



A Methodology for Developing User Interfaces to Workflow Information Systems

By Josefina Guerrero Garcia

A dissertation submitted in fulfillment of the requirements
for the degree of

Doctor of Philosophy in Economics and Management Sciences
Option Information Systems
of the Université catholique de Louvain

Committee in charge:

Prof. Jean Vanderdonckt, Université catholique de Louvain, Advisor
Prof. Manuel Kolp, Université catholique de Louvain, Examiner
Prof. Michaël Petit, Facultés Univ. Notre Dame de la Paix, Examiner
Prof. Stéphane Faulkner, Facultés Univ. Notre Dame de la Paix, Examiner
Prof. Philippe Palanque, Université Paul Sabatier, Reader
Prof. Marco Winckler, Université Paul Sabatier, Reader
Prof. Philippe Chevalier, Université catholique de Louvain, President of Jury

June 2010

Abstract

Supporting business processes through the help of workflow systems is a necessary prerequisite for many companies to stay competitive. An important task is the specification of workflow, i.e. the parts of a business process that can be supported by a computer-based system.

This thesis introduces a methodology for developing user interfaces for a workflow information system in a systematic way. The methodology involves a set of models that capture the various aspects required for this purpose, a user interface description language to specify the corresponding user interface, a method to structure the usage of these models, and software support. The methodology is delineated by a set of requirements that are elicited and motivated by the state of the art and relying on a framework to model workflow. The validation of the proposed methodology is achieved by applying it over different real-world case studies belonging to different domains of human activity. The methodology provides designers with methodological guidance on how to derive user interfaces of workflow information systems from a series of models, which is unprecedented.

For this purpose, a workflow is recursively decomposed into processes that are in turn decomposed into tasks. Each task gives rise to a task model whose structure, ordering, and connection with the domain model allows a semi-automated generation of corresponding user interfaces by model-to-model transformation. Reshuffling tasks within a same process or reordering processes within a same workflow is straightforwardly propagated as a natural consequence of the mapping model used in the model-driven engineering. The various models involved in the method can be edited in a graphical editor based on Petri Nets and simulated interactively. This editor also contains a set of workflow user interface patterns that are ready to use.

To: Juan, José Manuel, Andrea, and madre.

Acknowledgements

“Agradece a la llama su luz,
pero no olvides el pie del candil que,
 constante y paciente,
 la sostiene en la sombra”.
{Thank the flame for its light,
but do not forget the lampholder
 standing in the shade
 with constancy of patience}.
Rabindranath Tagore

This thesis would never have become reality without the help and suggestions of many supportive people.

My biggest thanks go to Professor Dr. Jean Vanderdonckt, my supervisor, for his wisdom, invaluable guidance and professionalism from the beginning to the end in the course of my research. Professor Vanderdonckt has been an excellent mentor and has provided unfailing support throughout my Ph.D. course and my stay in Belgium.

I want to express my gratitude to Professors Manuel Kolp, Michaël Petit, Marco Winckler, Philippe Palanque, Stéphane Faulkner, and Phillippe Chevalier for accepting to participate to the jury of this thesis. A lot of helpful comments came from the members of my jury, who read my manuscript very carefully providing me with a lot of valuable feedback.

It should be recognized that the success of this thesis was through the cooperation of Juan Manuel González, “mi chico”, whose endless support and understanding have been profound throughout the difficult times of this Doctorate course. I am thankful to my beloved José Manuel and Andrea. I would like to extend my heartiest thanks to my mother, my brothers and sisters, all my large family, and those who left (my father, Quique, and Mary). Without your love and support I am sure that I would not have been able to achieve so much.

My gratitude also goes to my Mexican, Chilian, Romanian, and Spanish friends; to BCHI members for their fruitful sharing through our research experiences, to all members of ISYS and all others at UCL who have rendered assistance and support in one way or another to make this possible. Thanks to Dr. Jaime Muñoz Arteaga for his friendship and constant interest on creating a workgroup.

This thesis was accomplished thanks to the support of:

The CONACYT, the high level Mexican scholarship programme under contract 206774 (www.conacyt.mx).

