Published in B. Ganter, A. de Moor, editors, Contributions to ICCS 2003, Auxiliary Proceedings of the 11th International Conference on Conceptual Structure, pages 83–96, Dresden, Germany, July 2003, Shaker Verlag.

Model Operations for Quality-Driven Multimedia Delivery

Olivier Gerbé[1], Brigitte Kerhervé[2], and Uma Srinivasan[3]

[1] HEC - Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7
Olivier.Gerbe@hec.ca

[2] Université du Québec à Montréal
CP 8888, succursale centre ville
Montréal (Québec) Canada H3C 3P8
Kerherve.Brigitte@uqam.ca

[3] CSIRO/CMIS
Locked Bag 17
North Ryde NSW 1670, Australia
Srinivasan.Uma@csiro.au

Abstract. With the recent advances in distributed systems and wireless technology, users can access any information, from anywhere with any device. Multimedia delivery services are currently under development to operate in such environments. In this context, it appears essential to offer and support different levels of service according to users requirements and expectations and to work towards quality-driven delivery (QDD). Implementing QDD mechanisms leads us to consider different issues such as system components interoperability, quality information management, distributed execution of QDD activities and multi-criteria optimization. In this paper, we focus on quality information management to support QDD. We propose a model management approach to the problem and we introduce metamodel and model operations for that purpose. We use conceptual graphs formalism to develop our QDD metamodel and we show how the conceptual graph derivation mechanism can be applied to implement some fundamental model operations.

1 Introduction

In the last two decades, we have been faced with tremendous evolution of distributed multimedia systems in order to support emerging applications such as electronic commerce, health-care applications, digital publishing or infotainment. These applications integrate large amounts of voluminous data, located on several sites interconnected through various communication networks and potentially accessed by a large number of users. Such complex environments require the integration of system management mechanisms providing system scalability, application adaptation and quality of service (QoS) support [14]. Scalability refers to the capacity of the system to evolve according to the charge it faces.

Published in the Proceedings of the Fourth International Conference on Conceptual Structures (ICCS'96)

Using Conceptual Graphs for Methods Metamodeling

Olivier Gerbé, Benoit Guay, Mario Perron

DMR Group Inc.
1200 Mc Gill College Avenue, Montréal ,Québec, Canada H3B 4G7
e-mail: Olivier.Gerbe@dmr.ca, Benoit.Guay@dmr.ca, Mario.Perron@dmr.ca

Abstract. This paper presents a metamodel that has been designed to support the representation of any information technology method. Using the conceptual graph formalism, we have structured the knowledge in some modular way and introduced abstraction levels. Our contribution is a new approach that focuses first, on concepts of the application domain, then on information elements that document these concepts and, finally, on activities that produce information elements. We have developed a system including a conceptual graph engine to store, retrieve and display methods. This work has been done as part of the IT Macroscope Project at the Research and Development Department of DMR Group Inc.

1. Introduction

Information technology (IT) is among the most important factors of change in the 1990s and is now playing a leading role in effecting change. Mastering the evolution and management of IT requires methods, processes, software tools, training programs, as well as a systematic, comprehensive and consistent approach.

DMR Group Inc. has initiated the IT Macroscope project [4], a joint research project that aims to develop methods allowing organizations: i) to use IT to increase competitiveness and innovation in both the service and product sectors; ii) to organize and manage IT investments; iii) to implement information system solutions both practically and effectively; and iv) to ensure IT investments are profitable.

The methods developed during the IT Macroscope project are stored in the Methodological Repository. This repository is intended to fulfill the needs for designing and maintaining methods, designing training courses, and managing and promoting IT Macroscope products.

The Methodological Repository is faced with a problem of volume and complexity of knowledge to be managed. This requires to structure the knowledge in some modular way and to introduce abstraction levels. For that purpose, we propose a three layers approach: the Knowledge aspect is pure static data, the Presentation aspect supports the presentation of the knowledge structure, and the Behavioral aspect supports modeling of the dynamics of knowledge objects.

• *Knowledge Aspect*

Two ideas, *concept* and *symbol*, are introduced to properly differentiate the essence of the method from the way it is presented.

Concepts are the basic elements of the method; for example, documentation items, processes and system components. The concepts and relations between them make up the method itself. Symbols are texts, graphics, images, sounds and videos which name, define, explain or illustrate the concepts. The concepts and symbols are linked through description associations; they define the *knowledge aspect*.

• *Presentation Aspect*

The Presentation aspect defines how to display the knowledge. Displaying knowledge involves two steps: i) information retrieval; ii) formatting and displaying the information. For each step we have defined a complete set of concepts. This *presentation aspect* is detailed in [3].

• *Behavioral Aspect*

Behavioral aspect contains concepts that are used to represent the dynamics of the methods as well as the operations provided by repository tools.

This paper focuses on the *knowledge aspect* that supports and stores the methods resulting from the IT Macroscope project. We present the metamodel we have designed to store the IT Macroscope methods.

While designing this metamodel, we aimed at proposing a generic metamodel that would support any method formalization. That lead us to consider several knowledge representation formalisms. Among them we chose conceptual graphs [13] for their powerful formalism as well as for their mathematical foundations.

In this paper we present in detail the metamodel we have designed and implemented using conceptual graphs. This metamodel is currently implemented in the Methodological Repository of the first commercial version of the IT Macroscope.

This paper is organized as follows: Section 2 introduces the knowledge representation and notation used to model methods. Section 3 presents an overview of the components of a method. Sections 4 to 7 present further detail on the following components: Concepts, Information Elements, Activities and Participants. Section 8 presents applications of this work, followed by the conclusion and discussion of future work.

2. Meta-Metamodel and Notation

Developing the repository raised a problem: how to represent knowledge—or more precisely—methods and their promotional and training materials. Is there a single, homogeneous representation paradigm that can be used to describe the knowledge, presentation and behavioral aspects of our methods?

Classical IT modeling techniques use two fundamental notions to represent the real world: type and relationship. A type is used to represent things or objects that have same properties like attributes, relationships, behavior. Relationships represent association between things or objects but are established between types in the model. These IT techniques are well adapted in most cases but in some particular cases, it is very difficult to represent the real world. For example, let us consider the following two sentences:

- An employee is a person employed by an organization.
- An United Nations University (UNU) employee is employed by UNU.

The figure below presents the most natural way to represent these sentences. The relationship between Employee and UNU employee is a "is a" relationship. Any UNU employee is an employee. The relationship between UNU and Organization should be a "is an instance of" relationship but the "is employed by" relationship may not link a type and an instance.

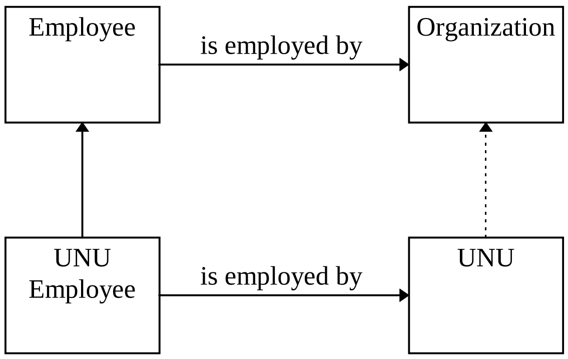

Figure 1 - Type versus instance

To solve those kinds of problems, we have chosen conceptual graphs formalism where relationships are always defined at instance level.

