Model Management for Quality of Service Support

Brigitte Kerhervé[†], Olivier Gerbé[‡]

[†]Université du Québec à Montréal
Département d'Informatique
CP 8888, succursale centre ville
Montréal, H3C3P8, Canada
E-mail: Kerherve.Brigitte@uqam.ca

[‡]HEC Montréal
3000 Chemin de la Côte Sainte Catherine
Montréal, H3T 2A7, Canada
E-mail:Olivier.Gerbe@hec.ca

Abstract

Quality of Service (QoS) management strategies have been first introduced in distributed multimedia systems to decide whether and controlling how multimedia streams can be delivered to the user within the given delay, cost or quality constraints. With the recent advances in the development of large-scale distributed applications where different services are provided to a large number of users, QoS management becomes and end-to-end functionality requiring the cooperation of all the system components. That leads to consider system components interoperability, management information integration and distributed execution of QoS activities. In this paper we examine modelling and meta-modelling issues for QoS management. More specifically we propose a meta-model for QoS management and we illustrate how such a meta-model is used for mapping QoS requirements to system constraints.

Keywords: Quality of Service Support, Model Management, Metamodels, Distributed Multimedia Systems

1. Introduction

In recent years, considerable research effort has been dedicated to Quality of Service (QoS) management, mainly in the field of telecommunication networks and multimedia systems [12]. This effort led to proposals for QoS management strategies aimed at deciding whether and controlling how multimedia streams can be delivered to the user within the given delay, cost or quality constraints. These constraints are expressed during a specification step where the user specifies his requirements which may concern system performance, such as the delay needed to transfer objects, the quality of the provided information or the financial costs attached to document delivery. The system then works to deliver the specified level of service and for that purpose transforms the users' requirements into various constraints targeted at the system components: client machines, database systems, server machines or transport system.

While considering QoS in the framework of emerging applications such as digital libraries or electronic commerce, or more generally electronic services, we need to give a broader definition of QoS [3]. The traditional categories and dimensions used to express QoS requirements, such as performance, reliability or cost need to be extended to integrate notions such as security, data quality or availability. Such categories and dimensions constitute our QoS specification metadata model that is a data model for QoS specification information. This metadata model has to be extensible in order to allow addition of categories and dimensions specific to new applications.

From the system perspective, QoS requirements have to be mapped onto system constraints for resource allocation compatible with the requested level of service. We then need to manage metadata for declaring the QoS level offered by the components of the distributed system such as the network performance or the transaction throughput of the database system. Once again, this declaration is done according to QoS categories and dimensions, which are more-system oriented and correspond to technical characteristics of the components. These categories and dimensions constitute our QoS declaration metadata model.

It clearly appears that such metadata models for QoS specification and declaration cannot be enough general, nor enough specific, to support QoS specification for all types of applications, or QoS declaration for every system component. These metadata models have then to be extensible and adaptable. For example, one can refine the *image quality* category in incorporating the *color definition* dimension for QoS specification, or drop the *time-to-repair* dimension of the *reliability* category for the QoS declaration of a system component. For that purpose, we are working on a QoS metadata engine allowing to define, store and manage QoS metadata models for users and system components [9]. This engine is based on a metamodel we propose for QoS management.

In this paper we examine modeling and metamodeling for QoS management. More specifically we show that QoS support requires to propose models and metamodel for describing QoS information. Such models are used during the different QoS activites for mapping requirements to system constraints, for exchanging QoS informations and for checking compatibility between QoS informations.

The paper is organized as follows. Section 2 gives an overview of QoS management and points out different modeling problems. Section 3 presents the models and metamodels we propose for generic and extensible QoS management. Section 4 describes how the QoS mapping activity can be supported through model and metamodel manipulations. Section 5 concludes and presents our future work.

2. Quality of Service Management

QoS research activities are mainly conducted in the field of telecommunication networks and multimedia systems where the concept of QoS was first introduced. They have led to proposals for management strategies whose purpose is to decide whether and how multimedia streams can be delivered to the user within the given delay, cost or quality constraints. In the case of distributed multimedia systems, QoS cannot be evaluated at the

communication level only. QoS can be appreciated by the users, who should have the opportunity to describe their requirements. In multimedia systems, QoS management can thus be viewed as an essential end-to-end functionality that should be an integral part of the system [8].

