A QoS Meta Model to Define a Generic Environment for QoS Management

Jérôme Daniel [1,2], Bruno Traverson [1], and Sylvie Vignes [2]

[1] EDF Division R&D, 1 avenue du Général De Gaulle,
F-92140 Clamart, France
{ Jerome.Daniel, Bruno.Traverson } @der.edf.fr
[2] ENST, 46 rue Barrault,
F-75634 Paris Cedex 13, France
Sylvie.Vignes@enst.fr

Abstract. The management of Quality of Service (QoS) in Information Systems allows their users to request services under certain conditions varying in function of their requirements but also in function of the current capabilities of the computing environment. The kind of the requirements and the notation used to formulate them vary with regards to application area and computing environment used. This paper presents a generic architecture that can manage QoS independently of the QoS domain and of the distributed environment used. We also introduce a QoS meta model (defined with a meta modeling facility – the so-called MOF – standardized by OMG) to exchange QoS models that allows a generic approach for QoS notation. By this way, this QoS management architecture is the first and only one platform which can manage all QoS contract properties (which are guaranty, observation, negotiation and composition) under a generic approach with regards of domain, environment and notation.

1 Introduction

The constant improvement of computers and systems has allowed the definition and the development of high level distributed environments. These environments, which are able to provide very wide functionality to their users, are now more and more used. However, the users are more demanding than in the past, because they do not only want a service but they want to get a service under some conditions (guaranty of response time, proximity constraint, and so on).

These concepts are encompassed in the term: Quality Of Service (QoS). The mechanisms to apply in order to manage QoS are very difficult to define and develop in existing systems. Moreover, several problems are implied by this management as for example the specification of user requirements, but also their observation and guaranty during the application life cycle.

Very often, there are several solutions to manage QoS that are dedicated to a specific domain. For example, some works suggest a QoS management for multimedia

C. Linnhoff-Popien and H.-G. Hegering (Eds.): USM 2000, LNCS 1890, pp. 334-339, 2000.

systems or QoS management for telecommunication systems (where time, jitter, … are taken into account). But, there is no platform to manage in a generic way all domains of requirements for QoS. The main reason for this, is the difficulty to express in the same notation distinct QoS approaches.

OMG (Object Management Group) defines a meta modelling facility (MOF) [3] that gives numerous advantages as for example the exchange of models via XMI (XML Metadata Interchange) [7] which is a specific XML (eXtended Markup Language) [8] document.

Several application domains have defined their own model to manage QoS with for example a dedicated QoS notation. As we will see in the fourth part of this paper, we have also defined such a model. If we use MOF to create a model of our QoS model (a QoS meta model), we will be able to use XMI to exchange QoS models.

By this way, any QoS notation could be used to express QoS offers and QoS requirements and thus we will also have a generic approach to describe QoS. In conclusion, our platform will be generic with regards to QoS notations.

But is it always possible to transform a QoS model to another one? It is very difficult to provide a definitive answer to this question, but if we have a very wide QoS model that is able to manage any QoS domains, its QoS meta model would be the best candidate for enabling the transformation of one QoS model to this QoS model (because this one is the most complete model).

Moreover, if we use MOF, we will model our QoS model (and by this way our QoS platform and mechanisms). Thus, our technical approach to develop this QoS management architecture will be more efficient and reliable.

At least, with our QoS meta model and MOF capabilities, we will be able to automatically get the IDL interfaces for the QoS model creation and we will be able to maintain a QoS model repository. By this way, it will be easier to lookup for an asked QoS description into the QoS model repository.

This paper is divided in two parts. In the first part, we explain and define our QoS meta model and an example that uses it. In the second part, we expose our generic architecture to manage QoS in distributed environments.

2 A QoS Meta Model

In this part, we describe our QoS meta model. This part is divided in three sections that. The first section explains the needs for a QoS meta model. The second section describes the QoS meta model. Finally, the third section presents the overall QoS management architecture that includes the MOF capabilities.

2.1 The Description of the QoS Meta Model

Note: we are using an UML notation to represent our QoS meta model.

In our model, each functional object is linked to a QoS object. This QoS object contains one QoS offer, and several QoS requirements. A QoS offer is a set of properties where each property can be basic (a simple value and a name), evaluated (requires an external evaluator) or constrained (requires some other QoS objects to be evaluated, this kind of properties is a constraint to some other properties of one or more QoS objects). In the same way, a QoS requirement is a constraint that is applied to a set of properties. The following figure summarises this first subset of the meta model.

