Published in the Proceedings of the Sixth International Conference on Conceptual Structures (ICCS'98)

Conceptual Graphs for Representing Business Processes in Corporate Memories

Olivier Gerbé[1], Rudolf K. Keller[2], and Guy W. Mineau[3]

[1] DMR Consulting Group Inc.
1200 McGill College, Montréal, Québec, Canada H3B 4G7
Olivier.Gerbe@dmr.ca
[2] Université de Montréal
C.P. 6128 Succursale Centre-Ville, Montréal, Québec, Canada H3C 3J7
keller@IRO.UMontreal.ca
[3] Université Laval
Québec, Québec, Canada G1K 7P4
mineau@ift.ulaval.ca

Abstract. This paper presents the second part of a study conducted at DMR Consulting Group during the development of a corporate memory. It presents a comparison of four major formalisms for the representation of business processes: UML (Unified Modeling Language), PIF (Process Interchange Format), WfMC (Workflow Management Coalition) framework and conceptual graphs. This comparison shows that conceptual graphs are the best suited formalism for representing business processes in the given context. Our ongoing implementation of the DMR corporate memory – used by several hundred DMR consultants around the world – is based on conceptual graphs, and preliminary experience indicates that this formalism indeed offers the flexibility required for representing the intricacies of business processes.

1 Introduction

Charnel Havens, EDS (Electronic Data Systems) Chief Knowledge Officer, presents in [5] the issues of knowledge management.

> *With a huge portion of a company's worth residing in the knowledge of its employees, the time has come to get the most out of that valuable corporate resource – by applying management techniques.*

The challenge companies will have to meet is the memorization of knowledge as well as its storage and its dissemination to employees throughout the organization. Knowledge may be capitalized on and managed in corporate memories in order to ensure standardization, consistency and coherence. Knowledge management requires the acquisition, storage, evolution and dissemination of knowledge acquired by the organization [14], and computer systems are certainly the only way to realize corporate memories [15] which meet these objectives.

DMR Consulting Group Inc. has initiated the IT Macroscope project [7], a research project that aims to develop methodologies allowing organizations: i) to use IT (Information Technology) for increasing 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 that IT investments are profitable. In parallel with methodology development, tools for designing and maintaining these methodologies, designing training courses, and for managing and promoting IT Macroscope products were designed. These tools implement the concept of corporate memory. This corporate memory, called the Method Repository, plays a fundamental role. It captures, stores [3], retrieves and disseminates [4] throughout the organization all the consulting and software engineering processes and the corresponding knowledge produced by the experts in the IT domain.

During the early stage of the development of the Method Repository, the choice of a knowledge representation formalism was identified as a key issue. That lead us to define specific requirements for corporate memories, to identify suitable knowledge representation formalisms and to compare them in order to choose the most appropriate formalism. We identified two main aspects: knowledge structure and dynamics – business processes, together with activities, events, and participants. The first part of the study [2] lead us to adopt the conceptual graph formalism for structural knowledge. Uniformity of the formalism used in the Method Repository was one issue but not the all-decisive one in adopting conceptual graphs for the dynamic aspect, too. Rather, our decision is based on the comparison framework presented in this paper.

In our comparison, we studied four major business modeling formalisms or exchange formats, UML (Unified Modeling Language), PIF (Process Interchange Format), WfMC (Workflow Management Coalition) framework, and conceptual graphs, against our specific requirements. Choosing these four formalisms for our study has been motivated by the requirement for building our solution on existing or de facto standards. Our study demonstrates that conceptual graphs are particularly well suited for representing business processes in corporate memories since they support: (i) shared activities, and (ii) management of instances.

The paper is organized as follows. Section 2 introduces the basic notions of business processes as used in this paper. Section 3 defines specific requirements for the representation of business processes in corporate memories. Section 4 compares the four formalisms. Finally, Section 5 reports on the on-going implementation of the Method Repository and discusses future work.

2 Basic Notions

In this section, we present basic notions relevant to the representation of business processes. Main notions of representation of the dynamics in an enterprise are processes, activities, participants (input, output, and agent), events (preconditions and postconditions), and notions of sequence and parallelism of activity

executions. These notions build upon some commonly used definitions in enterprise modeling, as summarized in the following paragraph.

