IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4427

A Methodology for Adaptive Workflows

Liu Shuzhou, Angela Goh Eck Soong
School of Computer Engineering, Nanyang Technological University, Singapore, 639798
Email: asesgoh@ntu.edu.sg

ABSTRACT

A workflow methodology is required to analyze business logic and model workflow components. Current studies handle business process modeling and workflow modeling in an independent manner, making it hard to analyze the information system and organization requirements while modeling business logic. Moreover, current methodologies also lack support for modeling complex dynamic processes in adaptive workflows. In this paper, a methodology is proposed to unify business process modeling and workflow automation. The information system and organization perspectives of workflow are identified in each development phase. An analysis of the domain and an object notation in the methodology can help to build a stable system.
Keywords: workflow, change control, methodology

1. INTRODUCTION

Workflow provides a way to automate business processes. A workflow methodology is important for the analysis of business logic, and to model workflow components. Many commercial products provide guidelines for workflow design. These guidelines can be regarded as methodologies[1]. However, they are based on specific workflow products. A general methodology can facilitate heterogeneous workflow application. Unfortunately, there are only a few general approaches for analysis and design, such as WIDE[2] and Derung's approach[3].

WIDE[2] proposes a methodology for supporting workflow design, which is divided into three phases: workflow analysis, workflow design and mapping to target workflow systems. The workflow analysis starts from existing well-defined business processes, goals and external Information systems. The business process goals and characteristics are analyzed to determine the "workflowability" of the proposed business processes. In the design phase, the normal process flow is decomposed into sub-processes, super-tasks, business transactions, and tasks. A pattern-based approach is adopted for specifying typical exceptions to the normal flows. In the mapping phase, the results of design are mapped onto commercial workflow products and standard workflow models.

Derungs et al [3] propose a methodology to support the transformation of business processes into workflow applications. The methodology contains three central steps: Requirement Specifications, Conceptual Design and Realization. The Requirement Specification investigates the data and function requirement gaps between the process design and the As-Is-System, and then identifies the relevant workflow. Following that, the requirements and a plan for the infrastructure are determined. The Conceptual Design identifies activities, assigns them to work steps and describes the activities for the program design from the business viewpoint. The integration mechanism and the dialog for the user will be determined in this phase. The organization structure is designed and implemented too.

Michael Amberge[4] proposes a two-stage modeling procedure for the development of workflow relevant application. The first-stage uses business process modeling to identify the relevant business tasks to be supported. The second stage uses workflow modeling to specify in detail the domain-related requirements for workflow application. However, no further details on notation and step-wise guidance are presented.

In these studies, business process modeling and workflow modeling are separated. Workflow methodology is used to automate a well-defined process. This is also true for most workflow products. For example, ARIS[5] is a process modeling tool and FlowMark[6] is a workflow modeling tool. An ARIS-to-FlowMark interface[7] is used to integrate them. However, this separation results in several drawbacks. Firstly, since the process model and workflow model use different notations and tools, integration can be costly. Secondly, workflow automation can affect business logic. For example, a traditional commerce payment process is different from e-commerce payment process, since the latter has to consider network security problems. A well-defined business process may have to be changed when workflow is automated. It would be better to identify such changes as early as possible. Thirdly, workflow systems usually focus on hardware/software integration, resource integration and organization coordination. On the other hand, business processes only emphasizes on functional requirements of the system, and does not cover all aspects of the workflow. Loss of requirement analysis in organization and information systems may result in project failure or system re-design.

It is expected that adaptive workflow should support complex dynamic processes, and that process definition should be allowed to change accordingly[8]. Exceptions can be used to define error and unexpected conditions, as seen in WIDE[2]. However, workflow change is not limited to exception handling. Sometimes, the main business logic may be changed. Such changes require rebuilding business logic through re-analysis and re-design. An adaptive workflow methodology should be able to identify, trace and manage these changes.