The UsiXML Consortium.

The Information Systems Unit of Louvain School of Management of Université catholique de Louvain.

The ITEA2 Call 3 UsiXML Project under contract #2008026.

The Human Project (Model-based Analysis of Human Errors during Aircraft Cockpit System Design), supported by the 7th Framework Programme of the European Commission, under contract FP7-AAT-2007-RTD-1 (www.human.aero).

The SIMILAR network of excellence, the European research task force creating human-machine interfaces similar to human-human communication, supported by the 6th Framework Program of the European Commission, under contract FP6-IST1-2003-507609 (www.simila.cc).

Table of Contents

ABSTRACT

TABLE OF FIGURES

TABLE OF TABLES

CHAPTER 1	INTRODUCTION	1
1.1	CONTEXT	1
1.2	CONCERNS	2
1.3	THESIS	4
1.3.1	<i>Thesis statement</i>	4
1.3.2	<i>Focus and scope</i>	4
1.3.2.a	Workflow management coalition's standardization work	8
1.3.2.b	User Interfaces for Information Systems	9
1.4	SOME DEFINITIONS	13
1.5	READING MAP	14
CHAPTER 2	STATE OF THE ART	17
2.1	CONTEXT	17
2.2	USER INTERFACE DESCRIPTION LANGUAGE	17
2.3	ORGANIZATIONAL MODELING	23
2.4	WORKFLOW	24
2.5	TASK MODEL	27
2.6	COMPUTER-SUPPORTED COOPERATIVE WORK	29
2.7	CHAPTER SUMMARY	36
2.7.1	<i>Shortcomings</i>	38
2.7.2	<i>Requirements</i>	39
CHAPTER 3	CONCEPTUAL MODEL FOR WORKFLOW INFORMATION SYSTEMS	43
3.1	CONTEXT	43
3.2	SEMANTICS: META-MODELING FOR WORKFLOW INFORMATION SYSTEMS	44
3.2.1	<i>Workflow model</i>	46
3.2.2	<i>Process model</i>	46
3.2.3	<i>Task model</i>	48
3.2.3.a	Existing task model	49
3.2.3.b	Extended task model	50
3.2.4	<i>Organizational model</i>	51

3.2.5	<i>Mapping model</i>	55
3.2.5.a	Existing mapping model	55
3.2.5.b	Extended mapping model	56
3.3	SYNTAX	60
3.4	STYLISTICS	62
3.5	UIDL FOR THE CONCEPTUAL MODEL	64
3.6	CHAPTER SUMMARY	66
CHAPTER 4 DEVELOPING WORKFLOW USER INTERFACES		69
4.1	CONTEXT	69
4.2	METHOD FOR SPECIFYING WORKFLOW INFORMATION SYSTEMS	70
4.2.1	<i>Workflow Information systems: Elicitation from Scenarios</i>	72
4.2.1.a	Identification Criteria	73
4.2.2	<i>Workflow Information System Design</i>	77
4.2.2.a	<i>What to do?</i> Processes specification	79
4.2.2.b	<i>How to do it?</i> Tasks specification	80
4.2.2.c	<i>Where to do it?</i> Organizational units specification	81
4.2.2.d	<i>Who will carry out it?</i> Job and user specification	81
4.2.2.e	<i>Whom?</i> Applying workflow resource patterns	82
4.2.3	<i>Workflow Information Systems Implementation</i>	88
4.2.3.a	User Interfaces for Task Models	89
4.2.3.b	Workflow User Interface Patterns	96
4.2.3.c	User interface flow	98
4.2.3.d	Error recovery	100
4.3	CHAPTER SUMMARY	101
CHAPTER 5 SOFTWARE FOR SUPPORTING THE METHOD		103
5.1	CONTEXT	103
5.2	WORKFLOW TOOLS	104
5.2.1	<i>Elicitation from textual scenario</i>	105
5.2.2	<i>Task identification</i>	106
5.2.3	<i>Resource specification</i>	106
5.2.4	<i>Workflow modeling</i>	107
5.2.5	<i>Managing resources</i>	109
5.2.6	<i>Task model editor</i>	110
5.2.7	<i>Managing files</i>	112
5.3	WORKFLOW MANAGER	113
5.3.1	<i>List of cases</i>	113
5.3.2	<i>Agenda and workflow list</i>	115