A *process* is seen as a set of activities. An *activity* is a transformation of input entities into output entities by agents. An *event* marks the end of an activity; the event corresponds to the fulfilment of both the activity's postcondition and the precondition of its successor activity. An *agent* is a human or material resource that enables an activity. An *input* or *output* is a resource that is consumed of produced by an activity. The notions of sequence and parallelism define the possible order of activity executions. *Sequence* specifies an order of executions and *parallelism* specifies independence between executions.

Figure 1 presents the notions of activity, agent, input, and output. Activities are represented by a circle and participants of activities are represented by rectangles and linked to their respective activities by arcs ; directions of arc define their participation: input, output or agent. There is no notational distinction between input and agent. Note that this simple process representation exclusively serves for introducing terminology and for illustrating our requirements.

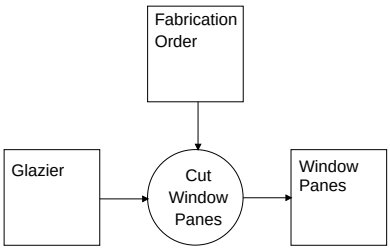

Fig. 1. Activity with input, output and agent.

Figure 2 illustrates the notions of sequence and parallelism by a process composed of five activities. The activity `Write Production Order` is the first activity of the process, the activities `Build Frame` and `Cut Panes` are executed in parallel, `Assemble Window` follows the activities `Build Frame` and `Cut Panes`, and finally `Deliver Window` terminates the process. Note that we only have to consider the representation of parallel activities or sequential activities; all other cases can be represented by these two cases by splitting activities into sub-activities.

3 Requirements

This section introduces the two main requirements underlying our study: representation of processes sharing activities and management of instances that are involved in a process. It is obvious that there exists a lot of other requirements to represent a business process in a corporate memory. Since these other requirements are mostly met by all the formalisms studied, we decided to focus on the two main requirements mentioned above.

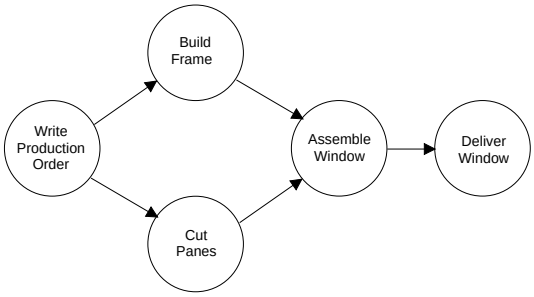

Fig. 2. A process as a set of activities.

3.1 Sharing Activities

Let us consider the case of two processes that share a same activity. Figure 3 illustrates this settings.

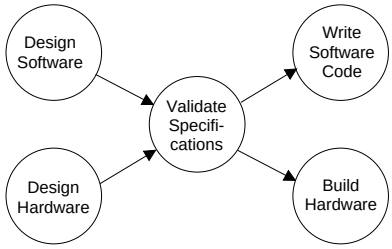

Fig. 3. Sharing Activities.

The example depicted in Fig. 3 deals with the fabrication of a product which is made out of two components: a software component and a hardware component, as in a cellular phone, a microwave oven or a computer. The first process describes the development of the software component. It is composed of `Design Software`, `Validate Specifications`, and `Write Software Code`. The second process describes the development of the hardware component. It is composed of activities `Design Hardware`, `Validate Specifications`, and `Build Hardware`. The activity `Validate Specifications` is an activity of synchronization and is shared by the two processes. The problem in this example is the representation and identification of the two processes. Each process is composed of three activities, with one of them being in common. Therefore the formalism must offer reuse of parts of process definitions or support some kind of shared variable mechanism.

To support the representation of business processes in corporate memory, a formalism must offer features to represent processes sharing the same activities.

3.2 Instance Management

To illustrate the problem of instance management, let us assume the example of a window manufacturer who has a special department for building non standard size windows. Figure 4 presents the window fabrication process.