- *Conceptual Graphs*

First reminder for readers not familiar with conceptual graphs. 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; for example, each word of a glossary corresponds to a concept. The IT Macroscope manipulates several hundred of these concepts. A conceptual relation makes it possible to associate these concepts. Conceptual graphs were developed by John Sowa in the early 80s [13]. They are a system of logic based on the existential graphs of C.S. Peirce and semantic networks. Today, an international scientific community is working on conceptual graphs in fields as varied as natural language, artificial intelligence and corporate modeling. Our work takes place in the latter.

• *Meta-Metamodel and Notation*

In order to store conceptual structures, we have defined the meta-metamodel of the knowledge aspect. Concepts defining the meta-metamodel are also based on conceptual structures described in [13]. We made some modifications and extensions to adapt the formalism to the habits of IT professionals. We have first defined the notion of concept, conceptual relation and conceptual graph, as well as concept specialization like individual concept, generic concept and universal concept. Then we have defined the notion of concept type, relation type, and cardinalities to be able to describe the above notions. This section describes all these notions.

• *Concept*

A concept is a representation in mind of an object of the universe of discourse. A concept is the association of two referents: one refers to a type and the other refers to the object itself. For example, the mug on my desk can be seen in one context as a manufactured object and in another context as a piece of artwork. In that case there are two different concepts:

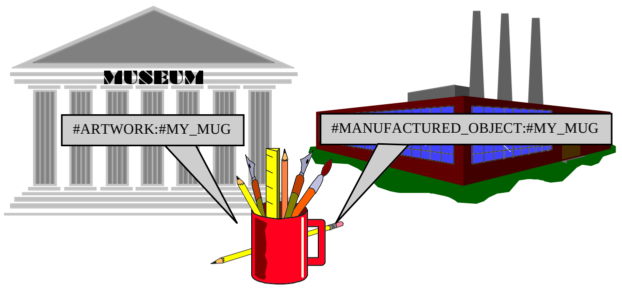

Figure 2 - My mug

`[#MANUFACTURED_OBJECT:#MY_MUG]` and `[#ART_WORK:#MY_MUG]` depending on the context.

• *Conceptual Relation*

A conceptual relation represents an association among an ordered set of one or two concepts and refers to the role played by each concept in the association.

• *Conceptual Graph*

A conceptual graph is a finite, connected, bipartite graph where the two kinds of nodes are concepts and conceptual relations, and where every conceptual relation has one or more arcs, each of which is linked to some concept.

• *Individual, Generic and Universal Concept*

An *individual concept* refers to a definite, i.e., an identified or determined, object. In a natural language like French or English, an individual concept corresponds to entities identified by a noun introduced by a definite article or identified by a proper noun. A *generic concept* represents an indefinite object (introduced by an indefinite article in natural language). It introduces the existential quantifier and corresponds to "There is a concept..." A *universal concept* represents any object of a given type. It introduces the universal quantifier and corresponds to "For all..."

• *Concept Type Definition Graph and Cardinalities*

A concept type refers to a set of concepts that have similar conceptual relations with other concepts. A concept type may be seen as a viewpoint. Different concept types allow us to look at things from different perspectives. An example in the context of methodology would be: the project manager interprets the notion of activity differently than the developer. One is interested in the duration of the activity; the other in how to perform it.

A concept type is defined by its position in the hierarchical tree of types and by a definition graph. In the concept type definition graph the universal concept refers to any concept instance of the concept type, and specifies which relations can be established between an instance of the type and other instances.

In order to describe in a more precise way conceptual relations that may be established between instances, we have introduced the notion of cardinality as in the entity-relationship formalism [1][2]. The use of cardinalities in concept type definition graph allows us to define constraints about conceptual relations between instances. This notion can be seen as a condensed form of type definition and schemata cluster as defined in [13]. Cardinalities in definition concept types allows the control of inputs. A conceptual relation may link two concepts if and only if this conceptual relation appears in at least one of the concept type definitions of the involved concepts.

We have extended the notation of conceptual graphs to support the paradigm of cardinalities. On the link from the universal concept, two figures specify respectively the minimum and the maximum number of times that any given conceptual relation may appear.

In the example below, the definition graph specifies that a domain concept has at least one text symbol, may have zero, one or more aliases or graphic symbols. and has one and only one definition.

```
[DOMAIN_CONCEPT:∀]-
  1,N(SYMB)<-[TEXT_SYMBOL:*]
  0,N(ALIAS)<-[TEXT_SYMBOL:*]
  0,N(SYMB)<-[GRAPHIC_SYMBOL:*]
  1,1(ATTR)<-[DEFINITION:*].
```

Figure 3 - Domain Concept Definition Graph

3. Method Metamodel

This section presents an overview of method components. A *method* is a set of interrelated domain concepts, activities, products or deliverables, participants and techniques employed by a discipline. The diagram below presents a schematic graph for an IT method.

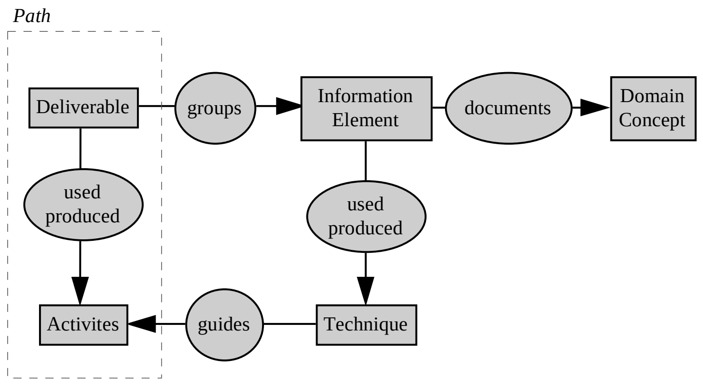

Figure 4 - Schematic Method Model

Domain concepts are the objects of interest of the domain or application area of the method about which information is to be provided in the process of applying a method. *Information elements* define a structure for gathering and presenting information about domain concepts. Information elements present vocabulary used and documentation produced when applying a method. *Activities* describe possible ways to use and produce deliverables. *Deliverables* assemble information elements for a particular purpose. *Techniques* define means to produce information elements.

Applying a method involves:

1. Determining which information elements will be produced;
2. Determining which techniques will be used to produce information elements;
3. Combining information elements into production and management deliverables;
4. Defining a process model to produce these deliverables; and
5. Executing this process model.

To help practitioners and project managers, method developers have defined paths. Paths are examples of steps 1 to 4 in the application of a method. A *path* is a set of activities, information elements grouped into deliverables, and resources.

In the next sections we focus on the four basic components: domain concepts, information elements, activities and participants (deliverables and resources).

4. Domain Concepts

Domain concepts are the objects of interest of the domain or application area of the method. They present vocabulary used in the field of the domain. Domain concepts are not necessarily defined in the context of a method; domain concepts may be part of a public ontology.

A domain concept is characterized by its attributes. Our repository is multilingual and a domain concept must have at least one text symbol which names it so that can be referred to in written text or diagrams. A domain concept may have aliases and graphic symbols to represent it. A domain concept must have one and only one definition. Following is the definition graph of a domain concept.

```
[DOMAIN_CONCEPT:∀]-
  1,N(SYMB)<-[TEXT_SYMBOL:*]
  0,N(ALIAS)<-[TEXT_SYMBOL:*]
  0,N(SYMB)<-[GRAPHIC_SYMBOL:*]
  1,1(ATTR)<-[DEFINITION:*]
```

Figure 5 - Domain Concept Definition Graph

5. Information Elements

Information elements structure information about domain concepts. They are the building blocks that will be produced and assembled into deliverables when applying a method. Information elements are categorized into information element types. This section gives definitions of information element, information element type and an example that illustrates these definitions.