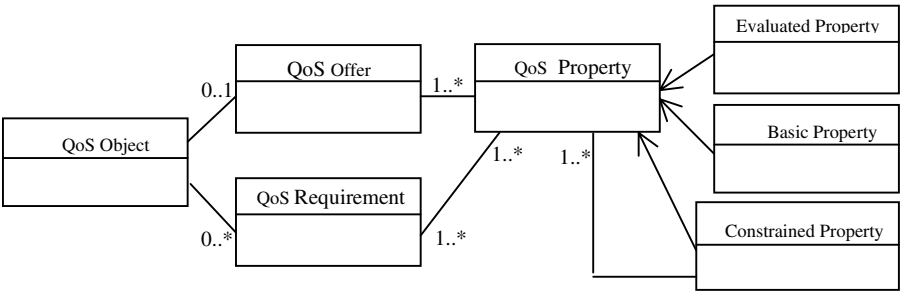

Fig. 1. First subset of the QoS meta model

Two QoS Objects can be linked by a composition (if the first object requires the second one to provide a QoS offer, it means in this case that the first object contains a constrained property on the second object). Moreover, when a QoS contract is established, it contains several QoS objects and more precisely several QoS compositions.

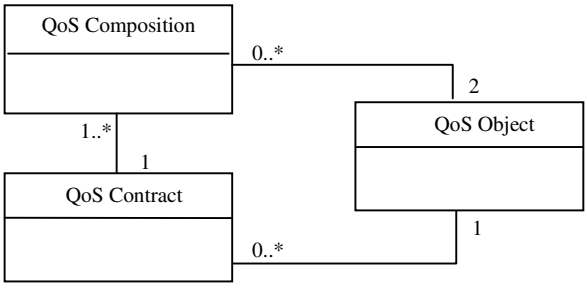

Fig. 2. A QoS contract into the QoS meta model

With these two simple meta models, we have describe all possible links and interactions between QoS Object, thus the QoS meta model is the union of them. To conclude this meta model presentation, let us take a simple example to discover that it corresponds to the QoS meta model.

2.2 A QoS Model Example: A Movie on Demand System

In this example, we distinguish four actors: a user (named foo), a network and two movie servers (A & B). Mr Foo requires a movie with a price and with a good transmission speed.

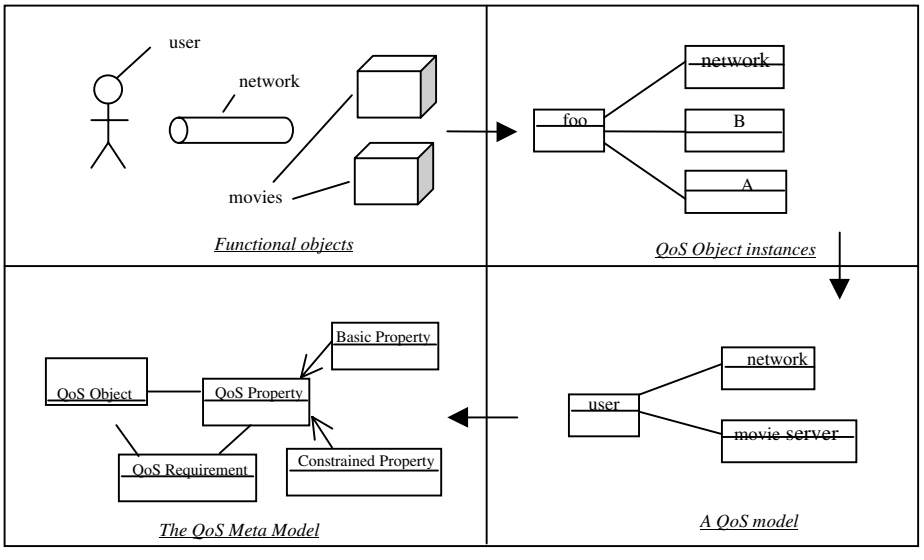

Fig 3. An example of QoS instances, objects, model and meta model

At the left upper corner, the figure illustrates the functional objects which are modelled in QoS by the QoS Objects represented on the right side. These QoS objects are linked by QoS relationships. QoS Objects « A » and « B » are two instances of the same QoS Object.

The corresponding QoS model is shown on the right bottom corner. Here we only have three QoS objects, the user QoS Object (which is only a requirement) and two other QoS objects (network and movie server) which provide QoS offers.

The model can be modelled and the result is illustrated on the left bottom corner. This meta model is compatible with our QoS meta model.

3 A Generic Architecture to Manage QoS in Distributed Environments

Since we have a QoS meta model, we can now use the MOF capabilities to automatically generate a QoS model repository and a DTD for XMI documents.

With XMI and our QoS models DTD, we can exchange QoS models and by this way be independent of QoS notations. For example, we have also defined a QoS description language called « QDL » [Daniel 99] that we have modelled to be able to generate an XMI document from this formalism.