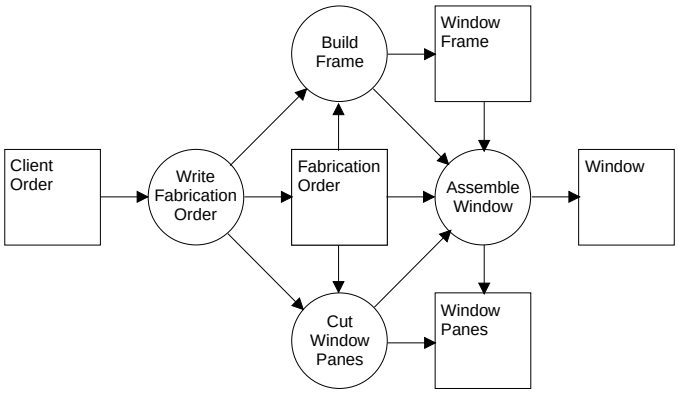

Fig. 4. The Window Problem.

A fabrication order is established from a client order. A fabrication order defines the size and material of the frame and the size and thickness of the glasses to insert into the frame. The fabrication order is sent to the frame builder and glass cutter teams which execute the order. Then the frame and glasses are transmitted to the window assembly team which insert the glasses into the frame. The problem of this team is to insert the right glasses (size and thickness) into the right frames (size and material). Some frames take more time to build than others, so the frames may be finished in a different order than the glasses are. This problem can be solved by the assembly team by assembling the frame and glasses in conformity with the fabrication order. At the notational level, this requires the possibility of specifying *instances* of input and output participants.

To support representation of business processes in corporate memory, the formalism must offer features to represent and manage the related instances needed by different processes.

4 Formalisms

This section presents the four business process modeling formalisms of our study. These formalisms offer representation features in order to describe, exchange, and execute business processes. Each of the studied formalisms supports the representation of the basic notions introduced in Section 2, so we concentrate on the specific requirements discussed above. Against these requirements we have evaluated the four formalisms, UML [1] (Unified Modeling Language), PIF [8]

(Process Interchange Format), WfMC framework [6] (Workflow Management Coalition) and conceptual graphs. Other formalisms, Petri net [16] and CML [11, 12], have been considered but not included in this study because not well-suited to represent business processes or not enough formal.

4.1 Unified Modeling Language

In [2] we presented how to represent the static structure in UML [1] (Unified Modeling Language). Let us recall that UML, developed by Grady Booch, Jim Rumbaugh and Ivar Jacobson from the unification of Booch method, OMT and OOSE, is considered as a de facto standard.

UML provides several kinds of diagrams that allow to show different aspects of the dynamics of processes. *Use Case diagrams* show interrelations between functions provided by a system and external agents that use these functions. *Sequence diagrams* and *Collaboration diagrams* present interactions between objects by specifying messages exchanged among objects. *State diagrams* describe the behavior of objects of a class or the behavior of a method in response to a request. A state diagram shows the sequence of states an object may have during its lifetime. It also shows responsible requests for state transitions, responses and actions of objects corresponding to requests. *Activity diagrams* have been recently introduced in UML. They are used to describe processes that involve several types of objects. An activity diagram is a special case of state diagram where states represent the completion of activities.

In the context of corporate memory , activity diagrams are the most relevant and we will present their main concepts in what follows. In UML, there are two types of execution of activities: execution of activities that represent atomic actions, they are called *ActionState*, and execution of a non atomic sequence of actions, they are called *ActivityState*. Exchange of objects among actions are modeled by object flows that are called *ObjectFlowState*. ObjectFlowStates implements notions of inputs and ouptuts. Agents are represented by *Swimlane* in activity diagrams. However it is possible to define agent as a participant to an activity and to establish explicitly a relationship between agent and activity.

Figure 5 shows how to model the cut window pane activity with participants. Activity diagrams shows possible scenarios; this means that activity diagrams

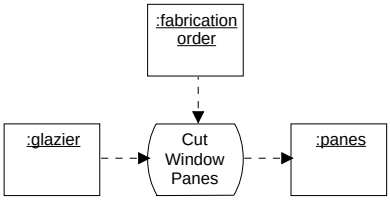

Fig. 5. UML - The cut window pane Activity.

show objects instead of classes. Dashed arrows link inputs and outputs to activities.

Processes may be represented using activity diagrams in UML and Fig. 6 shows an example of the window building process. Solid arrows between processes represent the control flow.