In this section we first present architecture and components for distributed multimedia systems, then we explain and illustrate the different activities required for QoS support in such systems.

2.1 Distributed multimedia systems

In the last two decades, we have been faced to tremendous evolution of distributed multimedia systems in order to support emerging applications such as electronic commerce, health-care applications or digital publishing. These applications integrate large amounts of voluminous, heterogeneous and/or time-dependent data. Data are located on several sites interconnected through various communication networks and potentially accessed by a large number of users using mobile computing equipment. The system architecture supporting such applications is heterogeneous, consisting of a large number of client machines, database servers, video servers or other specific servers, all interconnected through communication networks. Such complex environments require the integration of system management mechanisms providing system scalability, application adaptation and QoS support [4]. All the components of the distributed multimedia system have to contribute to this tasks and each of them should include specific mechanisms to support QoS locally and to provide QoS information for global, distributed QoS decision.

Implementing QoS mechanisms in such a complex environment lead to consider different issues: system components interoperability, management information integration and distributed execution of QoS activities. In this paper, we focus on a subset of management information, namely QoS information in order to support distributed execution of QoS activities.

2.2 QoS management activities

One can identify five distinct activities in QoS management: specification, mapping, negotiation, adaptation and monitoring. QoS specification consists of identifying the dimensions of QoS (time, cost or quality, for instance), and of defining the QoS level requested by the user and supported by the components of the distributed multimedia system. QoS mapping consists of mapping the user's requirements onto QoS parameters such as resolution, frame rate or throughput. Since these parameters pertain to different layers, one must set up a contract between the various layers and components of the distributed multimedia system in order to satisfy the user's requirements. This phase is called QoS negotiation and may be followed by an adaptation phase, since new situations may lead to renegotiation. Finally, it is necessary to examine the actual QoS level and compare it with the initial requirements. Thus any distributed multimedia system must include a QoS monitoring function.

Figure 1 illustrates the different components involved in a video-delivery application. The service provided by this application allows a user to express a query on a video database in order to select a pertinent video to be transferred. The database server stores metadata describing videos and their content and process queries to identify the video sequence that will be delivered to the user from the video server.

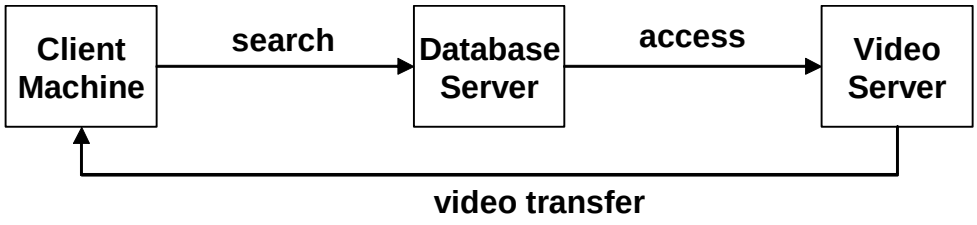

Figure 1 – Components involved in a video delivery application

2.3 Modeling issues

Introducing QoS support for such an application requires the integration of QoS mechanisms in each component. In the client component, we should find a QoS specification mechanism allowing the user to express his requirements concerning the service level he is interested to get. This level of service can be expressed through qualitative parameters directly perceived by the user, such as the video quality having the following possible values: *excellent, good, bad*. These qualitative parameters have then to be transformed to quantitative parameters directly measurable by the system components such as *frame rate* or *network throughput*. This mapping mechanism can be implemented on one or several components and requires mapping rules expressing the correspondence between qualitative and quantitative parameters. The QoS negotiation step is supported by a distributed mechanism implemented on the different components (client machine, database server, video server and telecommunication network). This step requires information describing the level of service provided by each different component and obtained through component monitoring. Such information can be homogeneous and obtained through different mechanisms, QoS negotiation step then needs QoS information integration as well as compatibility checking.

We can see that QoS support requires the management and transformation of QoS information describing the performance, availability or reliability of the different components of the distributed multimedia system. In wide-area systems, integration, federation and inter-operation of QoS information bases (QoSIB) should be provided. For that purpose, we are currently working on the design and implemention of an extensible QoSIB manager offering basic services to store, access, share, transfer, produce or analyze QoS information. This QoSIB manager should be extensible in the sense that it should offer mechanisms to integrate new QoS information and services. We believe that this can only be achieved by working on model and meta-model management for QoS information.