- *Information element*

Domain concepts are defined regardless of the method; at the opposite side, information elements are defined in the context of a method. An ontology may contain hundreds or thousands of concepts, but a method may not be interested in all the concepts defined in the ontology. Information elements specify which concepts are relevant and are documented in the method.

The diagram below is the concept type definition graph of an information element. An information element documents one or more domain concepts. An information element may be produced using a technique and may be characterized by one or more states. Information elements may be composite, so an information element may be included in bigger information elements.

```
[INFORMATION_ELEMENT:∀]-
  1,N(DOCUMENTS)->[DOMAIN_CONCEPT:*]
  0,1(PRODUCED_BY)->[TECHNIQUE:*]
  0,N(CHRC)<-[STATE:*]
  0,N(MEMBER)->[INFORMATION_ELEMENT:*].
```

Figure 6 - Information Element Definition Graph

- *Information element type*

Information elements are structured into category based on the documented concept and their structure. These categories are called information element types. Attributes of information element types are: text symbol that name the type, a content description text that paraphrases the definition graph of the type and possibly purposes that explain the “raison d’être” of information elements. Associated to an information element type, examples may exemplify instances of the type. The figure below shows the information element type graph definition.

```
[INFORMATION_ELEMENT_TYPE:∀]-
  1,N(ATTR)<-[TEXT_SYMBOL:*]
  1,1(ATTR)<-[CONTENT_DESCRIPTION_TEXT:*]
  0,N(ATTR)<-[PURPOSE:*]
  0,N(EXAMPLIFIES)<-[EXAMPLE:*].
```

Figure 7 - Information Element Type Definition Graph

• *Example*

The figure below shows an example of information element. This example defines what a mission description is. The first graph defines the information element type mission description with an English label “description of the organization’s mission”, a French label “description de la mission de l’organisation”, a reference to a content description text that should describe what an instance of mission description is, and an example. The second graph is the definition graph of the type. It states that a mission description documents a mission, may be produced by a JAD technique, may have states and is a part of a P100 information element.

```
[INFORMATION_ELEMENT_TYPE:MISSION_DESCRIPTION]-
  (ATTR)<-[ENGLISH_TEXT_SYMBOL:’description of the organization’s mission’]
  (ATTR)<-[FRENCH_TEXT_SYMBOL:’description de la mission de l’organisation’]
  (ATTR)<-[CONTENT_DESCRIPTION_TEXT:#14]
  (EXAMPLIFIES)<-[EXAMPLE:#15].

[MISSION_DESCRIPTION:∀]-
  1,1(DOCUMENTS)->[MISSION:*]
  0,1(PRODUCED_BY)->[TECHNIQUE:#JAD]
  0,N(CHRC)<-[STATE:*]
  1,1(MEMBER)->[P100:*].
```

Figure 8 - Example

6. Activities

One of the greatest problems we had to solve during the definition of the metamodel of method was how to define process models. The importance of process models or procedures needs no illustration. The International Organization for Standardization has based ISO 9000 on procedures. The Software Engineering Institute has based the Capability Maturity Model on the possibility of reproducing and controlling processes.

We have defined an *activity* as a change, over a specific period in time, of specific inputs into specific outputs (possibly) by specific agents according to activity conditions.

A *process model* is a network of activities; activities may be performed in sequence or in parallel. A complex activity is described by a process model that defines its sub-activities. Activities may be shared by different complex activities. For example the same activity, Design User Interface, may be performed by a software developer and the future user of the application. Looking at the software developer process model and at the future user process model, these models show the same activity, Design User Interface, because they are both participants of the same activity.

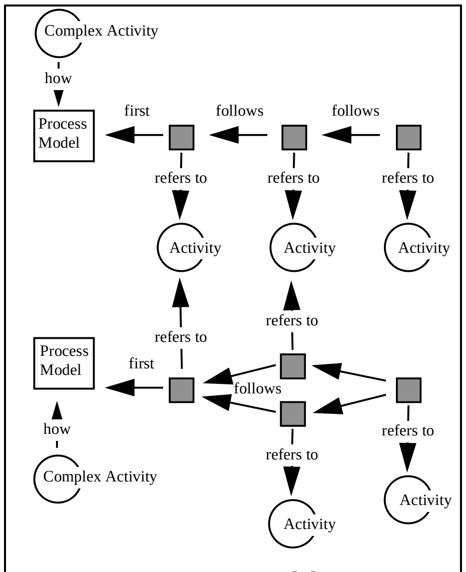

Figure 9 - Process Model Diagram

Figure 9 illustrates the decomposition of two complex activities sharing activities: One is decomposed into a sequential process model of three activities and the other is decomposed into a process model with parallel activities. Any sub-activity may itself be a complex activity.

The diagram below presents the type definitions of process model and event. A process model is a network of events that refer to the activities they sequence.

```
[PROCESS_MODEL:∀]-
  1,N(FIRST)<-[EVENT:*]
  1,1(HOW)->[ACTIVITY:*].

[EVENT:∀]-
  0,N(FOLLOWS)->[EVENT:*]
  1,1(REFERS)->[ACTIVITY:*].
```

Figure 10 - Process Model & Event Definition Graph

In the process of developing a method, the documentation of activities is the key issue. Activities are what people have to do and they have to be described in detail. Following is the list of all the attributes and other components involved in the description of an activity.

An activity is documented by its purposes, objectives, and possibly a descriptive text. An activity has been defined as changes of inputs into outputs by resources according to activity conditions. An activity must have at least one output, one resource, and possibly inputs used to produce outputs. Each group of these participants may be described by a text (input text, output text, resource text) that introduces or explains the role of the group in the activity.

Activities are sequenced by defining process models. The sequence is also fully defined by the preconditions and postconditions of the activities [12]. The *preconditions* and *postconditions* are propositions that specify the conditions that must be true before and after the performance of the activity.

Activities may be linked to rules that are prescribed guidelines for action. The performance of some activities may require skills or know-how. This is indicated by links to techniques. The acquaintance of inputs, outputs, resources and techniques may not be sufficient to perform the activity efficiently. Depending on the types of resources, guidelines may be documented to help in the performance of the activity.

```
[ACTIVITY:∀]-
  1,N(SYMB)<-[TEXT_SYMBOL:*]
  "---------------------------------------------"
  0,N(ATTR)<-[PURPOSE:*]
  0,N(ATTR)<-[OBJECTIVE:*]
  0,1(ATTR)<-[DESCRIPTION_TEXT:*]
  "---Inputs------------------------------------"
  0,1(ATTR)<-[INPUT_TEXT:*]
  0,N(USED_BY)<-[PARTICIPANT:*]
  "---Outputs-----------------------------------"
  0,1(ATTR)<-[OUTPUT_TEXT:*]
  1,N(CHANGED_BY)<-[PARTICIPANT:*]
  "---Resources---------------------------------"
  0,1(ATTR)<-[RESOURCE_TEXT:*]
  1,N(ENABLES)<-[PARTICIPANT:*]
  "---Process Model-----------------------------"
  1,N(END)<-[EVENT:*]
  0,1(HOW)<-[PROCESS_MODEL:*]
  "---Pre & post Conditions---------------------"
  1,N(DEPENDS_ON)->[PRECONDITION:*]
  1,N(REALIZES)->[POSTCONDITION:*]
  "---Rules-------------------------------------"
  0,N(GOVERNS)<-[RULE:*]
  "---Guiding Techniques------------------------"
  0,N(GUIDES)<-[TECHNIQUE:*]
  "---Guidelines--------------------------------"
  0,1(ATTR)<-[PRACTITIONER_TEXT:*]
  0,1(ATTR)<-[CLIENT_TEXT:*]
  0,1(ATTR)<-[MANAGER_TEXT:*].
```

Figure 11 - Activity Definition Graph

7. Participants

The above section distinguishes three types of participants in activities: inputs that are used by activities, outputs that are produced by activities and resources that enable activities. This section details each of these types of participants.