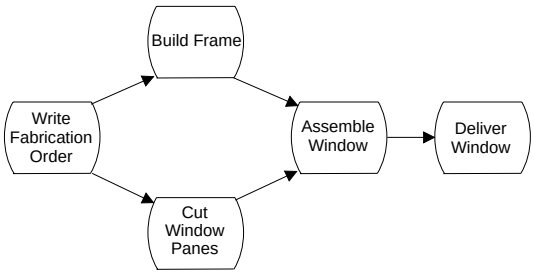

Fig. 6. UML - The whole Process.

Sharing Activities As detailed in [1], UML does not support adequate representation features for sharing activities. However activity diagrams are new in the definition of the language and all cases have not been yet presented.

Instances Management In opposition with the representation of structure [2], the process representation is done at the instance level. Activity diagrams involve objects not classes and therefore it is possible to represent the window problem by using the object `fabrication order` which specifies `frame` and `panes`. Figure 7 shows a representation for the window problem.

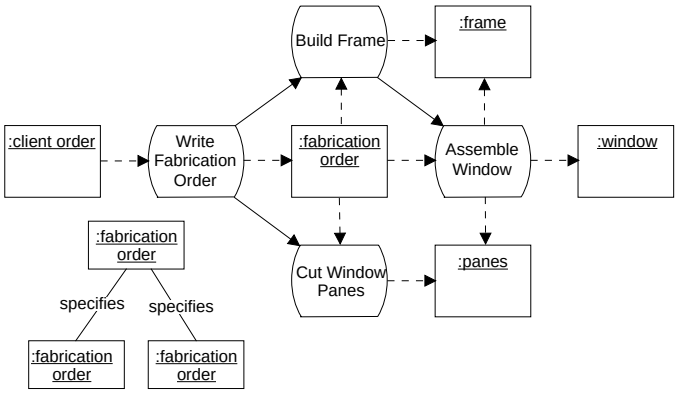

Fig. 7. UML - The Window Problem.

4.2 Process Interchange Format (PIF)

The PIF (Process Interchange Format) workgroup, composed of representatives from companies and universities developed a format to exchange the specifications of processes [8].

A PIF process description is a set of frame definitions. Each frame specifies an instance of one class of the PIF metamodel. Figure 8 shows PIF metamodel. It is composed of a generic class `ENTITY` from which all other classes are derived and of four core classes: `ACTIVITY`, `OBJECT`, `TIMEPOINT`, and `RELATION`. Subclasses

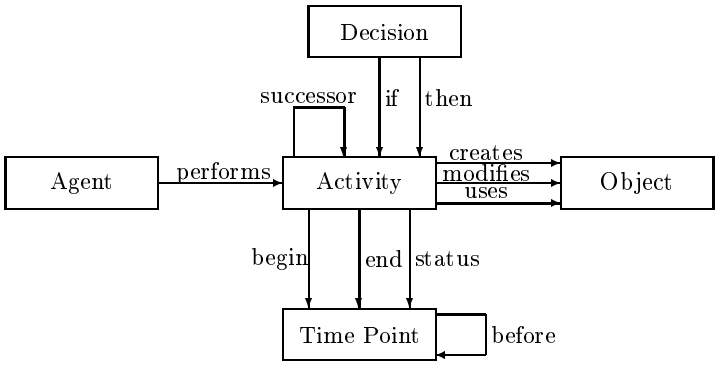

Fig. 8. PIF - Metamodel.

of `ACTIVITY` and `OBJECT` are respectively `DECISION` and `AGENT`. Class `RELATION` has seven subclasses, subclasses `CREATES`, `MODIFIES`, `PERFORMS`, and `USES` define relationships between `ACTIVITY` and `OBJECT`, the subclass `BEFORE` defines a predecessor relationship between two points in time, the subclass `SUCCESSOR` defines a successor relationship between two activities and, `ACTIVITY-STATUS` defines the status of an activity at a point in time.