To resolve these problems, this paper presents a methodology to unify business process modeling and workflow automation. Workflow processes, information systems and organizational policies will be analyzed and designed. In section 2, an overview of the approach will be described. Following that, the details of each phase are presented.

2. OVERVIEW OF THE APPROACH

The methodology contains four phases: domain analysis, business analysis, workflow design and workflow construction. The framework is shown in Figure 1. A domain model can be used to define the common features and variation in a specific domain [9]. The *domain analysis* defines the basic concepts and functions of similar applications. In addition, current technologies and organization responsibilities in the domain are identified. The *business analysis* identifies specific requirements of the system. Common functional requirements can be derived from the domain model by making use of domain knowledge. Application-specific require-

This paper appears in *Managing Information Technology in a Global Economy*, the proceedings of the Information Resources Management Association International Conference.

ments need to be specified separately. The non-functional requirements, such as cost, time and customer satisfaction, will determine choice of technologies. In *workflow design*, the workflow process will be decomposed to sub-processes. Roles and technologies to support sub-processes execution are identified too. These workflow components are organized into objects and relationships among them are identified. In *workflow implementation*, workflow components are mapped to specific systems for execution.

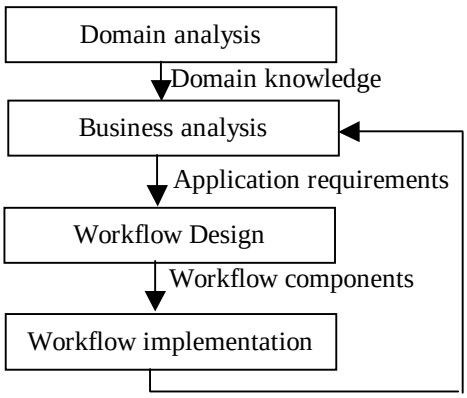

Figure 1. The process of workflow methodology

Applying information technology should not just result in doing 'original work faster', but also help to improve business process[10]. The methodology can support continuous improvement of business process since workflow process, information technology and organization policies are analyzed and designed simultaneously. The effect of information systems and organization to workflow process is identified in the early phase of development. In contrast with other approaches, there is no gap between business process and workflow process.

The development process is iterative and workflow process can be defined in an evolutionary way to respond to change. For a new application, the domain analysis should be carried out at the beginning. A changed workflow process can re-use existing domain knowledge to facilitate workflow analysis, and the variation in domain model can help to identify the components that require change. In the workflow design phase, the workflow model is organized as objects since the object-oriented paradigm can help to construct a stable system and facilitate change. As UML[11] is the de facto industry standard modeling notation for Object Oriented development, some UML notations will be adopted in the methodology.

Details of the methodology will be described through a library management system example.

3. DOMAIN ANALYSIS

Domain analysis is used to model common features and variations in a particular application domain. As a first step, domain experts will survey existing systems and find common features among them. The system scope will be identified. The basic functions of the system are described. For the library management system, the basic functions can include loan and return of books. Besides that, similar to FODA (Feature-Oriented Domain Analysis)[9], alternative and optional functions can be specified. They can be used to describe variation of the domain. Some library management systems can support reservation of books, and so, this is described as optional function. The querying operation is also the basic function of systems, but since queries may be invoked by staff or readers, querying is defined as alternative function. These functions are described in Use Case Diagrams and are shown in Figure 2.

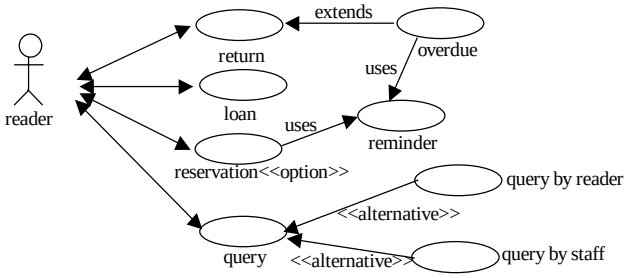

Figure 2. Use Case of library domain

The Use Case diagrams is extended with stereotypes << option >> and <<alternative>> to describe option and alternative functions. Here the 'reservation' Use Case is optional since some systems do not provide this function. 'query' function can be defined in two ways: 'query by reader' and 'query by staff', and hence, it is described as an 'alternative'. The 'overdue' feature is seen as an exception and described as extended Use Case. Both 'overdue' and 'reservation' will invoke reminders to the reader, so the 'reminder' is described as a common Use Case for them.