3. A meta-model for Quality of Service Management

This section proposes a meta-model for QoS management. In order to avoid confusion between modelling levels when using object-oriented formalism [2], we use a neutral formalism, conceptual graphs, to illustrate the QoS meta-model. Before presenting our proposition, we first introduce conceptual graphs and present our layered architecture.

3.1 Conceptual Graphs

J. Sowa introduced conceptual graphs (CGs) in 1984 [10]. They form a coherent system for the graphical representation of logic based on the existential graphs of C.S. Peirce [7] and semantic networks. This section presents the formalism of conceptual graphs. Only a minimum explanation is provided as required by the rest of the paper. More information on conceptual graphs can be found [11],[5].

Conceptual graphs are a formalism whereby the universe of discourse can be modeled by concepts and conceptual relations. A concept represents an object of interest or knowledge. A conceptual relation makes it possible to associate these concepts. Conceptual graphs define knowledge both at the type and instance levels. Concepts are represented by boxes and relationships by circles with arrows that link the concepts associated. Figure 2 represents the sentence “John is an employee that works for the University of United Nations (UNU)”.

Figure 2 - Conceptual Graphs: Concepts and Relationships

The same sentence may be represented in the following textual linear form:

```
[Employee:John]->(works-for)->[Organization:UNU]
```

Concepts may be categorized based on the type of conceptual relations they have with other concepts. Concept types define these categories. A concept type is defined by a definition graph to which any instance of that concept type must comply with. Figure 3 presents the definition graph of EMPLOYEE that means that an employee is a person that works for some organization.

```
Type employee(x) is
  [Person:x]->(works-for)->[Organization]
```

Figure 3 - Conceptual Graphs: Type Definition

There exists an operator called ? that allows translation of CGs to first order predicate logic. The statement "There exists a cat that is on a mat" may be rendered in linear form as

```
[Cat]->(on)->[Mat]
```

and in first order predicate logic as

```
(?x)(?y) (Cat(x)?Mat(y)?on(x,y))
```

Other research establishing the basis of correspondence between CGs and First Order Predicate Logic may be found in [1].

3.2 Model, Meta-model and Meta-meta-model

Before presenting our QoS Meta-model, we will define the following terms: model, meta-model and meta-meta-model.

In our context we define a model as an abstract representation of something that happens in the real world. A model is a simplification of some system and this is done with some goals in mind. Only pertinent details of the system that are related with these goals are represented in the model. The way and the vocabulary we will use to build models form the meta-model. At least the meta-meta-model defines the language used at the meta-model and model level.

Figure 4 illustrates the four layers architecture where:

- ? M3 is the meta-meta-model level. It contains the basic element of conceptual graphs: concept and conceptual relation.
- ? M2 is the meta-model level. Here is a subset of the meta-model for QoS Management. It contains all the vocabulary used at level M1 for example, Dimension, Value and Declaration.
- ? M1 is the model. It represents a particular environment. It defines types and instances that represent the real world.
- ? M0 is the real world where takes place the situation described at level M1.

In Figure 4, we see two kinds of relationships meta and instOf (instance of) that are often confused in modeling activities. This confusion comes from natural language where we use the same verb *to be* in "2 seconds is the response time" and in "2 seconds is a value". In order to avoid such confusion we called instOf the relationship in the former and meta in the latter. Others relationships shown in this example are contains and chrc (characteristics).