Figure 9 shows the representation of an activity using the PIF format. ACT1

```
(define-frame ACT1
:own-slots
((Instance-Of ACTIVITY)
(Name "Cut Window Panes")
(End END-ACT1)))

(define-frame END-ACT1
:own-slots
((Instance-Of TIMEPOINT)))

(define-frame AGT1
:own-slots
((Instance-Of AGENT)
(Name "Glazier")))

(define-frame PRFRMS1
:own-slots
((Instance-Of PERFORMS)
(Actor AGT1)
(Activity ACT1)))

(define-frame INPUT1
:own-slots
((Instance-Of OBJECT)
(Name "Fabrication Order")))

(define-frame USES1
:own-slots
((Instance-Of USES)
(Activity ACT1)
(Object INPUT1)))

(define-frame OUTPUT1
:own-slots
((Instance-Of OBJECT)
(Name "panes")))

(define-frame CRTS1
:own-slots
((Instance-Of CREATES)
(Activity ACT1)
(Object OUTPUT1)))
```

Fig. 9. PIF - Activity with participants.

defines the cut window panes activity as an instance of ACTIVITY with a name and a relation to END-ACT1. END-ACT1 represents the end of the activity and is defined as a point in time. Then come definitions of the three participants; each participant is defined in two parts: definition of the participant itself and definition of the relationship between the activity and the participant.

With the PIF process interchange format and framework, there is no explicit definition of a process. A process is the set of defined activities. Example shown in Fig. 10 shows how two activities ACT1 and ACT2 are linked by a BEFORE relationship.

```
(define-frame ACT1
:own-slots
((Instance-Of ACTIVITY)
(Name "Write Fabrication Order")
(End END-ACT1)))

(define-frame END-ACT1
:own-slots
((Instance-Of TIMEPOINT)))

(define-frame ACT2
:own-slots
((Instance-Of ACTIVITY)
(Name "Build Frame")
(End END-ACT2)))

(define-frame END-ACT2
:own-slots
((Instance-Of TIMEPOINT)))

(define-frame ACT1-ACT2
:own-slots
((Instance-Of BEFORE)
(Preceding-Timepoint END-ACT1)
(succeeding-Timepoint END-ACT2)))
```

Fig. 10. PIF - Process.

Sharing Activities The PIF format supports representation of several sequences of activities. It is possible to define in one file more than one sequence of activities by a set of frames instance of BEFORE. However it is not possible to explicitly identify several processes.

Instance Management With the PIF format activities and participants involved in the activities are described at the type level. Therefore, it is not possible to identify instances in PIF activity definitions.

4.3 Workflow Reference Model

The Workflow Management Coalition (WfMC) defines in the Workflow Reference Model [6] a basic metamodel that supports process definition. The Workflow Reference Model defines six basic object types to represent relatively simple processes. These types are: `Worflow Type Definition`, `Activity`, `Role`, `Transition Conditions`, `Workflow Relevant Data`, and `Invoked Application`. Figure 11 shows the basic process definition metamodel. The Workflow Management Coalition has also published a Process Definition Interchange in version 1.0 beta [17] that describes a common interface to the exchange of process definitions between workflow engines. Figure 12 presents the definition of the activity `Cut Window Panes` using this exchange format. Participants (inputs or agents) to an activity are defined explicitly. Data that are created or modified by an activity are defined in the postconditions of the activity or defined as output parameters of invoked applications during activity execution. In WFMC Process Definition Interchange format, a process is defined as a list of activities and a list of transitions that

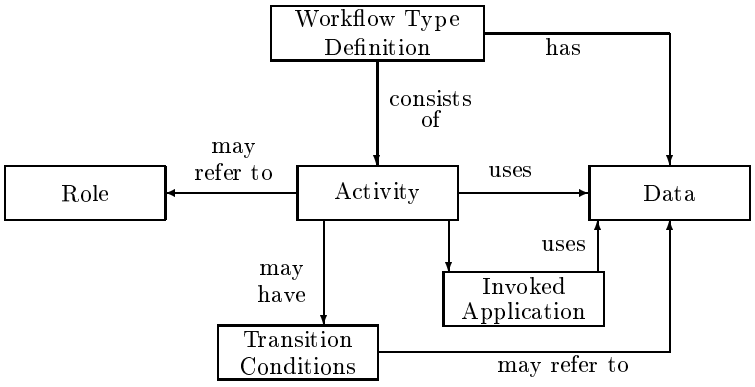

Fig. 11. WfMC - Basic Process Definition MetaModel.

specify in which order activities are executed. In Fig. 13 of the following section, examples of definitions of activities in WFMC interchange format are shown.

```
ACTIVITY Cut_Window_Panes
 PARTICIPANT Glazier, Fabrication_Order
 POST CONDITION Window_Panes exists
END ACTIVITY

PARTICIPANT Glazier
 TYPE HUMAN
END PARTICIPANT

DATA Fabrication_Order
 TYPE COMPLEX DATA
END DATA

DATA Window_Panes
 TYPE REFERENCE
END DATA
```

Fig. 12. WfMC - Activity with Participants.