• *Inputs*

An *input* is a participant that is used by an activity. An input must exist before the beginning of the activity and is transformed or consumed during the activity.

If the input is *consumed*, it disappears during the activity and no longer exists. If the input is *transformed*, this means that there is a state transition during the activity and the input no longer exists in its original state. In both cases, inputs in their orignal state no longer exist at the end of the activity.

Following is the definition graph of an input to an activity. The link to an activity is unique and mandatory.

```
[INPUT:∀]-
  1,1(USED_BY)->[ACTIVITY:*].
```

Figure 12 - Input Definition Graph

• *Outputs*

An *output* is a participant that is produced by an activity. An output may be created or modified by the activity.

If the output is *created*, it may be created from scratch or from a consumed input. If the output is *modified*, this means that it is the result of a modification applied to a transformed input

Following is the definition graph of an output of an activity. The link from output to activity is unique and mandatory.

```
[OUTPUT:∀]-
  1,1(PRODUCED_BY)->[ACTIVITY:*].
```

Figure 13 - Output Definition Graph

• *Resources*

Resources are participants in activity which are not inputs or outputs. A resource is neither consumed nor transformed by the activity. A resource is, at the end of the activity, in the same state it was at the beginning of the activity.

Following is the definition graph of a resource of an activity. A resource must have at least one *enables* conceptual relation with an activity. In the context of an activity, a resource may assume a responsibility.

```
[RESOURCE:∀-
  1,N(ENABLES)->[ACTIVITY:*]
  0,1(ASSUMES)->[RESPONSABILITY:*].
```

Figure 14 - Resource Definition Graph

• *Relationtype Hierarchy*

This section presents the relation types hierarchy we have defined to describe participation in activities. Indentation reflects subtyping.

Involved

An involved relationship is a relationship between a concept and an activity.

Enables

An enable relationship is an involved relationship for which:
- each involved object exists before the activity starts, and
- those involved objects are not changed.

Practices

A practices relationship is an enables relationship for which the involved object does the activity.

Informs

An informs relationship is an enables relationship for which the involved object feeds the activity with information .

Controls

A controls relationship is an enables relationship for which the involved object guides, manages, or controls the activity.

Supports

A performs relationship is an enables relationship for which the involved object supports the realization of the activity.

Used

A used relationship is an involved relationship for which the involved objects are either consumed during the activity, or the state they are at the start will change.

Consumed

A consumed relationship is an used relationship for which the involved object is consumed or deleted during the realization of the activity.

Transformed

A transformed relationship is an used relationship for which the involved object is transformed in its state during the realization of the activity.

Changed

A changed relationship is an involved relationship for which each involved object is either created , or its state has been changed during the activity.

Created

A created relationship is an involved relationship for which the involved object is created during the activity.

Modified

A modified relationship is an involved relationship for which the involved object has its state changed during the activity.

8. Conclusion

In this paper we have presented the metamodel of a method based on domain concepts. This work has been developed and implemented at the Research & Development Department of DMR Group Inc. This work is part of the IT Macroscope Repository that stores and displays the methods developed in the context of the IT Macroscope project.

In January 1996, five methods were commercially delivered: Information systems development, Architecture, Benefits, Technical Infrastructure, and Estimating in hypertext formats generated from conceptual graphs. From about 80,000 conceptual graphs, we generated more than 100,000 HTML pages that can be browsed using commercial Web browsers.

We developed a CG knowledge base to store the knowledge aspect (with its three levels of abstraction: the meta-metamodel, the metamodel of method, the methods themselves) and the presentation aspect. We also developed an engine that interprets presentation conceptual graphs and generates any kind of output.

This development lead us to manage a large amount of knowledge:

- 1,000 concept types for meta-metamodels of knowledge, presentation and operational aspects;
- 100 view types and 2,000 concept types that define how to extract information from the conceptual graph repository and how to generate HTML files;
- 5,000 domain concepts, 1,500 information elements, 20,000 deliverables, 5,000 activities, 200 techniques;
- 100,000 English and French HTML files, 2,000 generated GIF models and more than 1,000,000 hyperlinks;

This new approach will allow us to define methods that can be automated. We have extended the conceptual graph notation but these extensions have their equivalencies in conceptual graph formalism [10]. The next step is the complete integration of the three aspects: knowledge, presentation and behavior.

9. References

[1] Chen, P. (1976) *The Entity-Relationship Model - Toward a Unified View of Data*, in ACM Transactions on Database Systems, Vol.1 No.1, pp. 9-36.

[2] Creasy, P. & Ellis, G. (1993) *A Conceptual Graphs Approach to Conceptual Schema Integration*, in Mineau, G., Moulin, B., Sowa, J., Conceptual Graphs for Knowledge Representation, Springler-Verlag.

[3] Gerbé, O. & Perron, M. (1995). *Presentation Definition Language using Conceptual Graphs*, in Proceedings of the Peirce Workshop, Santa Cruz

[4] Groupe DMR inc. (1990) *Projet mobilisateur Le Macroscope*, Montréal.

[5] Jensen, K. (1995). *Coloured Petri Nets*, vol.1, Springler-Verlag, Berlin.

[6] Heym, M. & Österle, H. (1992). *A Reference Model for Information Systems Development*, in Kendall, K.E et al. (Eds), The Impact of Computer Supported Technologies on Information Systems Development, North Holland, Amsterdam.

[7] ISO (1993). *ISO 9000 International Standards for Quality Management*, Genéve.

[8] Mineau, G. W. & Godin, R. (1992). Automatic Knowledge Structuring for Browsing Retrieval, in Y. Yesha (Ed.), Proceedings of the 1st Int. Conf. on Information and Knowledge Management (CIKM-92), Baltimore, USA: Int. Society of Mini and Microcomputers (ISMM), pp. 273-281.

[9] Mineau, G. W. (1992). *Normalizing Conceptual Graphs*. In T. Nagle,J. Nagle,L. Gerholz, & P. Eklund (Eds.), Conceptual Structures: current research and practice, (pp. 339-348). Ellis Horwoo

[10] Mineau, G. W. (1994a). *Étude sur la modélisation conceptuelle du référentiel méthodologique à l'aide du formalisme des graphes conceptuels.* Groupe DMR Inc.

[11] Mineau, G. W. (1994b). *View, Mappings and Functions: Essential Definitions to the Conceptual Graph Theory,* in W. M. Tepfenhart,J. P. Dick, & J. F. Sowa (Eds.), Conceptual Structures: Current Practices, (pp. 160-174). Springer-Verlag.