QDL is a language that is very similar to IDL and particularly adapted for QoS notation about distributed objects and their properties. With a dedicated XML document translator like XLST (XML translator) we can automatically translate an XML document to another one. So a QDL description could be expressed with an XMI document that could be translated to another XMI document which respects our QoS model DTD. Then, this QoS notation could be used and managed by our generic QoS management architecture. The translation is always possible because the QoS meta model includes all possibilities for QoS compositions, offers and requirements.

The following figure summarises the architecture of our generic QoS management infrastructure.

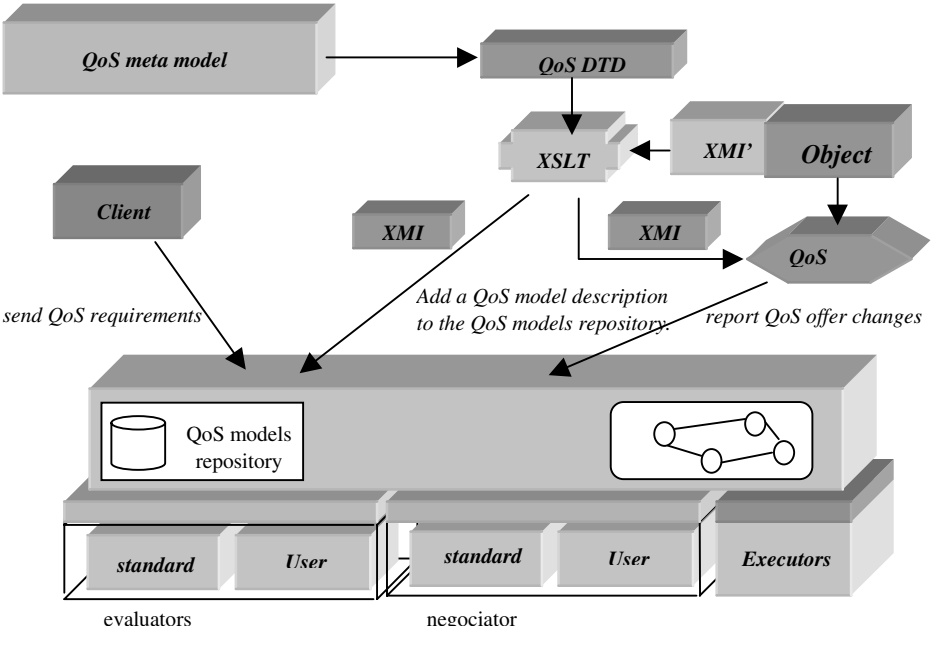

Fig. 4. An overview of the generic QoS management infrastructure

4 Conclusion

In this paper, we have described our approach to manage QoS and our QoS management infrastructure architecture. After that, we have concluded that a QoS model exchange is required to provide a generic QoS management for QoS notations.

This is the reason why we propose to use the MOF (Meta Object Facility) which provides several capabilities and in particular a way to exchange models. Then, we have presented our QoS meta model that can model any QoS model (with compositions, offers and requirements).

At last, we have extended our QoS infrastructure to include the QoS meta model capabilities. This, we can provide a generic QoS management architecture: generic about QoS domain, generic about environment, and generic about QoS notation.

Currently, we are developing such an architecture using Java for CORBA (with JavaORB [2]). The prototype will give us a way to validate our QoS management mechanisms and to observe potential performance problems. Moreover, into component platforms, configuration files and properties are usually expressed with XML. We suggest adding an additional descriptor for QoS (QoS Descriptor) that includes all QoS requirements and offers for the corresponding component. By this way, it will be possible to manage components with our QoS management architecture. So, we envisage applying in future this possibility into a component platform (such as a CORBA component platform or an Enterprise Java Bean platform).

References

1. Daniel, J., Traverson, B., Vignes, S.: Integration of quality of service in distributed object systems. International Conference on Distributed Applications and Information Systems (1999).
2. Distributed Object Group: JavaORB a free CORBA implementation". http:\\dog.exoffice.com.
3. OMG: Meta Object Facility (MOF) version 1.3. OMG Document (July 1999).
4. ISO/IEC: Open Distributed Processing Reference Model, parts 1, 2, 3, 4. ISO/IEC IS 10746-1.. 4 or ITU-T X901..4 (1995).
5. ISO/IEC: Open Distributed Processing Reference Model, Quality of Service. ISO/IEC Working Draft (January 1998).
6. OMG: The Common Object Request Broker Architecture and Specification, Revision 2.3. OMG Document (June 1999).
7. OMG: XMI (XML Meta Data Interchange). http://www.omg.org.
8. W3C: Extensible Markup Language (XML) 1.0. W3C Recommendation 10 Reference : REC-xml-19980210. http://www.w3.org/TR/REC-XML. (February 1998).