Automatic Derivation of Workflow Specifications from Organizational Structures and Use Cases[1]

Mª Carmen Penadés, José H. Canós, Juan Sánchez
Departamento de Sistemas Informáticos y Computación,Universidad Politécnica de Valencia
{mpenades|jhcanos|jsanchez }@dsic.upv.es

Abstract. Workflow technology has reached a reasonable degree of maturity, with a number of both research prototypes and commercial systems available. However, methodological issues have received little attention, and WF developers often have to face the WF development process with neither a methodological support nor a global view of the process. In this paper, we introduce a requirements engineering layer in the workflow development lifecycle. It is organization-based, and follows a bottom-up modeling strategy. In order to capture business processes requirements to obtain a workflow model, we describe the tasks in the process as use cases. The use case model is refined by applying specialization, use and extension relationships, as we go up in the organizational hierarchy. A preliminary workflow model implementing the business processes is obtained automatically from the use case model. The transformation is driven by a set of rules derived from the equivalencies between use case and workflow concepts plus a set of process patterns. A tool supporting our method has been implemented and is outlined in this work.

Keywords: Workflow specification, use cases, business processes requirements.

1. Introduction and Motivation

A workflow (WF) has been defined as "*the automation of a business process (BP), in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules*" [1]. Workflow Management Systems (WFMSs) are software systems that allow users to define, manage and execute WFs in heterogeneous and distributed environments.

During the last decade, WF technology has reached a reasonable degree of maturity, with a number of both research prototypes and commercial systems available. However, most of the work done so far in the WF management field has been tool-oriented and technological in nature. Methodological issues have received little attention, and WF developers often have to face the WF development process with neither methodological support nor a global view of the process [2]. This often leads developers to build WF models from scratch, without a clear statement of the BP

[1] This work has been supported by CICYT (Project DOLMEN-SIGLO) TIC2000-1673-C06-01)

they are trying to represent in terms of WF concepts. Moreover, the level of expertise needed to specify a WF model is rather high and WFMS-dependent due to the lack of a widely accepted WF model (the Workflow Management Coalition (WfMC) Reference model [3], though published in 1994, is not fully implemented in current WFMSs).

In [4], we claimed that many of the principles of Software Engineering could be applied to the WF development process if WFs are considered as complex software products. In particular, techniques, methods and tools used in Requirements Engineering can be applied to WF modeling and help us to build complete and correct WF models.

In this paper, we introduce a requirements engineering layer to capture BP requirements and obtain a WF model from them. It is organization-based and follows a bottom-up modeling strategy. We describe the tasks in the process as extended use cases (UC) that are refined by applying specialization, use (*uses*) and extension (*extends)* relationships, as you go up in the organizational hierarchy. A preliminary WF model implementing the BP is obtained automatically from the UC model. The transformation is driven by a set of rules which is derived from the equivalencies between UC and WF concepts, as well as a set of process patterns. The WF model generated is finally improved using the editing tool of the WFMS used to automate the BP.

This paper is organized as follows. The target WF model and the organizational model used in our proposal are defined in section 2 and section 3, respectively. Section 4 describes the extensions made to the UML's use case model [5] in order to cope with all dimensions of BPs. Section 5 describes the rules that govern the generation of WF models from UC models. The generation procedure consists of five steps, which are described using an example in section 6. Our method is supported by a tool, the architecture of which is shown in Section 7. The last section gives the conclusions and links this work with other related works.

2. Workflow Model

A *WF specification* (also called *WF type*) is the description in an executable language of a BP as a set of *activities* which use resources and which are performed in a given order (defined by the *control flow*) by zero or more *actors* within an *organization*. Different kinds of conditions specify the circumstances under which activities can be started or terminated, as well as how control flow passes from one activity (or a set or activities) to the following one. An activity may produce output data that can be used as input by its successor in the control flow (*data flow*).