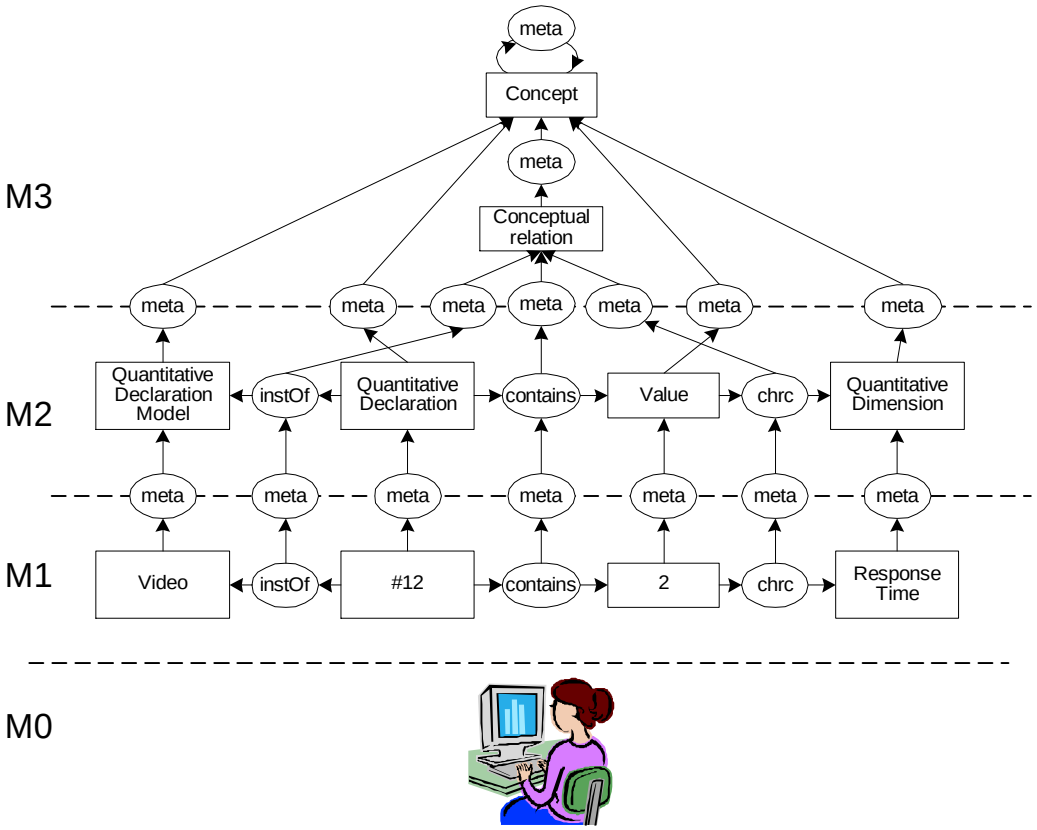

Figure 4 : Four Layer Architecture

3.4 The QoS Meta-model

The QoS meta-model specifies all the vocabulary used to describe the needs in QoS management and QoS activities.

QoS specification deals with the definition of the requested QoS level, and is provided by the user or by the application programmers. The definition is made using dimensions belonging to QoS categories such as performance or cost. A dimension can be defined as a qualitative or quantitative attribute of a category [6]. An example of a dimension in the *performance* category is *throughput*. In most of existing QoS approaches, categories and dimensions are more performance-oriented. They were generally proposed within the framework of distributed multimedia systems in order to support audio and video transfer with synchronization constraints. While considering QoS in the framework of emerging applications such as digital libraries or electronic commerce, we need to give a broader definition of QoS. The traditional categories and dimensions of QoS, such as performance, reliability or cost need to be extended to integrate notions such as data quality or security.

The QoS information, built with the concepts of dimensions and categories, is modeled in QoS information models. We have defined five types of QoS information models allowing a simple, homogeneous and extensible way of describing QoS information. The five types of models we propose are: Core Model, Environment Model, Qualitative Specification Model, Quantitative Specification Model and the Quantitative Declaration Model. They are all built with the concepts of category and dimension of QoS.

The first model, Core Model is unique and contains the set of generic categories and dimensions relevant for all types of application environments. Environment Models are QoS categories and dimensions relevant to an application environment, they are derived from the Core Model through selection and specialization of relevant categories and dimensions. The same derivation mechanism is applied to the Environment Model to generate Specification and Declaration models. Specification models allow the expression of constraints on QoS

dimensions to specify the expected level of service. Declaration models allow to associate values to QoS dimensions to express the level of service provided by a system component.

A Qualitative Specification Model corresponds to set of high level QoS categories and dimensions a user or an application needs for specifying the required QoS. The Quantitative Specification Model is computed from the Qualitative Specification Model. Its QoS categories and dimensions are the same than the Quantitative Declaration Model but it consists of constraints while the Quantitative Declaration Model consists of values.

A quantitative Declaration Model corresponds to QoS categories and dimensions along which the QoS is supported by a system component (communication network, database system, video server etc…).

Figure 4 presents our meta-model. Models presented on the left side are: Core Model with its derivation Environment Model with its own derived models, Qualitative Specification model, Quantitative Specification Model and Quantitative Declaration Model. At the top, the Core Model organizes categories that group dimensions. We distinguish Qualitative Dimension with its definition domain and its constraints and Quantitative Dimension with its definition domain, its constraints and its values. At the bottom, Application is modeled by the Environment Model and built with Component Types. Just above, the User performs tasks for which he or she defines Quality of Service requirement through a Qualitative Specification.