5.3.3	<i>User Interface Flow</i> -----	117
5.4	USIXML SOFTWARE FAMILY	118
5.5	CHAPTER SUMMARY	119
CHAPTER 6 VALIDATION -----		121
6.1	CONTEXT	121
6.2	EXTERNAL VALIDATION	122
6.2.1	<i>Case studies</i> -----	122
6.2.1.a	Case study 1. <i>Order personalised compression stockings over Internet.</i> -----	128
6.2.1.b	Case study 2. <i>Requesting a credit to buy a car.</i> -----	149
6.3	INTERNAL VALIDATION.....	170
6.3.1	<i>Requirements validation</i> -----	170
6.3.2	<i>Evaluation of the graphical notation</i> -----	173
6.4	CHAPTER SUMMARY	180
6.4.1	<i>Conclusion from internal validation</i> -----	180
6.4.2	<i>Conclusion from external validation</i> -----	180
CHAPTER 7 CONCLUSION -----		183
7.1	CONTEXT	183
7.2	SUMMARY OF CONTRIBUTIONS	183
7.2.1	<i>Theoretical and conceptual contributions</i> -----	184
7.2.2	<i>Methodological contributions</i> -----	185
7.2.3	<i>Software developed</i> -----	185
7.3	FUTURE WORK	185
7.4	REMARKS	186
REFERENCES -----		189
APPENDIX A -----		211
APPENDIX B -----		231
APPENDIX C -----		239
APPENDIX D -----		245
APPENDIX E -----		259
APPENDIX F -----		279
APPENDIX G -----		283

Table of Figures

FIGURE 1-1. OVERVIEW OF CURRENT APPROACHES.	2
FIGURE 1-2. TOGETHER WORKFLOW EDITOR.	3
FIGURE 1-3. SCHEMA FOR DESIGNING SCIENCE RESEARCH (ADAPTED FROM [HEVN07]).	5
FIGURE 1-4. AN OVERVIEW OF THE PROPOSAL.	5
FIGURE 1-5. CONTRIBUTION, REUSE, AND MODIFICATION OF CONCEPTS FOR THE METHODOLOGY.	6
FIGURE 1-6. METHODOLOGICAL DOMAIN CROSSROAD.	7
FIGURE 1-7. WORKFLOW SYSTEM CHARACTERISTICS [WfMC95].	9
FIGURE 1-8. GENERIC WORKFLOW PRODUCT STRUCTURE [WfMC95].	10
FIGURE 1-9. READING MAP.	16
FIGURE 2-1. DESIGNING SCIENCE RESEARCH, RELEVANCE AND RIGOR CYCLES [HEVN07].	17
FIGURE 2-2. OVERVIEW OF ENTERPRISE ONTOLOGY [USCH98].	23
FIGURE 2-3. METHODOLOGICAL PHASES AMENITIES.	32
FIGURE 2-4. CONCEPTUAL VIEWS OF THE COOPERATIVE MODEL OF AMENITIES.	33
FIGURE 2-5. METHODOLOGICAL PHASES CIAM.	34
FIGURE 2-6. MODELS RELATIONSHIPS.	35
FIGURE 3-1. SCHEMA FOR CONCEPTUAL MODELING DERIVATION (ADAPTED FROM [HEVN07]).	44
FIGURE 3-2. SIMPLIFIED OVERVIEW OF WORKFLOW META-MODEL.	45
FIGURE 3-3. WORKFLOW MODEL.	46
FIGURE 3-4. PROCESS MODEL.	48
FIGURE 3-5. TASK MODEL.	51
FIGURE 3-6. ORGANIZATIONAL MODEL.	52
FIGURE 3-7. MAPPING MODEL.	58
FIGURE 3-8. WORKFLOW META-MODEL.	59
FIGURE 3-9. GENERATION OF USiXML SPECIFICATION.	60
FIGURE 3-10. FROM UML CLASS TO USiXML SPECIFICATION.	61
FIGURE 3-11. TRANSFORMATION OF THE AGGREGATION RELATIONSHIP INTO USiXML SPECIFICATION.	61
FIGURE 3-12. TRANSFORMATION OF THE INHERITANCE RELATIONSHIP INTO USiXML SPECIFICATION.	62
FIGURE 3-13. USiXML PROFILE BASED ON CAMELEON UIDL LAYER.	65
FIGURE 3-14. TIMELINE OF CONTRIBUTIONS TO USiXML.	67
FIGURE 4-1. SCHEMA FOR METHOD DERIVATION (ADAPTED FROM [HEVN07]).	69
FIGURE 4-2. METHOD OVERVIEW.	71
FIGURE 4-3. SOME SPEM CONCEPTS.	72
FIGURE 4-4. REQUIREMENTS ELICITATION PACKAGE.	74
FIGURE 4-5. REQUIREMENTS ELICITATION ACTIVITY DIAGRAM.	75