Figure 1 shows the WF definition metamodel that we use in this paper. It is specified in UML notation [5]. The main elements of the model are the following:

- ***Process***: a process is composed by activities and/or subprocesses and/or transition conditions. The control flow connects these elements and establishes the correct process execution order. For a good understanding of the model, the relationships modeling the control flow between these elements are not explicitly

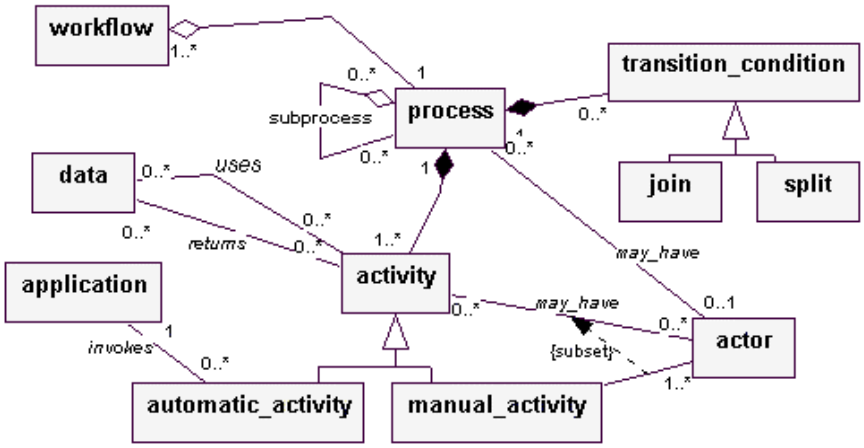

Figure 1. Workflow model

represented in figure 1. Among other attributes, every process has an identifier, a name, a description, a start condition, an end condition and a state.

- ***Activity***: it is any atomic piece of work that constitutes a logical step within a process. Like a process, every activity has an identifier, a name, a description, a start condition, an end condition, a state and a set of specifically associated actions An activity may be manual or automatic; human actors execute manual activities (e.g., filling out a form or making a decision), whereas the automatic ones are executed by a computer and normally consist in the invocation of an external application.
- ***Subprocess***: A subprocess is a process that is part of another process, that is, it constitutes a complex step in a process. This allows the introduction of modularity in WF models.

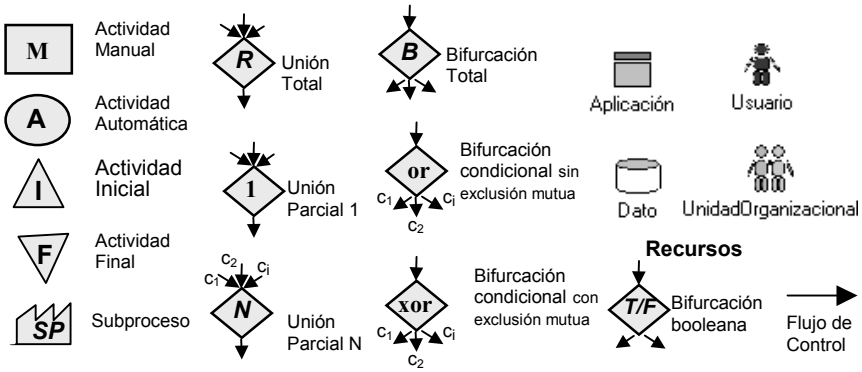

Figure 2. Graphical language to represent WF models

- ***Transition Condition***: It is possible to include the following transition conditions in the control flow of a process: *AND-Join*, *OR-Join*, *AND-Split* and *OR-Split*.
- ***Data***: they are all the information needed for the process execution (either as activity input/output or in the evaluation of the transition conditions). Normally, they are persistent (stored in a database or repository). When an activity begins, it consults the input data from the repository; when the activity ends, it stores the output data in the same repository.
- ***Actor***: it represents the human participation in the WF. It is the connecting point to the organizational model we introduce in the next section.

We have defined a graphical language to represent WF models in an intuitive way. The symbols representing the above concepts are shown in figure 2.

3. Organizational Model

The organizational model we use is represented in the UML class diagram of figure 3. It is similar to the organizational models proposed in the WF literature (e.g. [3], [6]). We describe an *organization* as a hierarchy of *actors* who perform different activities inside the organization. Actors may be individuals called *users* or *organizational units* (e.g. departments) including several actors. A user may play different *roles* in the organization replacing another user in the fulfillment of an activity.

The hierarchy of the organization is defined in terms of two relationships. First, a membership relationship defines the composition of each organizational unit. It is represented by the aggregations *has_users* and *has_units* in figure 3. Users may belong to one or more organizational units, whereas an organizational unit may be only part of a higher-level organizational unit. And second, every organizational unit is *managed_by* a user.