After analysis of the functional aspects, the information technology support for each Use Case should be identified. The domain experts survey the current information technology application in the domain, and then, for each Use Case, possible information technologies to support it will be identified. For each alternative Use Case, related information technologies are also specified. This is described in a function/technology matrix. Figure 3 shows the matrix for the library management system.

Use Case	choice	Database	Internet	Intranet	Self Check machine
return		+			
loan	Loan at counter	+			
	Loan at machine	+		+	+
query	Query by reader	+		+	
	Query by staff	+			

Figure 3. function/technology matrix

In the matrix, the technologies that can support each Use Case are identified by '+' sign in the form. Different technologies can result in different Use Cases. For example, loaning a book at a counter is different from loaning a book at a self-check machine. A choice column is used to specify such differences. For simplication, only three Use Cases are listed in Figure 3.

In Use Case diagrams, external actors are described and there is no description for internal actors. Since the organization structure can vary significantly, it is difficult to specify a common internal organization structure in the domain analysis. However, it would be better to identify responsibilities of automation operations (which is related to information systems) and manual operations (which is related to organization members) since they are executed by different workflow components. The Use Case description is extended to describe the responsibilities. For example, the responsibilities of 'return' can be:

"The reader sends the book to a staff. The staff clears the loan information by amending the database that stores the loan information".

Domain models define common features for a group of applications and are more stable than a specific application. However, the domain model can still be changed. For example, a new

technology or a new service may emerge. The extension of a Use Case diagram can help to define domain changes. An exception can be defined as an extended Use Case. A new function can be added by using the <<alternative>> stereotype. Finally, a function can be labeled as an <<option>> when it is no longer treated as a basic function.

4. BUSINESS ANALYSIS

The domain analysis identifies the common capabilities of an application. In business analysis, a specific requirement will be specified. Workflow automation is always driven by certain goals, such as improving the service of customers, decreasing cost and enhancing quality. A goal can be used to describe the reason that a system is needed, and requirements are used to specify how a goal should be accomplished by a proposed system[12]. Goals that are specified explicitly can help to identify workflow requirements.

In our approach, the goal-driven approach will be adopted. The goals are identified before the requirements are derived. The priorities of goals can be specified. There are three levels of priorities: mandatory, expected and optional. The mandatory goals define basic function and performance aims that must be attained. The goals stated as 'expected' should be attained as far as possible. Finally, the optional goals may be attained but not deemed necessary. The analyst will determine the goals from the library staff and the users. The domain model can also help to identify goals. The basic functions in domain, such as 'return book', 'loan book', must be achieved and defined as mandatory goals. The analyst chooses which alternative function will be achieved and then define it as a mandatory goal. The optional functions can be chosen to achieve or not. They can be defined as 'mandatory', 'expected' or 'optional' goals according to specific application system requirements. In a typical library management system, the goals are as follows:

(1) return book (mandatory): Reader should return their books before they are overdue.
(2) loan book (mandatory): Readers with validate library cards can borrow books from the library.
(3) reserve book (mandatory): Reader can reserve a book if it is not available.
(4) renew book (mandatory): Reader can renew a book if no one reserved it.
(5) query book (mandatory): Reader can query a book.
(6) reader satisfaction (expected): Reader should be satisfied with library services.
(7) low daily cost(expected): The daily cost of running the whole system should be low.
(8) low construction cost (optional): The cost to build the whole system should be low.