FIGURE 4-6. WORKFLOW INFORMATION SYSTEM DESIGN PACKAGE.....	78
FIGURE 4-7. WORKFLOW INFORMATION SYSTEM DESIGN ACTIVITY DIAGRAM.....	79
FIGURE 4-8. TASK MODEL REPRESENTATION.	80
FIGURE 4-9. TASK LIFE CYCLE.	82
FIGURE 4-10. RESOURCE ALLOCATION PATTERNS SELECTION.	83
FIGURE 4-11. IMPLEMENTATION PHASE.	88
FIGURE 4-12. IMPLEMENTATION STEPS.	89
FIGURE 4-13. CAMELEON REFERENCE FRAMEWORK AND UsiXML.	90
FIGURE 4-14. TASK MODEL.....	91
FIGURE 4-15. FROM TASK MODEL TO AUI MODEL.	93
FIGURE 4-16. FROM AUI MODEL TO CUI MODEL.	95
FIGURE 4-17. FROM CUI MODEL TO FUI.	96
FIGURE 4-18. DIRECT ALLOCATION PATTERN.	97
FIGURE 4-19. USER INTERFACES FLOW.	100
FIGURE 5-1. SCHEMA FOR SOFTWARE DERIVATION (ADAPTED FROM [HEVN07]).	103
FIGURE 5-2. SOFTWARE TOOLS.	104
FIGURE 5-3. WORKFLOW EDITOR.....	104
FIGURE 5-4. ELICITATION FROM TEXTUAL SCENARIO.	105
FIGURE 5-5. TASK TABLE.....	106
FIGURE 5-6. JOB EDITOR.	106
FIGURE 5-7. USERSTEREOTYPE EDITOR.....	107
FIGURE 5-8. WORKFLOW EDITOR.....	108
FIGURE 5-9. WORKFLOW COMPONENTS.....	109
FIGURE 5-10. MANAGING WORKFLOW RESOURCE PATTERNS.....	110
FIGURE 5-11. TASK EDITOR.	111
FIGURE 5-12. TASK PROPERTIES.	111
FIGURE 5-13. AUI MODEL.	112
FIGURE 5-14. MANAGING WORKFLOW.....	113
FIGURE 5-15. LIST OF CASES.....	114
FIGURE 5-16. UPDATE OF A LIST OF CASES PERTAINING TO A TRANSITION.....	114
FIGURE 5-17. SPECIFICATION OF THRESHOLD.	114
FIGURE 5-18. THREE POSSIBLE STATES OF A TRANSITION.	115
FIGURE 5-19. MANAGER AGENDA WINDOW.	115
FIGURE 5-20. ASSIGNMENT OF RESOURCES.....	116
FIGURE 5-21. RESOURCE DESKTOP WINDOW.....	116
FIGURE 5-22. GENERAL VIEW OF THE SYNCHRONIZATION MODEL TO SUPPORT THE COMMUNICATION BETWEEN UIS.	117
FIGURE 5-23. COMMUNICATION BETWEEN UIS.....	118