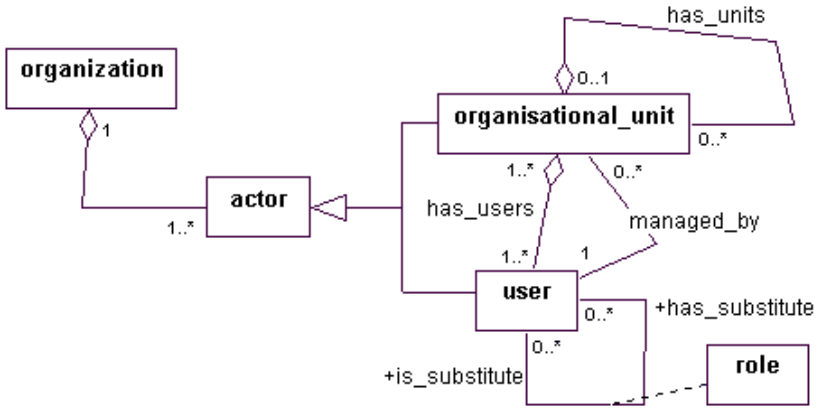

Figure 3. Organizational model

Our proposal consists in the definition of an iterative requirements elicitation process driven by the structure of the organization. Specifically, in section 6 we show how the BP requirements are captured by means of interviews with the members of the organization through a bottom-up traversal of the organizational hierarchy. We start describing the activities performed by individuals inside an organizational unit; later, activities are time-ordered using the knowledge the unit manager has about the global responsibility of the unit. This process iterates in higher levels of the organization until a WF model representing the BP is completed.

4. Modeling Business Processes with Extended Use Cases

Nowadays, building the UC model is the initial step in most of the object-oriented software development methods (e.g. Fusion [7], Octopus [8], UML [9][10]). UCs allow us to capture the software functional requirements in a structured, process-oriented way, and their simplicity makes them particularly valuable for interacting with both customers and project managers.

A UC describes a process or task as a sequence of *events* which are exchanged between the *actors* in the process and the system when (part of) the functionality defined in the UC is executed. A UC has a name, a numeric identifier, a purpose, a reference to the UC diagram in which it is placed and a description of the process it represents. This description is composed of the following elements:

- ***Precondition:*** the state of the system required for the UC to be executed.
- ***Post-condition:*** the state that the execution of the UC leads the system to.
- ***Actors:*** an ordered list of actors. By actor we mean any entity which is able to exchange information with the system (people, devices, other systems). The first actor in the list is called the main actor, and the remaining ones are secondary actors.
- ***Event flow:*** it shows the events generated by the actors and the system commitments, as well as the *extension points* of the UC that cope with the exceptional behavior of the system.

UC model refinement can be done by linking UCs using any of the following relationships:

- **Use** (*uses*[2])**:** literal insertion of a UC into another UC at a given point in the event flow (the *use point*).
- **Extension** (*extends*)**:** similar to the use relationship, but in this case, the insertion at the *extension point* takes place only if some condition (the *extension condition*) holds.
- **Specialization:** close to the notion of specialization in object-oriented models [5].

We say that a UC is *elementary* if it does not use nor is extended by another UC, and that it is a *complex* UC otherwise.

UCs are particularly useful to process-oriented modeling. However, some aspects of the WF model needed to perform an automatic generation must also be taken into account. In particular, we have extended the properties of a UC with the following:

[2] In recent versions of UML, the uses relationship among UCs has been renamed to includes.

- ***Input data:*** data needed by the UC for its execution; they can come from a database or any other source.
- ***Output data:*** data produced by the UC.
- ***Type of process***: in an elementary UC the process can be either manual or automatic. Regarding complex UCs, it may be impossible to precise the nature of the process as it can be composed of both manual and automatic activities. In this case, we just say that the type of the process is complex.

UCs can be described using two different notations: a graphical one (which is very concise and intuitive) shows only the UCs plus the relationships between them, and also shows the actors involved. A more detailed representation uses a textual template for each UC (see section 6).

5. Equivalencies between UC and WF Concepts.

As follows from previous sections, many of the concepts of the WF model are present in the extended UC model (Table 1 summarizes the equivalencies between both models). Our proposal consists of exploiting the intuitiveness of UCs to use them as a BP requirements description language.

Given a BP expressed as a UC model, the corresponding WF specification can be obtained by means of the following transformation rules:

R1. Every elementary UC in the UC model is transformed into an activity having as actors those specified for the UC.

R2. The event flow of the UC determines the action flow of the activity. Notice that the action flow is a property of the activity and is not shown in the WF model, as activities are the lowest level of granularity in WF models.