The 'return book', 'loan book', 'reserve book', 'renew book' and 'query book' describe basic system behavior and are defined as mandatory. The 'reader satisfaction' and 'low daily cost' measure the system performance and should be attained as far as possible. The 'low construction cost' is important but not necessary, so it is defined as optional.

The goals can be related to functional requirements and non-functional requirements. The functional requirements describe what the system does and the non- functional requirements describe how the system behaves with respect to some observable attributes such as performance, reliability and reusability[13]. As discussed above, the function requirements of common goals can be derived from the domain model. The specific-goal requirements, such as 'renew book', are not defined in the domain and need to be defined here. They are also described as Use Cases, and the possible technologies required to support them will be identified by a function/technology matrix, as shown in Figure 4.

Use Case	choice	Database	Internet	Intranet	Self Check machine
renew book	renew at counter	+			
	renew at machine	+		+	+

Figure 4. Technology support for 'renew book'

'Reader satisfaction', 'low daily cost' and 'low construction cost' are described as system attributes and can be associated with multiple processes. These goals can be used to choose appropriate Use Cases. For each Use Case, the effect on goals is analyzed and then the appropriate Use Case is selected to attain the goals. WIDE methodology[2] uses a case/goal matrix, but it does not consider the effect of organization and external information systems, such as database, Internet and Intranet. Here the Conflict Coordination Matrix will be used. For example, for 'loan book', the Conflict Coordination Matrix is shown in Figure 5.

Use Case	choice	reader satisfaction(expected)	Low daily cost (expected)	low construction cost (option)
loan	Loan at counter	0	-	+
	Loan at machine	+	+	-

Figure 5. Conflict Coordination Matrix for goals

If a Use Case can have a positive effect on certain factor, a symbol '+' will be used; if there is no effect, a symbol '0' will be used; otherwise, a symbol '-' will be used. When ' loan at machine' is used, the loan process will be faster, and the reader satisfaction will be enhanced. The 'loan at machine' can decrease daily cost since fewer loan counter staff is needed, but it will increase the construction cost since new hardware and software will be required. As 'low construction cost' is an optional goal, the 'loan at machine' approach is adopted. Using a similar analysis, the 'online reservation' 'query by reader' and 'renew at machine' are also selected. This is described with Use Case diagrams in Figure 6.

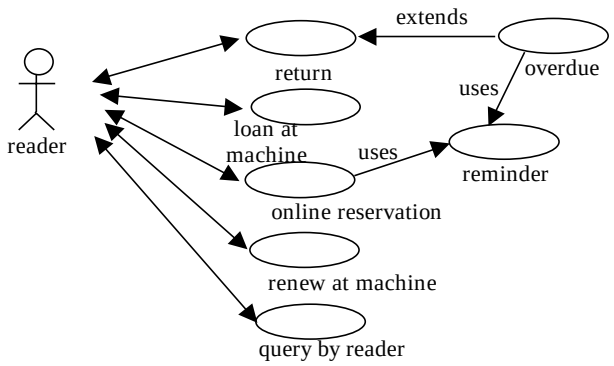

Figure 6 Use Case of library application

A textual description can be used to identify additional information for Use Case, such as how and when the Use Case begins and ends, and Use Case objectives [14]. In business analysis, Use Case version, pre/post condition, creating date and valid date etc. will be described. For example, the 'return' Use Case description is given in Figure 7.

Use case	Version	Pre condition	Post condition	Creation date	Valid date
return	1.0	The reader borrow the book	The book is returned	2000.9	2001.1-2003.1

Figure 7. Use Case description for 'return'

The version information is helpful in tracing changes to business logic. The pre condition specifies when the Use Case can be started. The post condition describes expected status after execution. The creation date and valid date specifies the duration of a Use Case.

Workflow supports cross organization boundaries business process. For each Use Case, the organization responsibilities are specified, as shown in the Use Case/Organizations matrix that is partially shown in Figure 8. The matrix specifies which department will take part in the Use Case and the responsibilities of each department. Here the responsibilities of roles are not specified, which will be attained in the further refinement.