[12] Petri, C.A. (1962) Kommunikation mit Automaten, Ph.D. dissertation, University of Bonny.

[13] Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley.

[14] Sowa, J. & Zachman, A. (1992) *Extending and formalizing the Framework for Information Systems Architecture,* in IBM Systems Journal Vol.31, No.3, pp.590-61

Application adaptation refers to the ability of the application to change its behavior according to the changes occurring in the processing environment. QoS is a more general concept referring to the capacity of the system to offer and support different levels of service according to users requirements and expectations. Scalability and application adaptation can be considered as some of the possible mechanisms used to provide QoS support.

QoS support was initially introduced in the field of telecommunication networks and multimedia systems [13] and led to proposals for management strategies aimed at deciding whether and controlling how multimedia streams can be delivered to the user with some constraints. These constraints are expressed during a specification step where the user specifies his requirements, generally concerning system performance relative to media delivery and synchronization. The system then works to deliver the specified level of service and for that purpose transforms the users' requirements into various constraints mainly targeted to the transport system [7].

If we consider the notion of QoS from a broader perspective, we have to position the user at the center of the process and allow him to express non-functional requirements. We can then talk about quality-driven delivery (QDD) where the user's quality requirements are taken into account in the different steps of the delivery. With this perspective in mind, implementing QDD mechanisms leads us to consider different issues such as system components interoperability, quality information management, distributed execution of QDD activities, and multi-criteria optimization. In this paper, we focus on quality information management to support QDD. We propose a model management approach to the problem and we introduce metamodel and model operations for that purpose. We use conceptual graphs formalism to develop our QDD metamodel and we show how the conceptual graph derivation mechanism can be applied to implement some fundamental model operations. The rest of the paper is organized as follows. Section 2 presents the principles of QDD and motivates the use of an approach based on model management. We present our modeling formalism and our architecture in Section 3. In section 4 we propose and explain our metamodel. Section 5 presents some operations we need to process on models in order to support QDD. Section 6 concludes and presents some future work.

2 Quality-driven Delivery

Quality-driven delivery refers to the capacity of services to deliver objects, while considering the users expectations in terms of non-functional requirements. For example, an adaptive video delivery service must consider user expectations in terms of the perceived quality of video sequences to be delivered, as well as the characteristics of the equipment used for delivery (cellular phone, PDA or other). In this case, the video delivery service has to choose among possible variants of the video sequence, the one satisfying the user expectations and the equipment constraints.

Some approaches have been proposed for multimedia application adaptation, more specifically for adaptation to the technical infrastructure used for accessing multimedia objects [8][6][15]. Most of them are more oriented towards resource allocation than user-perceived quality. We believe that it is time to consider maximizing the user-perceived quality as a main objective.

These considerations may be placed in the more general context of the Web Accessibility Initiative (WAI) initiated by the World Wide Web Consortium. The accessibility to the World Wide Web is important and there are barriers on the Web for many types of disabilities. A QDD approach based on the user requirements considers all personal (physical, cognitive and affective) parameters.

In this section we first introduce the general principles and activities involved in a QDD process and we illustrate them with the help of a simple adaptive video delivery application. We then motivate the use of a model management based approach.

2.1 Principles and Activities

QDD can be viewed as a generalization of QoS management, and some of the traditional QoS activities can be transposed for QDD. More specifically, when studying quality information management, we are mainly interested in the three following QoS activities: specification, monitoring and mapping.

Specification consists in identifying the quality dimensions to express user's requirements such as time, cost or data quality and of defining the expected level of quality. Monitoring consists in collecting information on the quality level that can be provided by the different components of the distributed multimedia system, such as the video server, the communication server, the database server or the client device. Mapping consists in converting qualitative and subjective quality levels into quantitative and measurable quality levels, as well as to convert these quantitative quality levels to constraints corresponding to resource requirements for the object delivery. A QDD system then requires the description and management of this quality information. Based on this information, the system takes decisions that are transmitted to the different components supporting QDD.

To illustrate the principles of QDD, we take the example of a simple adaptive video delivery service where the users specify their quality preference according to three dimensions: the language of the audio sequence and the size and the frame rate of the video. The adaptation process leads to take decisions in order to deliver a variant of the initial high quality video sequence compatible with the available resources.

2.2 Quality Information Management

The focus of the work presented in this paper is the management of quality information or quality metadata related to the user requirements, the objects to be delivered and the resources used for delivery. Quality information comes from different sources and can be heterogeneous. For example, quality information associated to video objects can differ depending on the encoding format and the standard used to describe associated metadata. The monitoring tools used to collect quality information about the service level of the system components can also produce heterogeneous information. We see that there is a need for homogenization and integration of quality information. Different factors can influence user-perceived quality, and any metadata associated to multimedia object should be considered as a potential candidate for being a quality factor. Thus, there is a need for extension and adaptation of quality information models as well as for tools allowing description, integration and translation of quality information sets coming from different sources and represented using different formalisms or standards.

This problem is similar to the problem of data migration or schema translation in the field of metadata management for data warehouses and web portals. For QDD, we are interested in a subset of metadata, metadata describing the quality of objects, data sources or resources. To solve the problem of data migration and schema translation, database researchers have recently proposed an approach based on model management [1] [2]. These authors propose to address the problem from a higher level of abstraction and to work on models rather than working on data. This approach would lead to the development of a generic infrastructure for managing models and to the introduction of model operations for integration and translation of data.

At the same time, in the field of software engineering and more specifically for software production, the Object Management Group (OMG) recently launched the Model Driven Architecture (MDA) to move from code-oriented software production techniques to model-oriented production techniques [10][3]. The objective is to allow abstraction, refinement and different viewpoints of models representing the function, structure or behavior of a system. The other important objective is to be able to design models independent of platform and implementation environments. The concept of platform-independent model (PIM) and platform-specific model (PSM) have been introduced for that purpose.

We believe that quality information management for QDD is a good candidate for model management, because not only are we concerned by integration and translation of quality metadata but also because QDD is provided in a distributed and heterogeneous environment where monitoring tools are fully platform-dependant.

3 Modeling Formalism

When looking at modeling techniques to deal with model management in the context of QDD, we have chosen conceptual graphs formalism for two reasons: (i) type definitions are made at instance levels, and (ii) there exists a very powerful mechanism called conceptual graph derivation, that can be used for fundamental model operations.

In this section, we first introduce our overall modeling architecture and identify where our work takes place, and then we give a brief introduction to conceptual graphs for those who are not familiar with this formalism.

3.1 Modeling Levels

Our overall modeling architecture is a four layers architecture, based on the notions of model, metamodel and meta-metamodel.

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 a situation that takes place in the real world. The way and the vocabulary we use to build models are called the metamodel. The metamodel is a precise definition of the constructs and rules that are used in models. At the highest level, the meta-metamodel defines the language used in metamodels. In this paper we use the conceptual graph formalism to represent these models and their relationships.

Conceptual graphs are graphs made of concepts (a box with a type label and a referent) and conceptual relations (a circle with a type label). The type label identifies the type of the referent or of the relationship. Conceptual graphs will be presented in more details in the next section.