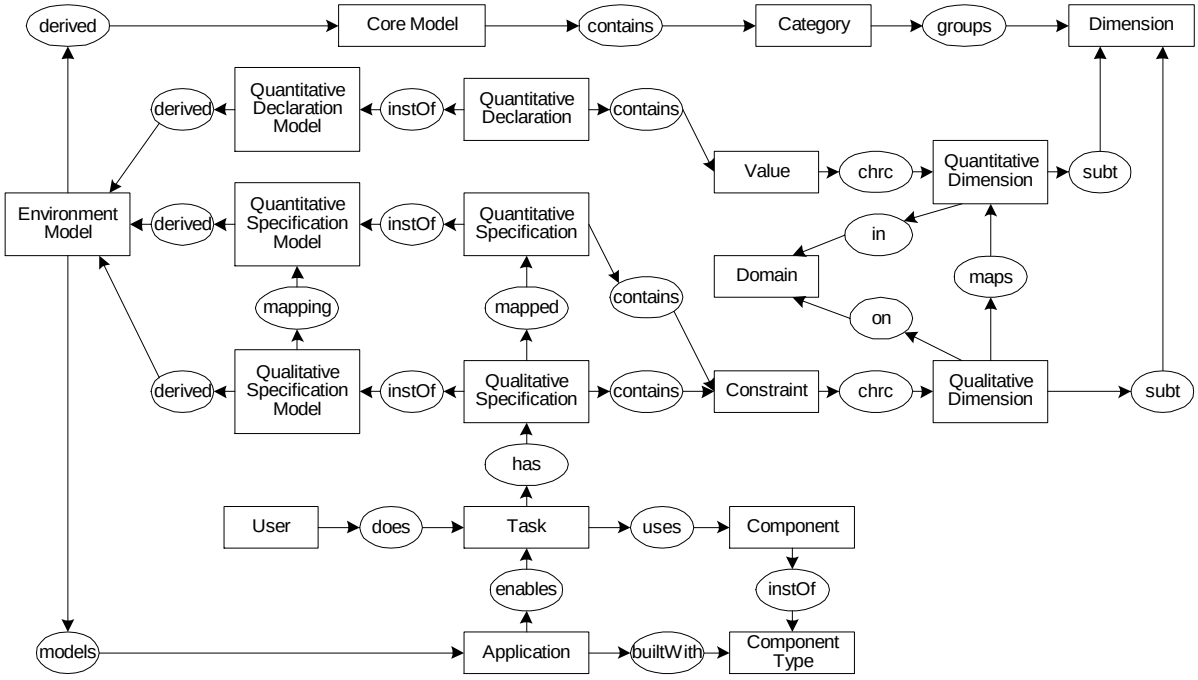

Figure 5 - QoS Metamodel

It is important to note that there are relationships between the different models. Indeed, dimensions and categories of a model can be related to those of its parent model. This relation is a derivation relation. The derivation is a process aiming at creating a QoS information model from a parent model by selection of its categories and dimensions.

Models and derivation mechanisms are the kernel of our QoS IB manager built to store, access, share, transfer, produce or analyze QoS information. Two prototypes of QoS information declaration model are currently developed, one for communication networks and one for database systems.

4. QoS support through Models Manipulations

In this section we examine how QoS mapping can be supported through model manipulations. We have previously seen that QoS mapping consists in transforming user requirements expressed in terms of qualitative

parameters into quantitative parameters that can be directly measured on system components. We can classify QoS mapping into two sub-classes: (i) qualitative-to-qualitative mapping and (ii) qualitative-to-quantitative mapping.

Qualitative-to-qualitative mapping consists in recursively decomposing a qualitative QoS specification to a lower level qualitative specification. For example, for a video delivery application, we can define the highest QoS specification level according to only one qualitative dimension: *VideoServiceLevel* that can take the following values: excellent, good, bad. This high level specification can be expressed in terms of three lower level qualitative dimensions such as: *VideoQuality*, *AudioQuality* and *ResponseTime*.