R3. The input data and the output data of the UC become the input data and output data of the activity, respectively.

Use Case Concepts	Workflow Concepts
Use Case Model	Workflow Model
Elementary UC	Activity
UC name	Activity name
Process type	Activity type
Input data	Input data
Output data	Output data
Precondition	Start condition
Postcondition	End condition
Actors	Actors
Event flow	Action flow
Complex UC	Subprocess
Event flow + Relationships	Process patterns

Table 1. Equivalencies between UC and WF concepts

R4. Pre- and post-conditions in the UC are transformed into the activity's start and end conditions, respectively.

R5. The process type of the UC determines the type of activity.

In addition, the analysis of the relationships between UCs allows for the automatic generation of complex activities or subprocesses by means of process patterns, two of which are described below.

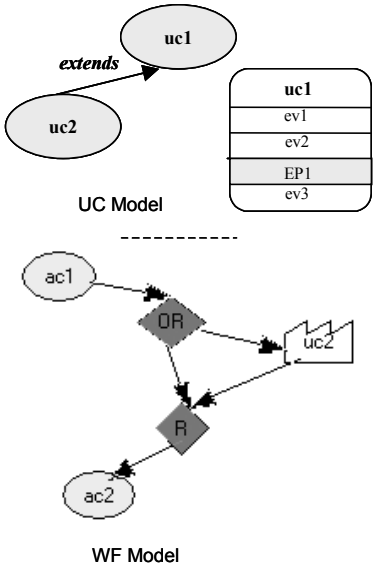

Figure 4. Extends-relationship pattern

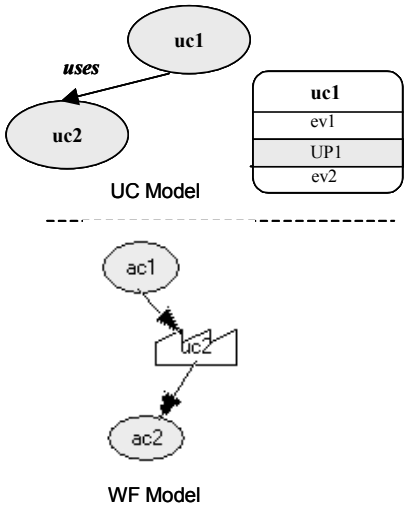

Figure 5. Uses-relationship pattern

5.1 The *extends*-relationship pattern

Figure 4 shows a UC *uc1* which is related to another UC *uc2* by extension; the event flow of *uc1* is *ev1 ev2, EP1, ev3*, where *EP1* is the extension point to *uc2*. The process pattern that corresponds to this case includes a conditional split in the control flow, corresponding to the extension condition of the *uc1*. The activity ac1 includes the event flow before the extension point, and, similarly, *ac2* includes the event flow after the extension point. *UC2* is considered a subprocess and could induce new subprocesses or become an activity, depending on its complexity.

5.2 The *uses*-relationship pattern

Figure 5 shows a UC *uc1* which is related by use to uc2; the event flow of *uc1* is *ev1, UP1, ev2*, with *UP1* being the use point to *uc2*. In this case, the process pattern consists of including the subprocess associated to *uc2* in the control flow between *ac1* and *ac2*. As in the *extends*-relationship pattern, *uc2* complexity could induce new subprocess or become an activity.

6. Deriving Workflow Models from Use Case Models

The WF specification corresponding to a given BP described as an extended UC model is obtained by following a five-step process. To illustrate it, we use the business process that a finance company uses to approve loans to their customers. The process begins when a customer requests a loan and the loan officer collects all credit information. A letter is sent to the customer with the approval or rejection notification when the process finishes. More details about the process are given as we proceed with the generation steps.

Step 1. **Enterprise Structure Modeling.** Having a well-defined organizational model is required to be able to perform steps 2 and 3. The organizational structure of the company is modeled in terms of the model introduced in section 3.

In the example, we will assume that the company is headed by an executive director and has two organizational units: contracts and loans. Each organizational unit has a number of people playing different roles: the contracts unit has two financial officers, whereas the loans unit has three loan officers. Figure 6 shows the organizational structure of the company.

