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.

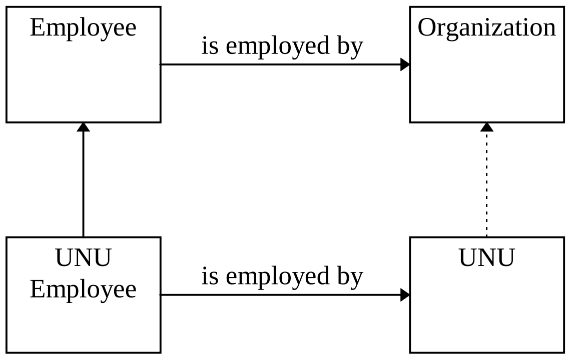

Figure 1 - Type versus instance

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

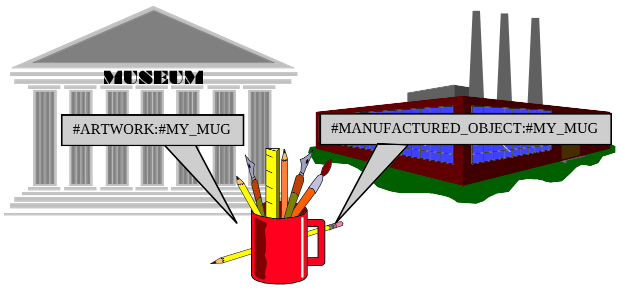

Figure 2 - My mug

• *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.

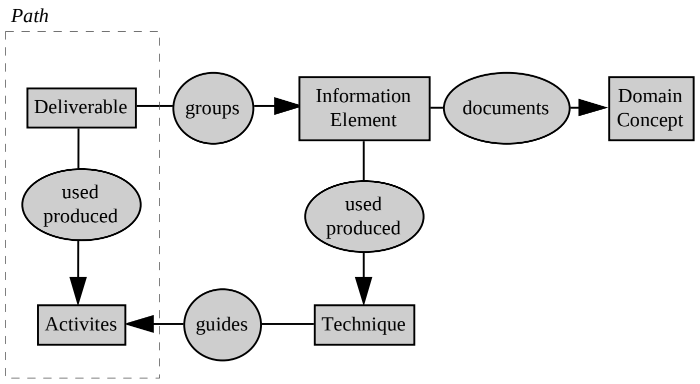

Figure 4 - Schematic Method Model

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.

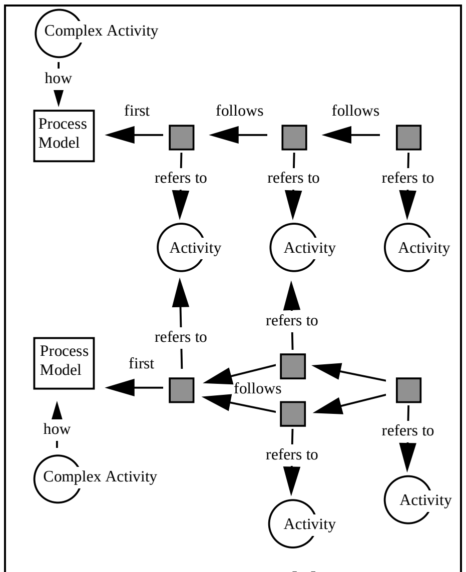

Figure 9 - Process Model Diagram

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