Sharing Activities Processes are defined using keyword WORKFLOW and END-WORKFLOW which respectively begins and ends a process definition. In a process definition, it is possible to use activities or participants that have been defined in another process definition. In the example shown in Fig. 13, two processes are defined with a common activity. The commom activity is defined in process 1 and reused in process 2.

Instance Management Process definitions are defined at type level. However conditions that fire activity or that are realized at the end of an activity are expressed using Boolean expressions with variables. In theory, it is possible to represent the window problem but the version 1.0 beta of Process Definition Interchange [17] gives few indications to realize it.

```
WORKFLOW PROCESS1
 ACTIVITY Design_Software
  . . .
 END_ACTIVITY

 ACTIVITY Validate_Specifications
  . . .
 END_ACTIVITY

 ACTIVITY Write_Software_Code
  . . .
 END_ACTIVITY

 TRANSITION
   FROM Design_Software
   TO Validate_Specifications
 END_TRANSITION

 TRANSITION
   FROM Validate_Specifications
   TO Write_Software_Code
 END_TRANSITION
END_WORKFLOW
```

```
WORKFLOW PROCESS2
 ACTIVITY Design_Hardware
  . . .
 END_ACTIVITY

 ACTIVITY Build_Hardware
  . . .
 END_ACTIVITY

 TRANSITION
   FROM Design_Hardware
   TO Validate_Specifications
 END_TRANSITION

 TRANSITION
   FROM Validate_Specifications
   TO Build_Hardware
 END_TRANSITION
END_WORKFLOW
```

Fig. 13. WfMC - Processes Sharing Activities.

4.4 Conceptual Graphs and Processes

In conceptual graph theory, there is no standard way to represent processes. Processes have not been extensively studied and only a few works are related to the representation of processes. John Sowa in [13] presents some directions to represent processes. Dickson Lukose [9] and Guy Mineau [10] have proposed executable conceptual structures.

We present below a possible metamodel to represent processes that fulfills corporate memory requirements as expressed in Section 3. The metamodel (Fig. 14) is composed of three basic concepts: `ACTIVITY`, `PROCESS`, and `EVENT`. An activity

```
TYPE ACTIVITY(x) IS
 [T:*x]-
  (INPUT)<-[T:*i]
  (OUTPUT)<-[T:*e]
  (AGENT)<-[T:*a]
  (DEPENDS-ON)->[PRECONDITION:*pre]
  (REALIZES)->[POSTCONDITION:*post]
```

```
TYPE EVENT(x) IS
 [T:*x]-
  (END)->[EVENT:*ev1]
  (FOLLOWS)<-[EVENT:*ev2]

TYPE PROCESS(x) IS
 [T:*x]-
  (FIRST)<-[EVENT:*].
```

Fig. 14. Conceptual Graphs - Metamodel.

is defined by its inputs and outputs, the agents that enable the activity, and by pre and post conditions. Preconditions define conditions or states that must be verified to fire the execution of the activity; postconditions define states or conditions that will result from the execution of the activity. An event is a point in time that marks the end of an activity; it marks the realization of the postcondition of the activity. A process is defined as a set of events that represent the execution of a set of activities.

Using this metamodel, the `Cut-Window-Panes` process is defined by the definition graph presented in Fig. 15 where two variables with the same name represent the same object.

```
TYPE CUT-WINDOW-PANES(x) IS
  [ACTIVITY:*x]-
    (AGENT)<-[GLAZIER:*]
    (INPUT)<-[ORDER:*o]
    (OUTPUT)<-[PANES:*v]
    (REALIZES)->[POSTCONDITION:[ORDRE:*o]-
          (CONFORMS)<-[PANES:*v]].
```

Fig. 15. Conceptual Graphs - The Cut Window Panes Activity.