Figure 1 illustrates the four layers architecture where:

– M0 is the real world where the situation we want to represent takes place. The M0 level is described at level M1.
– M1 is the model level. It represents a particular situation of the real world. It defines types and instances that represent the real world, M0. Models at level M1 are expressed using a language that is defined at level M2.
– M2 is the metamodel level: our meta-model for QDD. It contains all the vocabulary used at level M1.
– M3 is the meta-metamodel: the formalism of conceptual graphs. We show only constructs of the formalism that are involved in the example. A complete description can be found in [5]

In Figure 1, at level M0 Mary is specifying her requirements. She wants her video in French. This is represented at level M1 by Req. This requirement contains a qualitative dimension Language which is characterized by the value 'French'. At level M2 we find the vocabulary needed to describe M1. In the example, for clarity purpose we show only concept types QualitativeRequirement, QualitativeDim, Value and relation type chrc (characterizes). At level M3 is the meta-metamodel with Concept Type and Relation Type. The meta-metamodel is defined using itself and thus is the highest level.

The conceptual relation meta links two constructs from two adjacent levels or two constructs from level M3. In Figure 1 conceptual relations meta has been added to better understand the different modeling levels and their relationships. They are implicitly defined in the type of concepts.

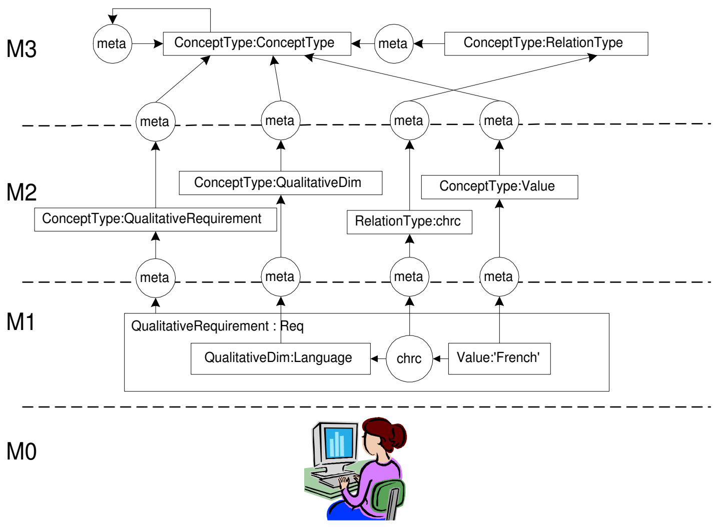

Fig. 1. Four Layers Architecture.

3.2 Conceptual Graphs

This section presents a brief introduction to the formalism of conceptual graphs introduced by John Sowa in 1984 [11]. Only a minimum explanation is provided as required to understand the rest of the paper. More information on conceptual graphs can be found [12] and [4].

Conceptual graphs are a formalism whereby the universe of discourse is modeled by concepts and conceptual relations. 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 any instance of that concept type must comply with. Figure 2 presents the definition graph of EMPLOYEE that means that an employee is a person that works for some organization.

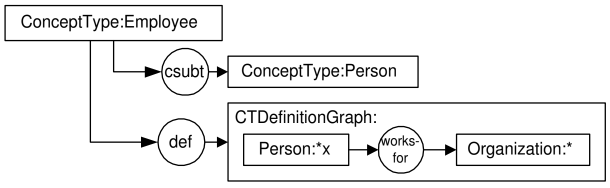

Fig. 2. Conceptual Graphs: Type Definition.

We now present two theorems from [11] that specify how conceptual graphs can be structured.

Theorem 1. *Generalization, denoted* $\leq$, *defines a partial ordering of conceptual graphs, called the generalization hierarchy. For any conceptual graphs* u, v *and* w *three conceptual graphs, the following properties are true:*

- Reflexive. *$u \leq u$;*
- Transitive. *If $u \leq v$ and $v \leq w$, then $u \leq w$;*
- Antisymmetric. *If $u \leq v$ and $v \leq u$, then $u = v$;*
- Subgraph. *If v is a subgraph of u then $u \leq v$;*
- Subtypes. *If u is identical to v except that one or more type labels of v are restricted to subtypes in u, then $u \leq v$;*
- Individuals. *If u is identical to v except that one or more generic concepts of v are restricted to individual concepts of the same type, then $u \leq v$.*

Last we present the fundamental operation on conceptual graph that calculate the specialization relationship.

Theorem 2. *For any conceptual graph u and v where $u \leq v$, there exists a mapping $\pi : v \rightarrow u$, where πv is a subgraph of u called a projection of v in u. The projection operator π has the following properties :*

- *For each concept c in v, πc is a concept in πv and $type(\pi c) \leq type(c)$. If c is individual, then $referent(c) = referent(\pi c)$.*
- *For each conceptual relation r in v, πr is a conceptual relation in πv and $type \pi r = type(r)$. If the ith arc of r is linked to a concept c in v, the ith arc of πr must be linked to πc in πv.*

We will use these theorems for model management operations. The reader interested in their demonstrations will find them in [11].

We can now understand why conceptual graphs are well adapted for model management. Types definitions are made at instance levels that, means that types are defined by graphs made of instances and part of these instances may be generic or individuals. Types and instances are represented with the same elements: concepts and conceptual relations. Therefore we define models (types) and requirements (instances) as graphs and we use generalization and derivation operations to manipulate them. Using Object oriented technology like UML is more problematic because classes (types) and objects (objects) are different in nature and cannot be mixed or manipulated by the same tools.

4 Metamodel for QDD

In this section, we present the metamodel we propose for QDD. This metamodel specifies the vocabulary and grammar we will use to describe quality information for users, media objects and system components.

4.1 Dimension

Quality information is built with the concept of dimension. Dimensions are used to describe objective or subjective characteristics relative to the quality of a delivery service or the quality of an object to be delivered. Subjective characteristics refer to the quality level perceived by the user while objective characteristics refer to a measurable quality level. An example of a dimension is network-throughput. This dimension is objective and can be measured using monitoring tools for communication networks. We call such a dimension a quantitative dimension. An example of a subjective dimension can be response-time with the values: (unacceptable, bad, good, excellent). This dimension a qualitative dimension since the possible values depends on the perception or the interpretation of the user. Figure 3 presents the concept type definition graph for Dimension. A Dimension is defined on a domain of possible values.

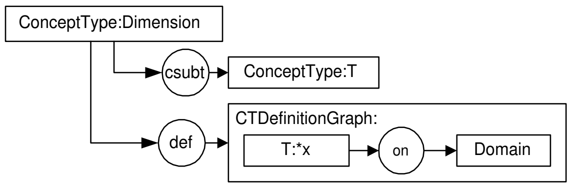

Fig. 3. QDD Metamodel : Dimension element.

We define two types of dimensions: qualitative dimensions and quantitative dimensions.

4.2 Quality Information Models

The quality information, built with the concept of dimension, is modeled in quality information (QI) models. QI models describe the structure of quality information and allow the reuse, transformation and extension of existing models. A Model is represented by a graph that contains Dimension elements. Figure 4 shows the concept type definition graph for Model.

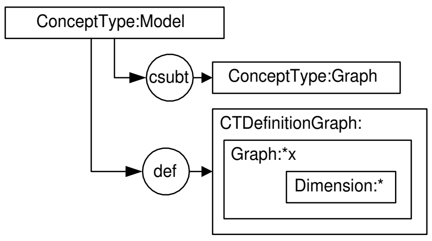

Fig. 4. QDD Metamodel : Model element.