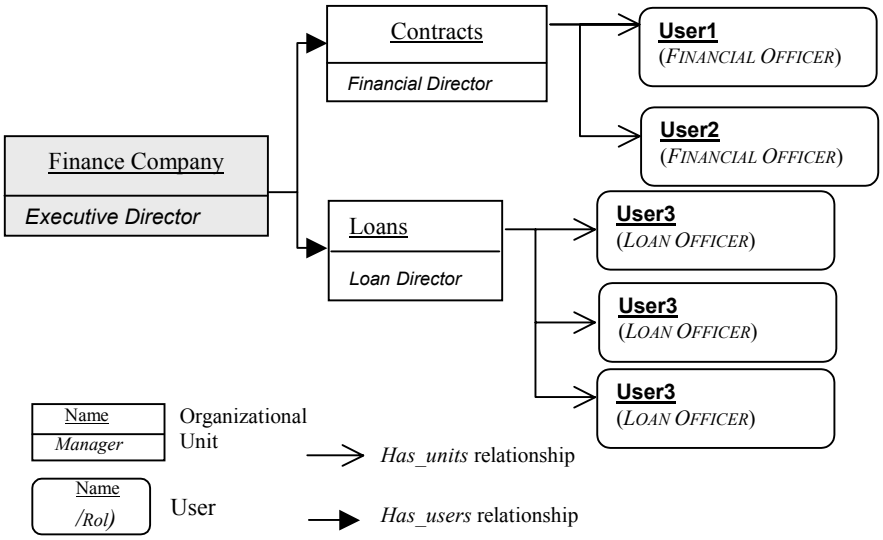

Figure 6. Organizational structure of the financial company

Step 2. Creation of the UCs corresponding to individual tasks in the organization. Each member of the company has its own view of BP. For instance, a financial officer can have a different perspective about the loan tasks than the one a loan officer has. But both views are valuable.

Identification		
ID: *CU0003*	**Name**: *Notify resolution*	
Purpose: Notify resolution to the customer		
Diagram: D1		
Relationships:		
***Specialization*:**	--	
***Uses*:**	"Send Notification"	
***Extends*:**	--	
Process/Description		
Process Type:	Complex	
Input Data:	Credit approval	
Output Data:	Letter to the customer	
Preconditions:		
Post conditions:		
Actors:	Loan officer	
Event Flow (Actor-System Communications)		
(User Intention)		*(System Responsibility)*
1. The loan officer requests the final resolution of the loan		
		2. The system searches for the customer credit approval
3. Use "Send Notification"		
Extensions.		
If at point 2, the client credit approval is true then "Accept Credit" If at point 2, the client credit approval is false then "Reject Credit"		

Figure 7. Textual template of *Notify resolution* use case

The requirements of a BP are captured by means of interviews with the employees that actually perform the activities. The UC representing a particular activity is obtained from the knowledge the actual performer of the task has about it and described in a textual template (see figure 7).

In the example, loan officers perform several tasks. They collect all information related to the loan request and check the risk of the loan. If the risk is high or the amount request is more than $10.000, they send the loan request to the financial unit. Finally, they notify the resolution to the customers sending them an acceptance or rejection letter. Financial officers evaluate the loan request and decide the approval or rejection of the loan. If the loan is rejected but the customer is a preferential customer, the executive director takes the final decision for approval or rejection of the loan.

Step 3. Representation of the relationships among UCs to obtain a UC model. UCs do not exist in isolation, rather they are usually interrelated. The UC model obtained in the previous step can be refined, or new UCs can be created from the previous ones by applying specialization, use and extension relationships. This task may be performed by organizational unit managers, who can produce a refined view of the model due to the wider vision of the business process they have. This may

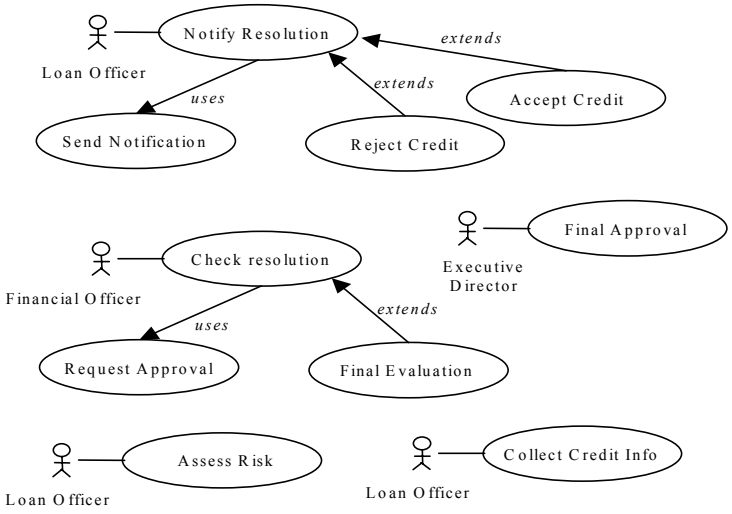

Figure 8. Loan Process Use Case Model

produce more concise models by removing redundancies and factoring duplicate behaviors.

