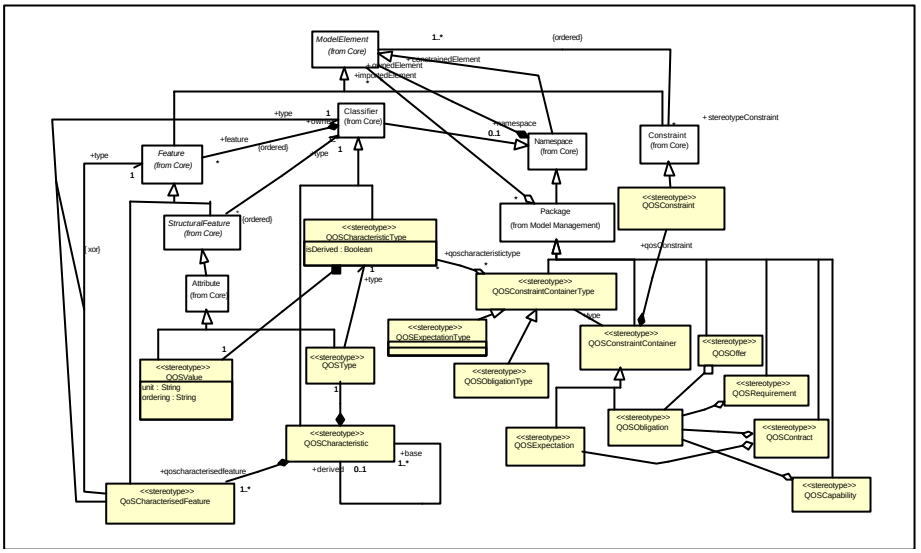

Figure 1. ***Portion of the UML-Q meta-model (shadowed elements are defined by UML-Q. Non shadowed elements belong to the UML meta-model)***

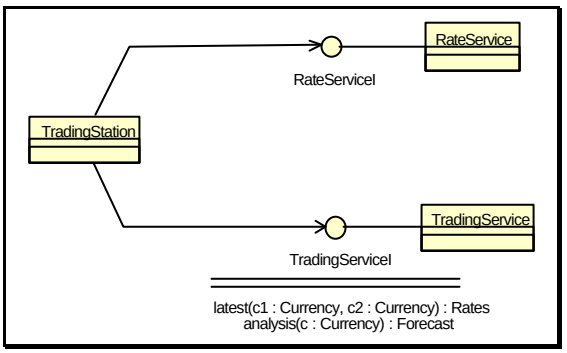

Figure 2. ***Simplified computational model (expressed in UML) of a distributed object-based application whose QoS management information has to be characterized.***

All this defined QoS Information can be "attached" to the application UML model by means of instances of several UML-Q "virtual meta-classes" in the way shown in Figure 4.

Figure 4 also shows some QoS requirements (instances of QOSConstraint, according to the "virtual meta-model" of Figure 1), which are OCL expressions.

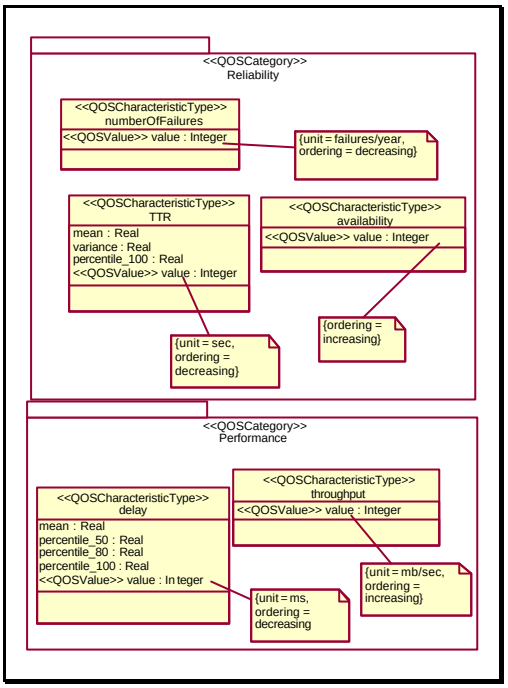

Figure 3. ***Example of UML-Q specification of QoS information.***

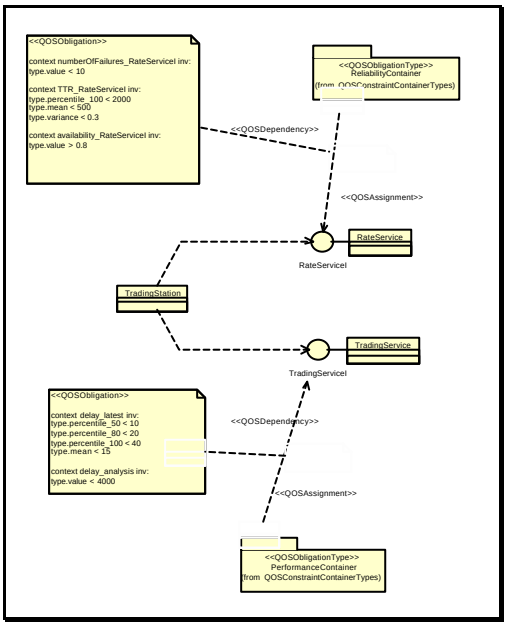

Figure 4. ***UML-Q specification of QoS information attached to the UML application model.***