QI models can be User Quality Model or Actor Quality Model. The model elements of a User Quality Model describe the dimensions used to specify the expected quality level. We make a distinction between Qualitative Quality Model where the dimensions included in the model are qualitative dimensions, and Quantitative Quality Model where the dimensions are quantitative dimensions. Figure 5 presents the type hierarchy for Model.

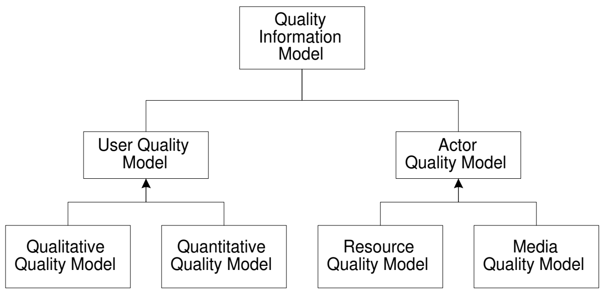

Fig. 5. Quality Information Models.

The model elements of an Actor Quality Model integrate the quantitative quality dimensions along which is described a quality level. We make a distinction between a Media Quality Model built with the dimensions used to describe the quality level of an object to be delivered, and a Resource Quality Model describing the quality level offered by a system component (communication network, database system, video server, user's device etc.).

We give in Figure 6 an example of a Quantitative Quality model. The model has been simplified in order to understand the relationship between the metamodel at level M2 and models at level M1. The VideoDeliveryModel is a graph that contains three dimensions Language, Frame Rate and Size.

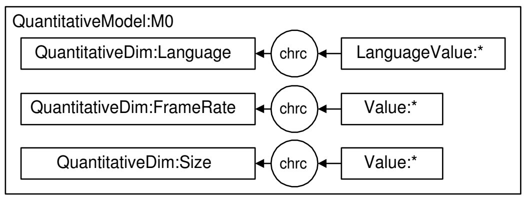

Fig. 6. QDD Models : an example.

4.3 Derived Models

In our approach, we consider that there exists a predefined Quality Information Model: the Core Model. The Core Model is unique and contains the predefined set of dimensions relevant for all types of QDD services. From this Core Model, we can derive other models. The Core Model can be built on the basis on existing standard such MPEG-7[6]

Derived models are models that are built from other models. A derived model is a graph that is a generalization of another model. Explanation of derivation mechanism will be detailled in Section 5

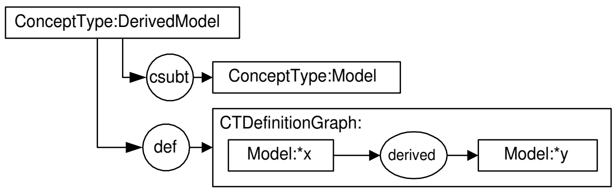

Fig. 7. QDD Metamodel : Derived Model element.

4.4 Instances of Models

From Quality Information Models we instantiate Quality Information. We distinguish : Qualitative Requirement, Quantitative Requirement, and Quality Level. From a Qualitative Quality Model we instantiate a Qualitative Requirement which is the user's specification of the expected quality level. From an Actor Quality Model we instantiate a Quality Level for a given actor (media or resource) and from a Quantitative Quality Model we instantiate a Quantitative Requirement

In Figure 8 is the concept type definition of Quantitative Requirement. Quantitative Requirements are built from Quantitative Quality models. A quantitative requirement is a graph that has a relationship instOf with a quantitative quality model. Explanation of instantiation mechanism will be detailed in Section 5

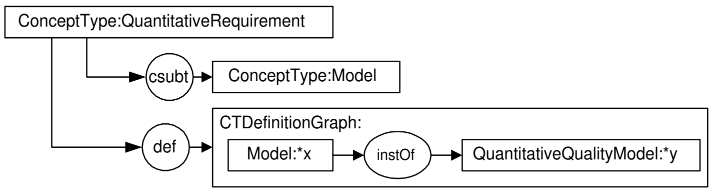
ConceptType:QuantitativeRequirement
csubt
ConceptType:Model
def
CTDefinitionGraph:
Model:*x
instOf
QuantitativeQualityModel:*y

Fig. 8. QDD Metamodel : Quantitative Requirement element.

5 Model Operations

While considering model management for QDD, we have to consider the different operations to be performed on models during the different steps of a QDD process. The first operation occurring is the definition of the different QI models supporting the process. In order to avoid repetitive tasks in model definition and creation and to favor reuse of predefined or existing models, we introduce the derivation operation. The second operation we consider in this paper is the model instantiation, where a container for quality information is created in associating values or constraints to the dimensions that constitute the QI model.

Another important operation is the model transformation, where semantic or implementation rules are expressed to transform instances of a source model to instances of a target model. The transformation operation is not discussed in this paper and is part of our future work.

In this section, we focus on model derivation and model instantiation and we show how the conceptual graph derivation mechanism can be applied for these two operations on QI models. We illustrate these operations with a given model.

5.1 A Quality Information Model

Let $\mathcal{M}0$ be a Quantitative Quality model, Figure 9 presents $\mathcal{M}0$, an example of generic quality model. The graph $\mathcal{M}0$ groups three quantitative dimensions describing the quality of a video Language, FrameRate and Size. Each dimension is characterized by a Value. In the case of Language, a specialized value LanguageValue has been defined as a subtype of Value.

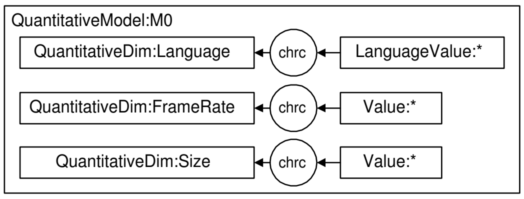

Fig. 9. A Quantitative Quality Model for video delivery..

5.2 Model Derivation

In our approach, derivation is defined as a model specialization. A generic model is specialized into a more specific model where each element of the specialized model is a specialization of an element of the more generic model. In object oriented modeling and in object oriented programming, specialization is well defined [9]. The specialization is implemented through the inheritance mechanism in UML which is based on segment descriptors and full descriptors. A full descriptor is the specification of characteristics of instances. A full descriptor is produced from a set of segment descriptors connected by generalization relationships. Segment descriptors are the elements defined in UML models.

Using conceptual graphs, the model derivation mechanism we defined for QI model management consists in a generalization. From a generic model, a generalization is done by the user selecting the dimensions which he or she is interested in. In this step, the user suppresses the dimensions that are not relevant for him. The result is a subgraph of the original one so the result model is a generalization of the generic model.

From a generic model, $\mathcal{M}0$ we want to derive two QI models for two specific video delivery applications where the users are only concerned by the language quality dimension in the first application and by Frame-rate and Size in the second one. The selection of the pertinent dimensions for the application is made.

The first one is only concerned by the dimension Language. The resulting graph $\mathcal{M}1$ is presented in Figure 10. It is reduced to one dimension. The second one is only concerned by the dimensions FrameRate and Size. The resulting graph $\mathcal{M}2a$ is presented in Figure 10. It is reduced to two dimensions.

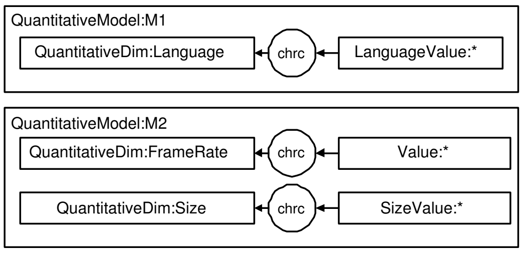

Fig. 10. Model Derivation : example 1 and 2.