The process to build a window is represented by the definition graph shown in Fig. 16.

```
TYPE BUILD-WINDOW(x) IS
  [PROCESS:*x]-
    (FIRST)<-[EVENT:*ev1]-
      (END)->[WRITE-FABRICATION-ORDER:*]
      (FOLLOWS)<-[EVENT:*ev2a]-
        (END)->[BUILD-FRAME:*],
      (FOLLOWS)<-[EVENT:*ev2b]-
        (END)->[CUT-WINDOW-PANES:*]
        (FOLLOWS)<-[EVENT:*ev3]-
          (END->[ASSEMBLE-WINDOW:*]
          (FOLLOWS)<-[EVENT:*ev3]-
            (END)->[DELIVER-WINDOW:*].
```

Fig. 16. Conceptual Graphs - Process.

Sharing Activities The proposed model allows the representation of processes that share a same activity (as indicated by variables under a global coreference assumption[1]). Figure 17 shows two processes that share the same activity

```
TYPE PROCESS1(x) IS
  [PROCESS:*x]-
    (FIRST)<-[EVENT:*ev1a]-
      (END)->[DESIGN-HARDWARE:*]
      (FOLLOWS)<-[EVENT:*ev2a]-
        (END)->[VALIDATE-SPECIFICATIONS:*vs]
        (FOLLOWS)<-[EVENT:*ev3a]-
          (END)->[BUILD-HARDWARE:*].
```

```
TYPE PROCESS2(x) IS
  [PROCESS:*x]-
    (FIRST)<-[EVENT:*ev1b]-
      (END)->[DESIGN-SOFTWARE:*]
      (FOLLOWS)<-[EVENT:*ev2b]-
        (END)->[VALIDATE-SPECIFICATIONS:*vs]
        (FOLLOWS)<-[EVENT:*ev3b]-
          (END)->[WRITE-SOFTWARE-CODE:*].
```

Fig. 17. Conceptual Graphs - Processes Sharing Activities.

`VALIDATE-SPECIFICATIONS`. Each process is defined by a sequence of events, and one event of each process marks the end of the activity.

[1] The proposed model assumes global coreference. Two variables with the same identifier represent the same concept.

Instance Management. Figure 18 shows that with the use of variables and the global coreference assumption, conceptual graphs support the representation of

```
TYPE WRITE-FABRICATION-ORDER(x) IS
  [ACTIVITY:*x]-
    (INPUT)<-[CLIENT-ORDER:*c]
    (OUTPUT)<-[ORDER:*o].

TYPE BUILD-FRAME(x) IS
  [ACTIVITY:*x]-
    (INPUT)<-[ORDER:*o]
    (OUTPUT)<-[FRAME:*f]
    (REALIZES)->[POSTCONDITION:[ORDER:*o]-
          (CONFORMS)<-[FRAME:*f]].

TYPE CUT-WINDOW-PANES(x) IS
  [ACTIVITY:*x]-
    (INPUT)<-[ORDER:*o]
    (OUTPUT)<-[PANES:*p]
    (REALIZES)->[POSTCONDITION:[ORDER:*o]-
          (CONFORMS)<-[PANES:*v]].
```

```
TYPE ASSEMBLE-WINDOW(x) IS
  [ACTIVITY:*x]-
    (INPUT)<-[ORDER:*o]
    (INPUT)<-[PANES:*p]
    (INPUT)<-[FRAME:*f]
    (DEPENDS-ON)->[PRECONDITION:[ORDER:*o]-
          (CONFORMS)<-[PANES:*p]
          (CONFORMS)<-[FRAME:*f]]
    (OUTPUT)<-[WINDOW:*w].
```

Fig. 18. Conceptual Graphs - The Window Problem.

the window problem. The concept type definition of `WRITE-FABRICATION-ORDER`, `BUILD-FRAME`, `CUT-WINDOW-PANES`, and `ASSEMBLE-WINDOW` specify that the frame and panes involved in assemble window are conformed to the fabrication order.