Use Case	Office	Finance
return	accept books	accept fines
renew at machine	renew books	

Figure 8 Use Case/Organizations matrix

The results of the business analysis can experience change when user requirements or goals change. Since a domain model defines common features for a group of applications, the change of business model can be derived from domain models in a controlled manner. Firstly, a new requirement or a new goal is established, and then the variation in the Use Cases and technologies in the domain models are re-evaluated. A modified Use Case is selected to replace the original Use Case. For example, after the construction of the system, it is found that some people are not familiar with self-check machines and their satisfaction is low. As an interim solution, a few staff may be stationed at the counter to service the loans until the readers are familiar with the machines. The domain model provides 'loan at counter' choice and it will be used. The new requirement requires the coexistence of 'loan at counter' and 'loan at machine'. They are both presented in the modified Use Case diagrams. So the change can be traced by comparing differences among domain models, original models and current models. However, if a specific function is not involved in the domain model, it will be analyzed in a normal way. After the analysis, domain experts will determine whether it is necessary to add the function to the domain models or not.

5. WORKFLOW DESIGN

Business analysis specifies system requirements. The details of the whole system will be identified in workflow design. In business analysis, the function requirements are described as Use Cases. Scenarios can refine Use Cases, and activity diagrams with "swimlanes" in UML will be used to describe them. UML activity diagrams do not specify how to decompose the process. In our approach, the following criteria are provided:

(1) If a process is executed by different organization roles, the process should be decomposed until it can be executed by one organization role.
(2) If a process is executed through information systems, the information system services will be described as separate activities.
(3) If there is information passing among organization members or information systems in a process, the process should be decomposed.
(4) If a process execution can change the control flow, it should be decomposed to specify the choice, split and joined topologically.

As an example, the 'return' process can be specified. When the reader returns a book, the counter staff will discharge the loan by modifying the loan record in the database. Since the process is executed through the database, the 'modify loan record' service of the database will be defined as a separate activity. The staff will check the loan status. If the loan is not overdue, the due date will be canceled. Otherwise, the reader will need to pay a fine to a cashier and receive an invoice. The activity diagram of 'return' is described in Figure 9.

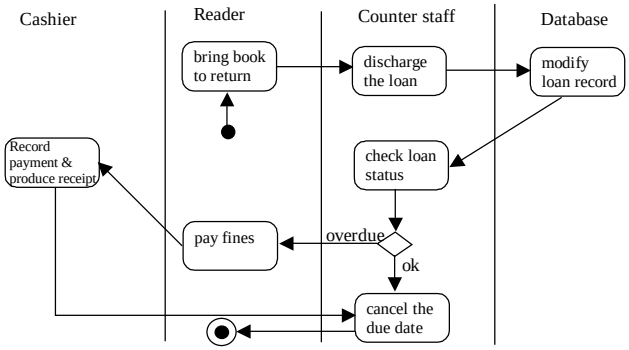

Figure 9. Activity diagrams for 'return'

The "swimlanes" group the responsibilities for organization roles and information systems. The diagrams show what services the information systems should provide, how to interact with the information system, and what data will be processed. Services provided by organization roles are identified. The information systems, workflow relevant data and organization roles can be defined as objects. A Class Diagram is used to describe them. In addition to that, the Class diagrams can also help to describe other workflow-related objects. For example, the security, audit data, workflow scheduling policy etc. Some class descriptions of the library management system are shown in Figure 10.