In a complex organization, steps 2 and 3 should be iteratively applied upwards in the organizational structure until the UC model is completed.

In the example, financial officers have a more global vision of the process that loan officers; similarly, the executive director knows the global business logic, but does not need to know each activity in depth. Figure 7 shows the final textual representation of the UC describing the task Notify Resolution. Figure 8 shows the graphical UC model of the process obtained when steps 2 and 3 are finished.

Step 4. Automatic generation of a preliminary WF specification from the UCM. The switch from the UC model to the WF model is accomplished by means of an automatic transformation. This transformation is driven by the set of rules derived from the equivalencies between UC and WF concepts plus the set of process patterns shown in section 5. The output of this step is a WF specification which include activities, subprocesses, data flow, an organizational model and part of the control flow between activities and/or subprocesses.

Figure 9 shows the global process and figure 10 shows the notify resolution subprocess obtained after applying the process patterns.

Step 5. Refinement of the WF specification. The lack of a precedence relationship between UCs hinders the generation of all the control flow aspects; hence, some of the WF activities must be manually connected in order to complete the WF specification. To perform this task, the editing facilities of the WFMS are used. At the end of the process, a WF specification representing the BP is available. Figure 11 shows the WF model obtained for the loan process.

The WF specification obtained can be animated in a prototyping environment and validated with the employees of the company [11]. The final WF implementation is

achieved by defining the execution environment infrastructure: host where applications will execute, databases, legacy systems and other resources. But all these details are out of the scope of this paper.

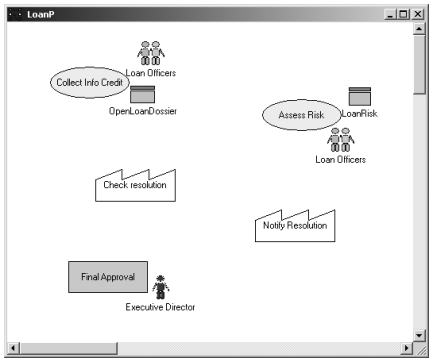

Figure 9. Global loan process

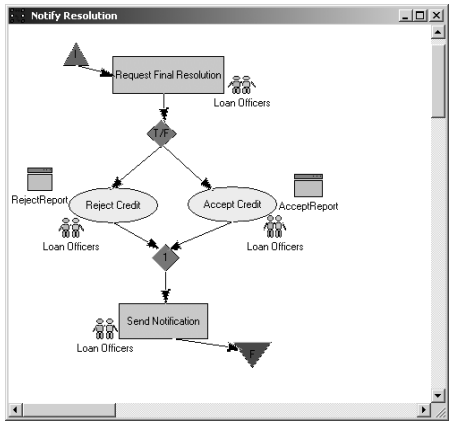

Figure 10. Notify Resolution subprocess

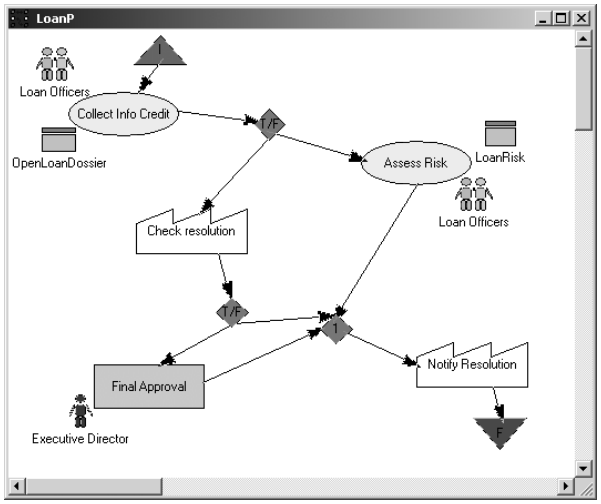

Figure 11. Loan Process WF model

7. Implementation of the Method.

We have developed a tool implementing the strategy described in previous sections. The shaded area in figure 12 shows the architecture of the tool. A UC editor is used to define the UC model corresponding to a BP in both graphical and textual representations. The model is stored in a UC repository. The UC-WF converter reads

it and generates the corresponding WF specification, which is stored in a WF repository. A WF editor is then used to make further refinements to the WF specification.