FIGURE 5-24. THE SUITE OF USiXML TOOLS STRUCTURED ACCORDING TO THE MDA CLASSIFICATION, [VAND05B].	119
FIGURE 6-1. REQUIREMENT VALIDATION SCHEMA FOR DESIGN SCIENCE RESEARCH CYCLES (ADAPTED FROM [HEVN07]).	121
FIGURE 6-2. TAKING PICTURE - TASK MODEL.	135
FIGURE 6-3. UPLOAD PICTURES - TASK MODEL.	136
FIGURE 6-4. QUALITY VERIFICATION – TASK MODEL.	136
FIGURE 6-5. SEND PICTURES – TASK MODEL.	136
FIGURE 6-6. CHOICE OF MODEL -TASK MODEL.	137
FIGURE 6-7. CALCULATE 3D MODEL - TASK MODEL.	137
FIGURE 6-8. ENTER DATA (1) - TASK MODEL.	137
FIGURE 6-9. ENTER DATA (2) - TASK MODEL.	138
FIGURE 6-10. ENTER DATA (3) - TASK MODEL.	138
FIGURE 6-11. ENTER DATA (4) - TASK MODEL.	138
FIGURE 6-12. SEARCH CUSTOMER (1) - TASK MODEL.	139
FIGURE 6-13. SEARCH CUSTOMER (2) - TASK MODEL.	139
FIGURE 6-14. SEARCH CUSTOMER (3) - TASK MODEL.	139
FIGURE 6-15. COMPARE TO CATALOGUE - TASK MODEL.	139
FIGURE 6-16. FINALIZATION OF ORDER - TASK MODEL.	140
FIGURE 6-17. PACKAGING - TASK MODEL.	140
FIGURE 6-18. ORGANIZATION CHART.	141
FIGURE 6-19. WORKFLOW.	146
FIGURE 6-20. PREPARATION OF MACHINE UI.	147
FIGURE 6-21. DYEING OF STOCKINGS UI.	147
FIGURE 6-22. PACKAGING UI.	148
FIGURE 6-23. USER INTERFACE FLOW.	148
FIGURE 6-24. ELICITATION.	153
FIGURE 6-25. TASK IDENTIFICATION.	153
FIGURE 6-26. BANK STRUCTURE.	160
FIGURE 6-27. ORGANIZATIONAL TREE.	160
FIGURE 6-28. JOB EDITOR VIEW.	160
FIGURE 6-29. PROCESS MODEL VIEW.	162
FIGURE 6-30. E-BANKING TASK MODEL.	165
FIGURE 6-31. SHOW PRODUCT INFORMATION TASK MODEL.	165
FIGURE 6-32. CREDIT SIMULATOR TASK MODEL.	166
FIGURE 6-33. REQUEST ADDITIONAL INFORMATION TASK MODEL.	167
FIGURE 6-34. BUYING USED CARS TASK MODEL.	167
FIGURE 6-35. COMPARE YOUR CAR TASK MODEL.	167
FIGURE 6-36. CREDIT TO BUY A CAR PAGE.	168

FIGURE 6-37. SIMULATE BUYING PAGE.....	169
FIGURE 6-38. PERSONAL INFORMATION PAGE.....	169
FIGURE 6-39. PERSONAL SUBJECTIVE REQUIREMENTS COVERAGE RATE.....	180