5. UML-M: A UML PROFILE FOR MODELLING QOS MONITORING MECHANISMS

QoS management information specification is only one aspect of the overall problem or QoS management. Different QoS management functions are needed in order to fulfil the specified users' QoS requirements: monitoring, negotiation, adaptation, etc. This paper deals with the problem of monitoring QoS aspects of distributed object-based applications. More concretely, it presents another UML profile for modeling, in a more flexible and efficient way, the instrumentation of monitoring mechanisms within distributed object-based applications whose QoS has to be monitored.

Lots of solution for the monitoring of distributed systems in general and of distributed object-based applications in particular have been proposed. Some of them presents very powerful monitoring languages for specifying how to combine and process monitoring information obtained from the monitored system although all that combination and/or processing is done in a centralized way. Other proposals present a distributed but fixed architecture for processing monitoring information that takes into account the possibility of introducing part of the processing into the application itself thus reducing the amount of interchanged monitoring information but reducing as well the performance of the monitored application. All of them use the same set of concepts (type of monitoring information, types of monitoring functional units) but use them in different and fixed ways. Nevertheless, every distributed application has its own characteristics (performance requirements, distribution properties, and so on) that make some of the existing monitoring solutions more suitable than others. Even in some cases, a combination of monitoring solutions can be the best approach.

In order to facilitate the application of all these monitoring concepts and techniques in a flexible way, this section presents UML-M (UML for modeling QoS Monitoring mechanisms). UML-M is a UML profile that defines new and specific modeling constructs that facilitate the modeling of QoS monitoring information and the corresponding processing functions. UML-M is intended to be used during the design phase of a distributed object-based application in order to model the introduction (instrumentation) of the necessary QoS monitoring information processing functions and for defining what monitoring information they are going to process and interchange. The added value of UML-M resides in that it enables the developers of the monitoring infrastructure to focus on generic monitoring concepts and procedures without taking into account the concrete mechanisms that can implement the modeled functionality. In other words, the goal of UML-M is modeling, in a flexible and generic way, how distributed object-based applications have to be instrumented in order to monitor their QoS (QoS that has been specified by means of UML-Q). UML-M does not propose new monitoring architectures or monitoring techniques but it tries to facilitate the modeling of QoS monitoring infrastructure in distributed object-based applications, infrastructure that can be implemented using existing monitoring architectures and techniques.

UML-M defines modeling elements for representing QoS monitoring information and generic QoS monitoring functional units. In UML-M the QoS monitoring information is based on events that are generated by the monitored application or are "extracted" from it by using the appropriate mechanisms (probes). Those events can be processed in different stages until the final (and useful) monitoring information arrives to the corresponding managing entity. The monitoring events can be correlated (for obtaining new events), and filtered (for selecting the interesting monitoring information). They can also be used for calculating metrics (measurements of QoS characteristics of interest, QoS characteristics that may have been defined by means of UML-Q). Those metrics can be reduced (metrics are used for calculating new ones) and can be monitored (in order to test if a particular QoS requirement is being satisfied).

According to the ODP framework that has been chosen as the conceptual base for the contributions presented in this paper, it is important to point out that the introduction of QoS monitoring mechanisms in distributed object-based applications should be done at the ODP engineering level. The reason for that is that by using this approach, the developers of the functional aspects of the monitored application (that focus their efforts in the enterprise, information and computational ODP viewpoints) are, in the majority of cases, not affected by the introduction of those QoS monitoring functions. According to this fact, UML-M can be considered as a complement to UML when it is used as a language for the ODP engineering viewpoint. In this sense, some stereotypes for representing ODP engineering concepts have been defined as a previous stage to the specification of UML-M.

diagram shown in Figure 5 can be instantiated into an object diagram with concrete values assigned to the attributes of those classes. Thus, for example, the monLanguage attributed of an instance of the DelayCalculator stereotyped class of Figure 5 might have the value "AML" in order to indicate that the behavior of that metric calculator is going to be determined by an AML expression. AML (*Activity Monitoring Language*) is a particular monitoring language proposed in [5]. This example shows how UML-M is able to reuse existing proposals related to the monitoring of distributed object-based applications.