The tool was implemented using the *Borland Delphi* development environment [12] and data repositories were implemented in a relational DBMS. The tool was integrated in a WF development environment in which graphical WF models are transformed into formal WF specifications in OASIS [13], a formal object-oriented language based on dynamic logic. OASIS specifications can be prototyped using KAOS, a deductive and object-oriented database system implementing the OASIS operational semantics [11]. They can also be transformed automatically into WF specifications written in the language of efficient process engines. Currently, we are working on the generation of WF models which are executable in IBM MQSeries WF [14], SAP R/3 [15] and OPERA [16].

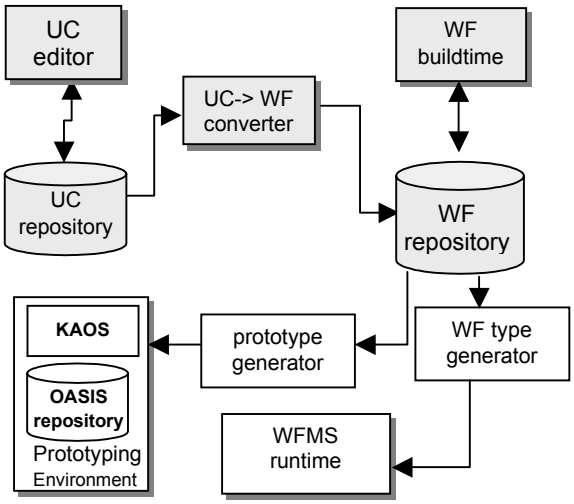

Figure 12. Tool architecture

8. Conclusions and Future Work

In this paper, we have introduced a requirements engineering layer on top of WFMSs that permits the description of business processes as use case models, which are automatically transformed into workflow specifications. The use case model is obtained in a bottom-up and iterative process which is driven by the organizational structure involved in the business process. Later, the equivalencies between use case and workflow models are used to apply the transformations leading to the implementation of the business process in an executable workflow language. A set of process patterns is used to transform relationships between use cases into control flow relationships between activities in a workflow.

Despite the growing interest of the Requirements Engineering community in organizational matters, to our knowledge there are not many proposals specifically

oriented to obtain correct and complete WF specifications. Castro et al., for instance, use the i* framework to model early requirements which can be transformed into pUML specifications through the application of a set of guidelines [17]. The transformation is essentially structure-oriented (it produces a context class diagram plus a set of constraints in OCL) and it is not evident how a process specification could be derived from the i* models. However, a motivating issue for further exploration is how dependencies between actors in an i* Strategic Dependence Model could be used in our approach to drive the BP discovery process.

Process discovery is a essentially cooperative activity [18]. Hence, some infrastructure supporting cooperation between organization members and BP modelers would be very helpful. Machado et al. [19] define a infrastructure supporting the different steps composing a domain engineering process. In particular, CSCW tools are used for cooperative scenario creation during the knowlegde acquisition stage of the process. Currently, our tool lacks any support for the cooperative elicitation of BP requirements, and how to add such capabilities to our prototype is under study.

The closely approach to ours is that of Baresi et al. [20]. In their methodological proposal to WF development, they introduce an analysis phase in which they describe the BP operational structure using UC along with sequential diagrams. In contrast to our proposal, the UC model obtained is not used to obtain a WF model, rather it is used to determine which processes should be included in the WF model, which should then be built from scratch. As we have described in this paper, our approach goes beyond the mere detection of processes, yielding a WF model lacking only the definition of the control flow between activities and/or subprocesses automatically.

Other authors have detected the lack of control flow between UCs in the UC model. To overcome it, Leite et al. define in [21] a partial ordering between scenarios that permits a limited specification of control flow. Their idea could be applied to our method since the scenarios are described by means of textual templates very similar to those we use to describe extended UCs. However, WF models include richer control flow than the offered by the Leite's proposal. We are currently working on the improvement of the method in order to automatically obtain a workflow model which also includes control flow aspects that are not generated in the current version; this should require the definition of some kind of time-ordering relationships between UC.