4.5 Summary

Table 1 presents a summary of this survey on business process representation formalisms. This summary shows that the framework proposed by the WfMC

Table 1. Summary

	Sharing Activities	Instances Management
UML	No	Yes
PIF	Yes	No
WfMC	Yes	Yes
CG	Yes	Yes

and conceptual graphs fulfill our requirements for the representation of business processes in corporate memories. However, the first part of our study [2] identified conceptual graphs as the best-suited formalism for knowledge structure. Therefore, for the sake of uniformity of formalism, we chose conceptual graphs.

5 Experience and Future Work

Using conceptual graph formalism, a corporate memory has been developed at the Research & Development Department of DMR Consulting Group Inc in or-

der to memorize the methods, know-how and expertise of its consultants. This corporate memory, called Method Repository, is a complete authoring environment used to edit, store and display the methods used by the consultants of DMR. The core of the environment is the CG Knowledge Base; it is a knowledge engineering system based on conceptual graphs. Four methods are commercially delivered: Information Systems Development, Architecture, Benefits Realization, and Strategy; their documentation in paper and in hypertext format is generated from conceptual graphs. About two hundred business processes have been modeled and from about 80,000 conceptual graphs, we generated more than 100,000 HTML pages in both English and French that can be browsed using commercial Web browsers.

This paper has described the research we have done to identify which formalism was the most suitable for the representation of business processes in corporate memories. We have compared four formalisms and this comparison has shown as in a previous study [2] how conceptual graphs are a good response to the specific requirements involved in the development of corporate memories.

References

[1] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language, Version 1.1.* Rational Software Corporation, 1997.

[2] O. Gerbé. Conceptual graphs for corporate knowledge repositories. In *Proceedings of 5th International Conference on Conceptual Structures*, pages 474–488, 1997.

[3] O. Gerbé, B. Guay, and M. Perron. Using conceptual graphs for methods modeling. In *Proceedings of the 4th International Conference on Conceptual Structures*, 1996.

[4] O. Gerbé and M. Perron. Presentation definition language using conceptual graphs. In *Peirce Workshop Proceedings*, 1995.

[5] C. Havens. Enter, the chief knowledge officer. *CIO Canada*, 4(10):36–42, 1996.

[6] D. Hollingsworth. *The Workflow Reference Model.* Workflow Management Coalition, 1994.

[7] DMR Consulting Group Inc. *The IT Macroscope Project*, 1996.

[8] J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, and Yost G. other members of the PIF Working Group. *The PIF Process Interchange Format and Framework (May 24, 1996)*, 1996. availaible at http://soa.cba.hawaii.edu/pif/.

[9] D. Lukose. Model-ecs: Executable conceptual modelling language. In *Proceedings of Knowledge Acquisition Workshop (KAW96)*, 1996.

[10] D. Lukose and G.W Mineau. A comparative study of dynamic conceptual graphs. In *Accepted for publication at the 11th KAW*, 1998.

[11] A. Schreiber, B. Wielenga, H. Akkermans, W. Van de Velde, and A. Anjewierden. Cml: The commonkads conceptual modelling language. In L. Steels, A. Schreiber, and W. Van de Velde, editors, *Proceedings of the 8th European Knowledge Acquisition Workshop (EKAW'94)*, pages 1–24. Springer-Verlag, 1994.

[12] G. Schreiber, B. Wielenga, H. Akkermans, W. Van de Velde, and A. Anjewiereden. Cml: The commonkads conceptual modelling language. In *Proceedings of the 8th European Knowledge Acquisition Workshop (EKAW'94)*, 1994.

[13] J. Sowa. Processes and participants. In P. Eklund, G. Ellis, and G. Mann, editors, *Proceedings of the 4th International Conference on Conceptual Structures, ICCS'96*, pages 1–22. Springer, 1996.

[14] E. W. Stein. Organizational memory: Review of concepts and recommendations for management. *International Journal of Information Management*, 15(1):17–32, 1995.
[15] G. van Heijst, R. van der Spek, and E. Kruizinga. Organizing corporate memories. In *Proceedings of the Knowledge Acquisition Workshop*, 1996.
[16] WG11. High-level petri net standard - working draft - version 2.5. 1997.
[17] Workflow Management Coalition. *Interface 1: Process Definition Interchange*, 1996.