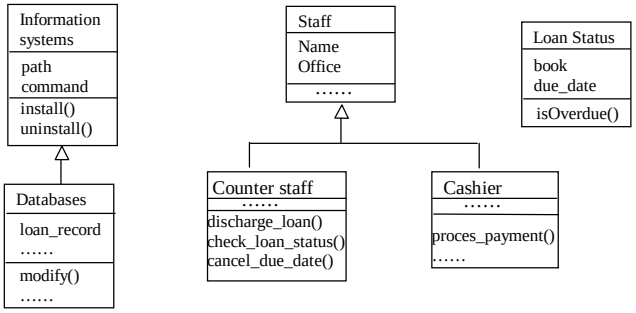

Figure 10. Class diagrams in library management system

In the class diagrams, the information systems record basic properties common to all information systems, and database management systems are defined as its sub-class. Similarly, the staff is used to the describe organization perspective, and has two sub-classes, Counter staff and Cashier. The Loan Status describes workflow relevant data. The *isOverdue()* method will be used to determine whether a book is overdue. Only the inheritance relationships are presented here. In addition, the composition relationship and substitution relationships are important too. A composi-

tion relationship can help to compose an organization element, such as department and division. The substitution relationship is helpful in role assignments: when a role is not available, another can replace his work. They can also be described by using composition and association notations in UML class diagrams.

It is well known that the object abstraction and inherit can help to construct a more stable system compared to a function-based description. Here the workflow organizations, workflow relevant data and workflow information systems are all organized as objects. Furthermore, as the process models and workflow information models are explicitly separated, changes to any one of these can be isolated and identified separately thereby decreasing the complexity of the model.

6. WORKFLOW IMPLEMENTATION

In workflow implementation, the workflow design model will be mapped to specific workflow management systems to enact the process. However, due to the wide variety of workflow products, different implementation approaches are usually adopted. It may appear that workflow design mapping has to be defined for each system since we want to provide a general workflow methodology.

Fortunately, the Workflow Management Coalition(WfMC) provides a workflow model to describe common characteristics of all workflow management systems[15]. The mapping will be done on WfMC models instead of a specific system. Common workflow models in WfMC can facilitate workflow management system integration and interoperation. Moreover, changes in the underlining workflow management systems will not affect workflow implementation models.

WfMC defines three components to help import and export workflow definitions between different workflow systems. The meta-model defines core objects in workflow process. These objects are common entities in workflow systems. The Workflow Process Definition Language (WPDL)[16] provides a formal way to define a process using the objects and attributes in the meta-model. The Workflow API(WAPI)[17] is used to manipulate process definition attributes. As the meta-model describes the workflow semantics, the workflow design specification will firstly be mapped onto the meta-model. Secondly, the WPDL will be used to describe the specification. Following that, a WfMC-complaint workflow system can access the definition with WAPI. A brief introduction to the mapping is given as follows.

The meta-model includes basic entities in a workflow process definition, their relationship and attributes. Some main entity types and their related attributes are listed in Figure 11. The mapping rules from workflow design model to WfMC meta-model are also presented.

Meta-model type	Meta-model type attributes	Workflow design model
Workflow type definition	Workflow process name	Use Case name
	Version number	Use Case version label
	Process start and termination condition	Use Case pre-condition and post-condition
	Security, audit or other control data	Workflow control objects
Activity	Activity name	Activity name in activity diagrams
	Participant assignment	Swimlanes in activity diagrams
	Automation mode	Object super class
	Other scheduling constraints	Workflow scheduling object methods
Transition conditions	Flow or execution conditions	Workflow data objects
Workflow relevant data	Data names and path	Workflow data objects attributes
	Data type	Workflow data object class
Role	Naming and organisational entity	Workflow role object attributes
Invoked application	Generic type or name	Workflow object class
	Execution parameters	Workflow object attributes
	Location of access path	Workflow object attributes

Figure 11 The mapping rules between workflow design models and WfMC models

In workflow type definition, the Use Case name is mapped to the workflow process name. The textual description of Use Case can provide version, pre-condition and post-condition information. They are mapped to WfMC meta-model directly. The security, audit or other control data can be obtained from related data object in the workflow design model.

In activity type, the activity name can be obtained from the activity diagrams. The participant assignment is determined by the swimlanes the activity belongs to in the activity diagrams. The super class of the activity executor determines the automation mode. If the super class is role, the automation mode is manual. Otherwise, it is automatic. The scheduling constraints can be derived from methods of a workflow scheduling object, which selects a preferred workflow to execute.