Qualitative-to-quantitative mapping consists in decomposing a qualitative QoS specification into a quantitative specification. For our video delivery application, we can define the qualitative *VideoQuality* QoS specification level according to four quantitative dimensions: *FrameRate*, *NbBitsColor* and *Format*. These quantitative dimensions can be directly measured for a given video sequence. Figure 6 shows the model manipulations required for qualitative-to-qualitative mapping and for qualitative-to-quantitative mapping.

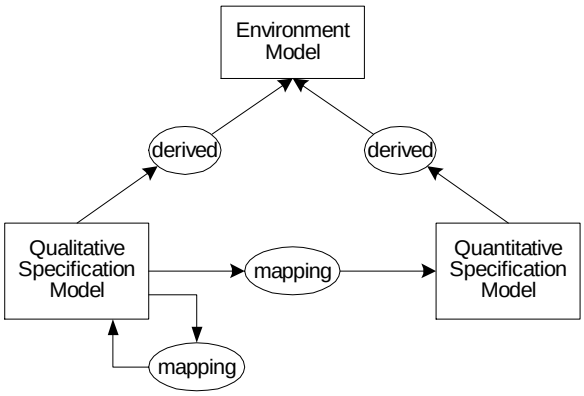

Figure 6 -QoS Mapping

5. Conclusion and Future Work

In this paper we proposed a meta-model for QoS management. This meta-model specifies all the vocabulary used to describe the needs in QoS management. We have presented this meta-model with conceptual graphs, a neutral formalism that simplifies and clarifies the description and explanation of the relationships between modeling layers and between models. We have illustrated the use of this meta-model to support a specifc and important activity, namely QoS mapping. We have shown that qualitative-to-qualitative mapping and qualitative-to-quantitative mapping can be supported through model transformations.

This meta-model and the model transformations we propose are the kernel of an extensible QoS information base (QoSIB) manager we are currently designing and implementing to offer basic services to store, access, share, transfer, produce or analyze QoS information. A first prototype has been implemented for model management and for the management of network and database system QoS information. We are currently working on the basic model transformation operations to support more complex QoS activities such as negotiation and adaptation.

6. Acknowledgements

Special thanks are due to Pierre-Emmanuel Ciron, Kim-Khoa Nguyen and Laila Fetjah for the work done on the prototype implementation.

7. References

[1] Amati, G. and Ounis I. Conceptual Graphs and First Order Logic. *The Computer Journal*, 43(1), 2000.

[2] Bézivin, J. and Gerbé O. Towards a Precise Definition of the OMG/MDA Framework. In *proceedings of the 16th IEEE International Conference Automated Software Engineering*, San Diego, USA. November 2001.

[3] Bochmann G., Kerhervé, B., Salem, M., Quality of Service Management Issues in Electronic Commerce, chapter 14, *Electronic Commerce Technology Trends: Challenges and Opportunities.* IIR Publications, Inc., MidrangeComputing, 2000, pp 227-238.

[4] Bochmann G., Kerhervé B., Lutfiyya H, Salem M, Ye H. Introducing QoS to Electronic Commerce Applications. In *Proceedings of the 2nd International Symposium on Electronic Commerce (ISEC)*, Hong-Kong, 26-28 April, 2001, Proceedings published by Springer-Verlag.

[5] Chein, M. and Mugnier M.L. Conceptual Graphs: Fundamental Notions. *Revue d'intelligence artificielle*, 6(4):365-406, 1992.

[6] Frolund, S., and Koistinen, J. (1998). Quality-of-Service Specification in Distributed Object Systems. *Distributed Systems Engineering Journal* (December 1998), 5(4):179-202.

[7] Houser, N., Roberts, D. and Van Evra, J. *Studies in the Logic of Charles Sanders Pierce.* Indiana University Press, 1997.

[8] Nahrstedt, K. (1995). End-to-End QoS Guarantees in Networked Multimedia Systems. *ACM Computing Surveys*, 27(4):613-616, December 1995 .

[9] Nguyen K., Fetjah L., Kerhervé B., 2001. Quality of Service Information Base Manager for Electronic Commerce Applications. Poster presented at the *CITR Annual Conference*, Aylmer, Canada, August 2001.

[10] Sowa, J. *Conceptual Structure: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[11] Sowa, J. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. BrooksCole, 2000.

[12] Vogel, A., Kerhervé, B., Bochmann, G. v., & Gecsei, J. (1995). Quality of Service Management: a survey. *IEEE Journal of Multimedia Systems*, 2(2):10-19, 1995 .