In both cases we have a subgraph of the original graph $\mathcal{M}0$. According to Theorem 1 (Section 3.2) $\mathcal{M}1$ and $\mathcal{M}2$ are subgraphs of $\mathcal{M}0$ then we have $\mathcal{M}0 \leq \mathcal{M}1$ and $\mathcal{M}0 \leq \mathcal{M}2$. In the context of conceptual graphs, we use generalization to define the model derivation operation as follows :

Definition 1. *A model $\mathcal{M}x$ is said a* derived model *from the model $\mathcal{M}y$ if there exists a projection $\pi : \mathcal{M}x \rightarrow \mathcal{M}y$. There are the following properties :*

– *$\pi(\mathcal{M}x)$ is a subgraph of $\mathcal{M}y$.*
– *$\mathcal{M}y \leq \mathcal{M}x$.*

5.3 Model Instantiation

Model instantiation corresponds to the creation of a container for Quality Information and the creation of expressions (values or constraints) on the dimensions that are part of a given Quality Information Model. Instantiation of a Quality Information Model produces a Quality Information. Instantiation of a Qualitative (Quantitative) Quality Model produces a Qualitative (Quantitative) Requirement, corresponding to a constraint specifying the quality requirements and expectations for a given user. Instantiation of an Actor Quality Model produces a Quality Level corresponding to the description of the quality level of a system component or of an object to be delivered.

This operation corresponds to the conceptual graph specialization. In the example 1 the requirement is on the language of the video. The language must be French. The concept that represents the value of the language is replaced by an individual concept that refers to the language French. Figure 11 presents the resulting graph $\mathcal{R}eq1$ which is an instance of model $\mathcal{M}1$.

In the second example we assume a new type has been defined. This new type SizeValue is a subtype of Value. Its domain is for example 320x240, 640x480, 720x480. In order to restrict the possible values of size to this set of values we

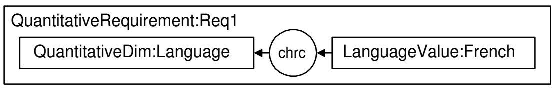

Fig. 11. Model Instantiation : example 1.

replaced Value by sizeValue. The concept remains generic, so it is one of the three possible values. As in the first example, the user wants a specific value for the frame rate, 30 images per second in this case. The generic concept `[Value:*]` is replaced by the individual concept `[Value:30]` . Figure 12 presents the resulting graph $\mathcal{R}eq2$ which is an instance of model $\mathcal{M}2$.

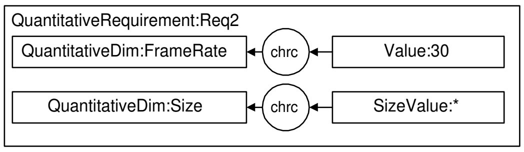

Fig. 12. Model Instantiation : example 2.

In both cases, we use the model instantiation operation which corresponds to the specialization mechanism of conceptual graphs. According to Theorem 1 (Section 3.2) every concept of $\mathcal{R}eq1$ and $\mathcal{R}eq2$ have the same type or a subtype and the referent is the same or a generic referent has been replaced by an individual one in respectively $\mathcal{M}1$ and $\mathcal{M}2$ then we have $\mathcal{R}eq1 \leq \mathcal{M}1$ and $\mathcal{R}eq2 \leq \mathcal{M}2$. The instantiation operation is a specialization in the context of conceptual graphs and we can define it as follows.

Definition 2. *A Quality Information $\mathcal{QI}x$ is an* instantiation *of a model $\mathcal{M}x$ if there exists a projection $\pi : \mathcal{M}x \rightarrow \mathcal{QI}x$.*

6 Conclusion and Future Work

In this paper we have presented a metamodel for quality-driven delivery using conceptual graphs as a modeling formalism. With this metamodel, we have specified the vocabulary and grammar we use to describe quality information for users, media objects and system components.

We have also showed how two model management operations: derivation and instantiation may be implemented in a very simple way through the derivation mechanism of the conceptual graphs. Using the conceptual graphs formalism helped us to clarify the concepts of our metamodel and the corresponding operations. Conceptual graphs appear to be a neutral and powerful modeling formalism for defining model management mechanisms.

In the future, we will work on another important operation for QDD: the mapping operation. Mapping allows the expression of semantic relationships between the concepts of different quality information models. These semantic relationships are defined on the quality dimensions and will be used to transform instances of a source model to instances of a target model. We believe that conceptual graphs with its inference capabilities and agents are certainly a good candidate to formally define mapping operations.

References

[1] P. Bernstein, A. Halevy and R. Pottinger. A Vision for Management of Complex Models. *ACM SIGMOD Record*, 29(4), pages 55–63, 2000.

[2] P. Bernstein, L. Haas, M. Jarke, E. Rahm and G. Wiederhold. Panel : Is Generic Metadata Management Feasible?. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB2000)*, pages 660–662, Cairo, Egypt, September 10-14 2000. Morgan Kaufmann.

[3] J. Bézivin and O. Gerbé. Towards a Precise Definition of the OMG/MDA Framework. In *Proceedings of the 16th Conference on Automated Software Engineering*, pages 273–280, San Diego, USA, November 2001. IEEE Computer Society Press.

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

[5] O. Gerbé, G. Mineau and R. Keller. Conceptual Graphs, Metamodeling and Notation of Concepts. In *Proceedings of the 8th International Conference on Conceptual Structures (ICCS'2000)*, Lecture Notes in Artificial Intelligence, volume 1867, pages 157–171, Darmstadt, Germany, August 2000. Springer-Verlag.

[6] ISO/IEC JTC1/SC29/WG11. MPEG-7 Overview. July 2001. available at http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm.

[7] C. Koliver, K. Nahrstedt, J-M. Farines, J. da Silva Fraga and S. Sandri. Specification, Mapping and Control for QoS Adaptation. *Real-Time Systems*, 23(1-2), pages 143–174, 2002.

[8] R. Mohan, J. Smith and C-S. Li. Adapting Multimedia Internet Content for Universal Access. *IEEE Transactions on Multimedia*, 1(1), pages 104–114, March 1999.

[9] Object Management Group. OMG Unified Modeling Language Specification. September 2001.

[10] Object Management Group. Model Driven Architecture. July 2001. Document number ormsc/2001-07-01.

[11] J. Sowa. Conceptual Structures: Information Processing in Mind and Machine. 1984. Addison-Wesley.

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

[13] A. Vogel, B. Kerhervé, G. von Bochmann and J. Gecsei. Quality of Service Management : a Survey. *IEEE Journal of Multimedia Systems*, 2(2), pages 10–19, 1995.

[14] G. von Bochmann, B. Kerhervé, H. Lutfiyya, M. Salem and H. Ye. Introducing QoS to Electronic Commerce Applications. In *Proceedings of the 2nd International Symposium on Electronic Commerce (ISEC2001)*, Lecture Notes in Computer Science, volume 2040, pages 138–147, Hong-Kong, China, April 26-28 2001. Springer-Verlag.

[15] H. Zhang. Adaptive Content Delivery: a New Research Area in Media Computing. In *Proceedings of the International Workshop on Multimedia Data Storage, retrieval, Integration and Applications*, Hong-Kong, China, January 13-15 2000.