In workflow design, the data objects provide methods to test the transition conditions and to define execution conditions in meta-models. The workflow relevant data object attributes specifies data names and the path needed in meta-models. The data object class itself defines a data type. Similarly, roles and invoked application attributes can be derived from related objects in design models.

WPDL can be used to describe workflow models formally to facilitate workflow definition exchange. The entities in workflow design models are mapped to the language. As an example, the library management can be described by WPDL as:

```
MODEL Library_Management
        // model head
        WPDL_VERSION 1.1
        … …
        //workflow definition
        WORKFLOW return
                // workflow head
                CREATED  2000.9
                //activities in workflow
                ACTIVITIY discharge_the_loan
                        PERFORMER counter_staff
                        MOD       manual
                        // other activity attributes
                END_ACTIVITIY
                //other activities
        END_WORKFLOW
        //other workflow definition
END_MODEL
```

So far, the transition from the workflow design model to WPDL can not be done automatically. The mapping has to be done in a manual way. Future work will include a program to support automatic or semi- automatic transition.

After the workflow design model is mapped to WPDL models, a WfMC compliant workflow management system can make use of the API to access the workflow models. For example, the 'WMFetchProcessDefintion' will extract a process definition from a set of process definitions that meet the selection criteria. The'WMStartProcess' can start a specific process. The details of the API can be obtained in [17].

7. CONCLUSION

In this paper, a methodology to support adaptive workflow is proposed to unify business process modeling and workflow automation. Domain models and object models can help to construct a stable system in an adaptive workflow. A library manage-

ment system was used to explain the methodology. Future work includes formal support for the methodology and an attempt to automate the transition between workflow design and implementation.

REFERENCES

[1] Changengine Admin Edition (AdminFlow) Process Design Guide. Hewlett Packard, 1998.
[2] L. Baresi, F. Casati, et al. WIDE workflow development methodology. Proceedings of the international joint conference on Work activities coordination and collaboration. February 1999, San Francisco, CA USA.
[3] Marc Derungs, Petra Vogler, Hubert Osterle. From BPR models to workflow applications. in Workflow Handbook 1997: workflow standard. John Wiley & Sons Ltd. 1997
[4] Michael Amberg. The Benefits of Business Process Modeling for Workflow Systems. in Workflow Handbook 1997: workflow standard. John Wiley & Sons Ltd. 1997
[5] IDS, "ARIS Toolset", in http://www.ids-scheer.com/indexp.htm, IDS Prof. Scheer Gmbh, 1997.
[6] IBM, "Programming Guide", in "IBM FlowMark Manuals", AOS Group, 1996.
[7] IDS, "ARIS FlowMark Interface" in "ARIS Toolset Manuals", Oct, 1997.
[8] Amit Sheth. NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future, Athens, GA 1996. In http://ra.cs.uga.edu/activities/NSF-workflow/final-summary.html
[9] Sodhi, Jag. Software reuse: domain analysis and design processes. McGraw-Hill, 1999.
[10] Chaffey, Dave. Groupware, workflow, and intranets : reengineering the enterprise with collaborative software. Digital Press, 1998.
[11] Grady Booch, James Rumbaugh and Ivar Jacobson. The Unified Modeling Language user guide. Addison-Wesley, 1998.
[12] Anton, A.I. Goal-based requirement analysis. Proceedings of the Second International Conference on Requirements Engineering, April 1996.
[13] Putting non-function requirements into software architecture. Ninth International Workshop on Software Specification and Design, April 1998
[14] Objective Systems SF AB. Objectory Process, 1993
[15] Peter Lawrence. Workflow handbook 1997: workflow standard. John Wiley & Sons Ltd. 1997
[16] WfMC TC-1016-P. Workflow Management Coalition Interface 1: Process Definition Interchange Process Model.
[17] WFMC-TC-1009. Workflow Management Application Programming Interface (Interface 2&3) Specification. 1998.