Another proposal close to ours is Jacobson's [22], who uses a UC model for BP reengineering activities on already existing processes in an organization. However, he does not use a WF model as the final result of the reengineering process.

We also plan to extend the tool we have implemented in order to use XML [23] to store the generated workflow models in a format which is compliant with the WfMC interchange standards [24].

References

[1] Workflow Management Coalition, "Terminology & Glossary". Technical report WfMC-TC-1011, WfMC, June, 1996. (http://www.wfmc.org/).

[2] Sheth, A., Georgakopoulos, D., Joosten, S., Rusinkiewicz, M., Scacchi, W., Wileden, J. and Wolf., A. "NSF Workshop on workflow and Process Automation in Information Systems",

Technical Report, UGA-CS-TR-96-003, Computer Science Department University of Georgia, October 1996. (Available at http://lsdis.cs.uga.edu/activities/NSF-workflow/final-report.ps).
[3] Hollingsworth, D., "The Workflow Reference Model", Technical Report TC00-1003, WfMC, December, 1994. (http://www.wfmc.org/)
[4] Canós, J.H., Penadés, M.C., Carsí, J.A. "From Software Processes to Workflow Processes: the Workflow Lifecycle", Proc. of the International Software Technology Workshop, Grenoble, France, 1999.
[5] Booch, G., Rumbaugh, J., Jacobson, I., "The Unified Modeling Language". Addison-Wesley, 1999.
[6] Leymann,F., Roller, D., "Production Workflow. Concepts and Techniques", Prentice Hall, 2000.
[7] Coleman,D., et al, "Object Oriented Development:The Fusion Method", Prentice Hall 1994.
[8] Awad, M., Kuusela J., Ziegler, J., "Object Oriented Technology for Real-Time Systems: A practical approach using OMT and Fusion". Prentice Hall 1996.
[9] Jacobson, I., Booch, G., Rumbaugh, J., "The Unified Software Development Process". Addison-Wesley, 1999.
[10] Larman C., "UML and Patterns". Addison Wesley, 1998.
[11] Canós, J. H., Penadés, M.C., Ramos, I., Pastor, O., "A knowledge-base architecture for object societies", Proc. of the DEXA-95 Workshop, OMNIPRESS, 1995.
[12] "Borland Delphi". http:\\www.inprise.com
[13] Pastor,O., Ramos,I., Canós,J.H. "Oasis v2: A Class Definition Language"; Proc. of DEXA-95, Lecture Notes in Computer Science (978), pags: 79-91 Springer-Verlag, 1995.
[14] IBM MQSeries Workflow (Available at http://www.redbooks.ibm.com)
[15] SAP AG. http:\\www.sap.com
[16] C.J. Hagen. "A generic kernel for reliable process support". Ph. D. Thesis, ETH Nr. 13114. ETH Zurich, 1999 (http://www.inf.ethz.ch/department/IS/iks/research/opera.html)
[17] Castro, J., Alencar, F., Filho, G., Mylopoulos, J., "Integrating organizational requirements and object oriented modeling", Proceedings of 5th IEEE International Symposium on Requirements Engineering (RE'01), Toronto (Canada), August 2001.
[18] Borges, M., Mendes, R., Cerqueira, B., "Bridging the gap between organizations and their software processes – An approach based on patterns and workflow systems", Technical Report, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, 2001.
[19] Machado, M., Santos, F., Werner, C., Borges, M., "Uma infra-estrutura de Apoio à Aquisiçao Cooperativa de Conhecimento em Engenharia de Domínio", Proc. XIII Simposio Brasileiro de Engenharia de Software (SBES'99), Florianopolis (Brasil), Outubro, 1999.
[20] Baresi, Castano, Ceri et al. "Wide Workflow Development Methodology". WACC'99, San Francisco, CA, USA, 1999.
[21] Leite, J.C, Hadad, G., Doorn, J.H., Kaplan, G. "A Scenario Construction Process". Requirements Engineering Vol. 5 (2000) pp 38-61. Springer-Verlag, 2000.
[22] Jacobson, I. "The Object Advantage. Business process reengineering with object technology". ACM Press, 1995.
[23] D. Box, A. Skonnard, J. Lam. "Essential XML". Addison-Wesley, 2000.

[24] Workflow Management Coalition, "Workflow Standard-Interoperability. Abstract Specification". Technical report WFMC-TC-1012, WfMC, October, 1996. (http://www.wfmc.org/).