Table of Tables

TABLE 1-1 . EXPRESSION OF INTERACTION STYLES IN TERMS OF TASK PARAMETERS	12
TABLE 1-2. EXPRESSION OF INTERACTION STYLES IN TERMS OF USER PARAMETERS.....	13
TABLE 2-1. PROPERTIES COMPARISON OF UIDLS.	21
TABLE 2-2. GENERAL FEATURES OF UIDLS.....	22
TABLE 2-3. MAIN FEATURES OF TASK MODELS.	30
TABLE 2-4. COMPARISON OF TASK RELATIONSHIPS.	31
TABLE 3-1. STYLISTICS FOR INTERACTION OBJECTS.	64
TABLE 4-1. IDENTIFICATION CRITERIA.....	76
TABLE 4-2. CLASSIFICATION OF WF RESOURCE PATTERN.	85
TABLE 4-3. CONSTRAINTS TO CHOOSE OFFERING PATTERNS.....	86
TABLE 4-4. CONSTRAINTS TO CHOOSE ALLOCATION PATTERNS.....	87
TABLE 4-5. AIC FACETS.	92
TABLE 4-6. MAPPINGS BETWEEN CONCRETE AND ABSTRACT EVENTS.	95
TABLE 6-1 CASE STUDIES FEATURES	127
TABLE 6-2. RESULTS OF THE TASK IDENTIFICATION FOR THE BANK.	159
TABLE 6-3. JOBS IDENTIFICATION.....	161
TABLE 6-4. TYPE OF ACTIVITY AND FLOWiXML RELATEDNESS.	176
TABLE 6-5. CD CRITERIA AND FLOWiXML RELATEDNESS.	179
TABLE G-1. SOME MAPPINGS BETWEEN CONCRETE AND ABSTRACT EVENTS.	294

Chapter 1 Introduction

Although Information Systems (ISs) are considered vital in any organization, they do not have necessarily the expected impact on carrying out interactive and non-interactive tasks of these organizations. Many causes may explain this lack of exploitation: tasks were developed with the implicit assumption that they would primarily be performed by people, an organizational structure would be developed in which particular tasks are allocated to groups of users. Only then people consider whether computers – or rather, ISs – could partially support, or even take over, the work. This approach does not closely examine the opportunities offered by information systems. *“We have now reached a turning point: we first design business processes in a more abstract way, without considering implementation, and then we design the information systems and the organization hand in hand. In fact, we decide whether each task in a process should be performed by an information system or a person”* [vand02]. In this context, ISs may fail to meet organizational needs.

Information technology professionals understand that there is a divide between the business side, with its business requirements, and the support that is being provided to address these requirements. Looking for ways to bridge this gap, a workflow could be considered as an appropriate mean to address this need.

As a starting point we reviewed the current approaches that are the base of this thesis (Figure 1-1). There has been a growing interest in Workflow Management Systems (WfMSs) and flexible workflow support. However, when defining a workflow, software rarely supports designers in creating User Interfaces (UIs) corresponding to this workflow, i.e. the UI that helps end users to carry out their interactive work. Workflow IS (WfIS) is a concept that we introduced to define the use of workflow technology for IS development, focusing on the UI development.

1.1 Context

An *Information System* is hereby defined as “a set of interrelated components that collect (or retrieve), process, store, and distribute information” [Laud06]. Information systems are a fundamental part of most modern organizations. It is important to build successful IS, for at least two reason: 1) for an IS to be understandable for the organizations it is part of, it must take current work practices into account; 2) also because an IS determines, in certain degree, what work can be done and how it can be performed, it must be designed according to the organizations objectives and goals [Trae99].

Workflows are activities involving the coordinated execution of multiple tasks performed by different resources to achieve a common business goal. A task de-

finishes some work to be done by a person, by a software system or by both of them.

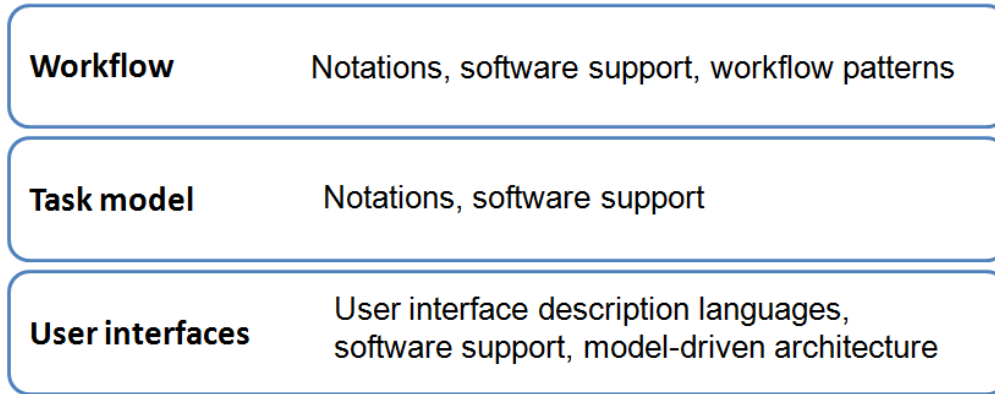


Figure 1-1. Overview of current approaches.