It is important to mention the tagged value {mgmtType=JIDM} attached to the TradingMgmtHandler instance of the MonQOSStore virtual meta-class. This is a clear example of how UML-M models are intended to be implemented by using concrete techniques. In this case, that tagged value indicates that

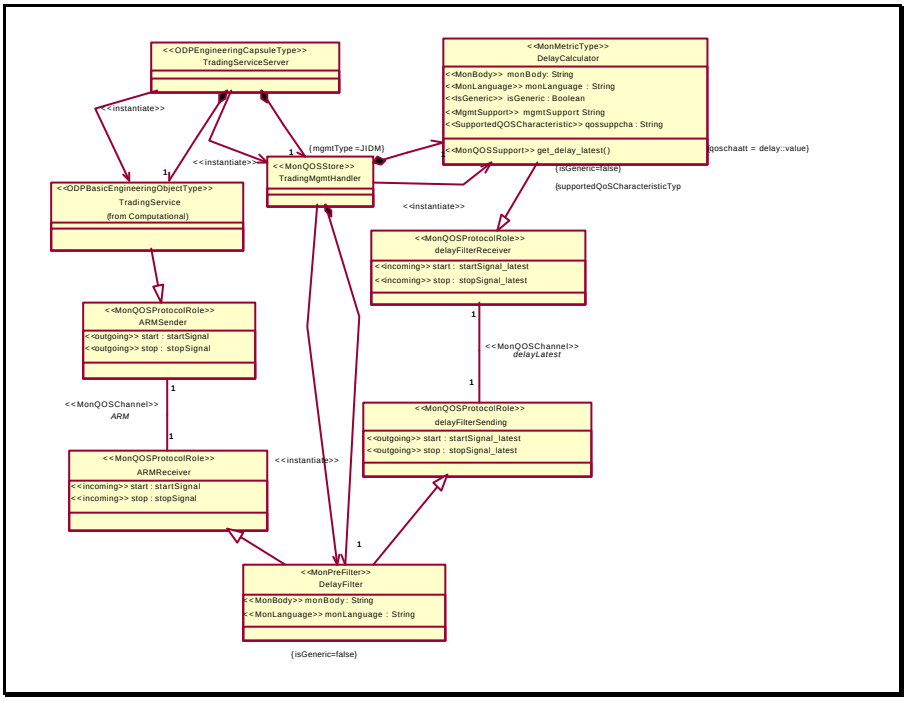

Figure 5. ***A UML-M example.***

UML-M defines a set of modeling constructs by extending the UML meta-model in a similar way as UML-Q and thus, creating a new "virtual meta-model".

Figure 5 presents a simple example of how UML-M can be used. That figure shows a basic ODP engineering object (TradingService) that is the engineering view of the TradingService computational class shown in Figure 2.

That basic engineering object is maintained by an engineering capsule called TradingServiceServer. The TradingService basic engineering object generates start/stop events associated to the invocation of its operations that are filtered by the DelayFilter instance of the MonPreType UML-M virtual meta-class in order to select only the monitoring events associated to the latest() operation. Those events are processed by the DelayCalculator instance of MonMetricType in order to generate values of the delay_latest QoS Characteristic. This is basically the meaning of the contents of Figure 5. The class TradingMgmtHandler is going to implement management interfaces that are compliant with the JIDM (Joint Inter-Domain Management) [10][2] specification and interaction translation rules in order to allow the access to the QoS monitoring information (obtained by the monitoring functions contained within TradingMgmtHandler) from an SNMP-CORBA gateway. Nevertheless, this fact does not affect the way the monitoring infrastructure has been modeled. This is the idea behind UML-M: details related to concrete monitoring techniques or architectures can be avoided at the design stage. Even those details might be completely avoided by the developers if the appropriate development tools are available (according to the presented example, one of those tools might map the UML-M model into a JIDM-like monitoring infrastructure for CORBA-based applications. There might be other alternatives such as JMX, CIM, etc.).

6. JOINT USAGE OF UML-Q AND UML-M

In order to illustrate how UML-Q and UML-M can be jointly used, this section briefly presents the experience of applying both of them to the development of a concrete example of distributed object-based e-commerce application: a prototype, based on CORBA (using JAVA as the programming language), of an electronic information and services broker [2]. That prototype was developed within the scope of the ABS project (Architecture for an information Brokerage Service) funded by the European Union.

Figure 6 summarizes how UML-Q and UML-M were used during the development of the ABS application in order to obtain a prototype of an e-commerce application whose QoS might be monitored. In this case, and from an implementation point of view, a JIDM-based SNMP-CORBA gateway was developed in order to integrate the QoS monitoring of the application with network and system management aspects [2]. CORBA interceptors were the basic instrumentation mechanism used to make the application instrumentation transparent to the developers of the functional aspects of the electronic broker. All the implementation process was based on the UML-Q and UML-M models. Those models might have also been used as the starting point for other types of application instrumentation and integrated management (a JMX approach instead of JIDM-CORBA, etc).

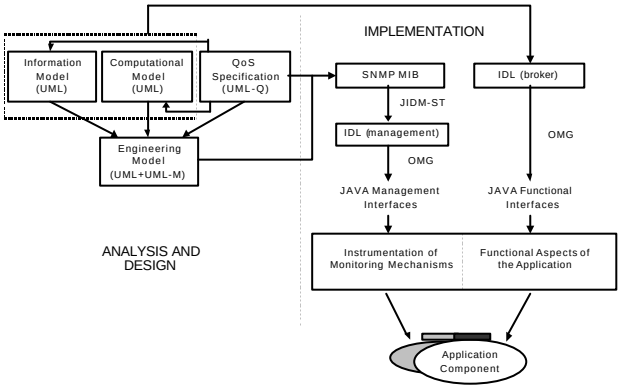

Figure 6. ***UML-Q and UML-M applied to the development of a distributed object-based application.***

7. CONCLUSIONS AND FUTURE WORK ITEMS

This paper has presented the UML-Q and UML-M UML profiles as a way of facilitating the development of distributed object-based applications whose QoS can be monitored in an integrated way and separately from the development of the functional aspects of the application to be monitored. UML-Q and UML-M take advantage of the adoption of UML as the de facto standard language for OO analysis and design. Both UML-Q and UML-M have been applied to the development of a CORBA-based application with support for QoS integrated monitoring using SNMP managing tools [2].

One of our future work items is the extension of existing UML-based CASE tools in order to support UML-Q and UML-M so as to facilitate and automate, as much as possible, the development of object-based applications with QoS monitoring support.

8. REFERENCES

[1] S.S. Alhir. "Extending the Unified Modeling Language (UML)". Internet Document http://home.earthlink.net/~salhir/ExtendingTheUML.PDF, 1999.

[2] J.I. Asensio, V.A. Villagrá, J.E. López de Vergara, and J. Berrocal. "Experiences with the SNMP-based integrated management of a CORBA-based electronic commerce application". In Proceedings of the sixth IFIP/IEEE International Symposium on Integrated Netork Management. Boston, MA, USA. May 1999. IEEE Publishing, 1999., pages 517--530. 1999.

[3] Campbell, C. Aurrecoechea, and L. Hauw. "A review of QoS Architectures". ACM Multimedia Systems Journal, 1996.

[4] S. Frolund and J. Koistinen. "QML: A Language for Quality of Service Specification". Technical Report HPL-98-10, HP Laboratories, 1998.

[5] S. Frolund, M. Jain, and J. Pruyne. "SoLOMon: Monitoring End-User Service Levels". Technical Report HPL-98-153, HP Laboratories, 1998.

[6] M.J. Katchabaw, S.L. Howard, H.L. Lutfiyya, and M.A. Bauer. "Making distributed applications manageable through instrumentation". In Proceedings of PDSE'97, Boston, Massachusetts, USA, May 1997.

[7] Object Management Group. "Meta Object Facility (MOF)". OMG document ad/99-09-05, September 1999.

[8] Object Management Group. "OMG Unified Modeling Language Specification version 1.3". OMG document ad/99-06-08, June 1999.

[9] Object Management Group. "White Paper on the Profile Mechanism". OMG document ad/99-04-07, April 1999.

[10] The Open Group, "Inter-Domain Management: Specification and Interaction Translation". Open Group Document C802, January 2000

[11] Hafid, G. von Bochmann, and R. Dssouli. "Distributed Multimedia Applications and Quality of Service: a Review". The Electronic Journal on Networks and Distributed Processing, issue 6, February 1998.

[12] ISO. "Working Draft for: Open Distributed Processing - Reference Model - Quality of Service". ISO/IEC JTC1/SC N 10979 Ed 6.4, January 1998.

[13] ITU-T. "Information Technology - Open Distributed Processing - Reference Model: Overview". ITU-T Recommendation X.901 (ISO/IEC DIS 10746-1), December 1997.

[14] ITU-T. "Information Technology - Quality of Service: Framework". ITU-T Recommendation X.641 (ISO/IEC IS 13236), December 1997.

[15] MCI Systemhouse. "Relationship of the Unified Modeling Language to the Reference Model of Open Distributed Computing". MCI Systemhouse white paper, September 1997.