Specification of a workflow involves describing those aspects of its component tasks (and the resources that execute them) that are relevant to control and coordinate their execution, as well as the relations between the tasks themselves.

Information in a workflow mainly concerns when a certain task has to start, the application information needed for performing the tasks, the criteria for assigning the task to resources, and the ending of the task.

The development of workflow technology can be traced back to various origins, such as: office information systems [Brac84], computer supported cooperative work (CSCW), imaging and document management as well as advanced database technologies.

A Workflow Management System (WFMS) allows both to specify workflows and to control their execution. During a workflow execution, a WFMS has to schedule tasks (including their assignment to resources) on the basis of the (static) workflow specifications, of the (dynamic) sequence of events signaling the completion of tasks, of available data, and of generic events produced within the execution environment.

1.2 Concerns

In the context of this thesis, we identify hereafter a set of concerns that are considered important for developing workflow information systems. They represent our perception of the problem and became the first problems to be addressed in this work:

1. *Limited impact of ISs on carrying out interactive and non-interactive tasks of the organizations.* Many reasons explain this lack of exploitation: tasks were developed with the implicit assumption that they would primarily be performed by

Chapter 1. Introduction

people; an organizational structure would be developed under which groups of people were allocated particular tasks. Only then people consider whether information systems could partially support, or even take over, the work.

2. *Lack of considering organizational concepts.* Important aspects of a WfMS are the ability to handle information needed to perform the tasks and to represent the organization structure.
3. *Limited communication support for resources.* People use technologies as email and chat that are not well integrated with other resources involved in a workflow.
4. *Limited support for task assignment to resources.* Assigning a task to a resource is considered complicated due to the different levels of skills they have, e.g. experience or to do the task.
5. *Limited support for workgroup.* There is a need to explicitly include multi-user interaction during the design process of ISs.
6. *Lack of a method to developing workflow UIs.* When defining a workflow, software rarely supports designers in creating UIs corresponding to this workflow. For instance, TOGETHER enables the designer to create a workflow and to attach a UI, but this is done manually (Figure 1-2).

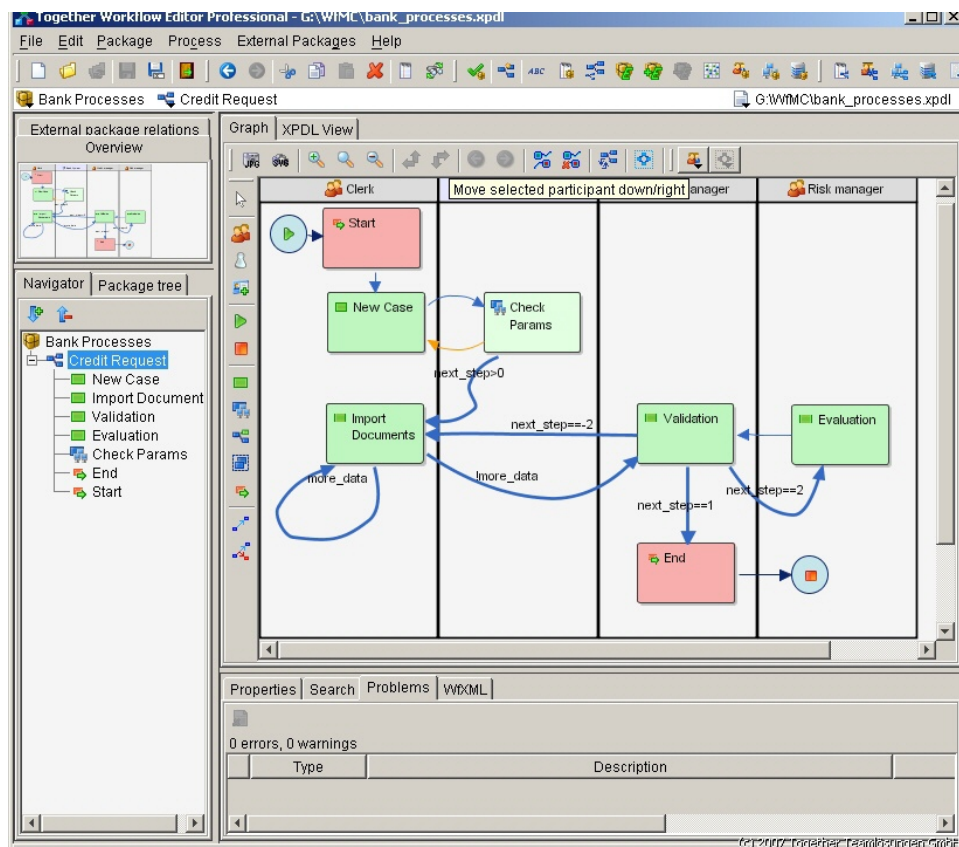


Figure 1-2. Together workflow editor.

1.3 Thesis

According to Hevner [Hevn04], the design of an information technology artifact, its formal specification, and an assessment of its utility are integral to design science in information systems research. The design science research considers the *design process* as a sequence of expert activities that produces an artifact (the *design artifact*). Taking into account the identification of two different design processes (i.e., build and evaluate) and four design artifacts (i.e., constructs, models, methods, and instantiations) [Marc95], Hevner [Hevn04, Hevn07] suggests a conceptual framework for understanding, executing, and evaluating IS research, and analyze design science research as an embodiment of three closely related cycles of activities: the relevance cycle, the rigor cycle, and the design cycle. We use, and illustrate in an evolutionary way, this framework to position our dissertation.

1.3.1 Thesis statement

In this thesis, we argue that developing workflow user interfaces is an activity that would benefit from the application of a **methodology** which is composed of:

(1) A set of **models** defined according to an ontology, (2) a constructor which provides the **language** that expresses these models, (3) a **method** manipulating these models, and (4) an instantiation (**software**) which shows that models, languages, and methods can be implemented in a working system.

Thus, we will defend the following thesis statement:

A model-based methodology expressing a workflow in terms of processes, tasks, workflow resource patterns, and organization enables structured designing of graphical user interfaces to workflow information systems based on workflow user interface patterns.

By identifying problems and opportunities (concerns) in an actual application environment (workflow), and taking into account resources available (knowledge base) it is expected to contribute to addressing the problem raised in the thesis statement (Figure 1-3).

1.3.2 Focus and scope

The current thesis basically concentrates on the following aspects:

- This dissertation introduces a method for developing user interfaces of a workflow information system in a systematic way. The method involves a set of models that capture the various aspects required for this purpose, a UI

Chapter 1. Introduction

description language to specify the corresponding UI, a methodology to structure the usage of these models, and a software support.

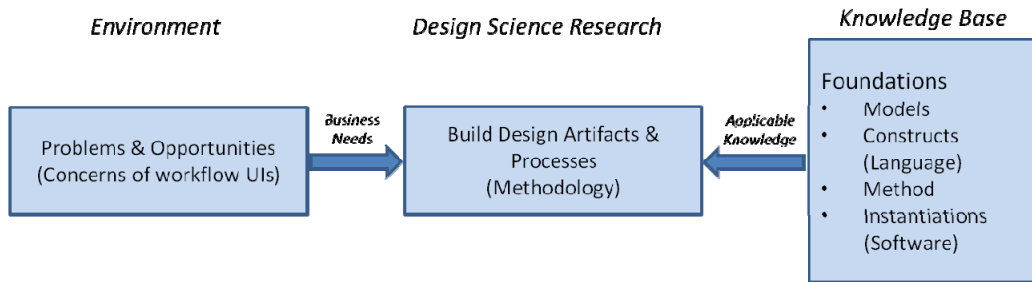


Figure 1-3. Schema for designing science research (adapted from [Hevn07]).

- For this purpose, a workflow is recursively decomposed into processes that are in turn decomposed into tasks. Each task gives rise to a task model whose structure, ordering, and connection with the domain model allows a semi-automated generation of corresponding UIs (Figure 1-4).

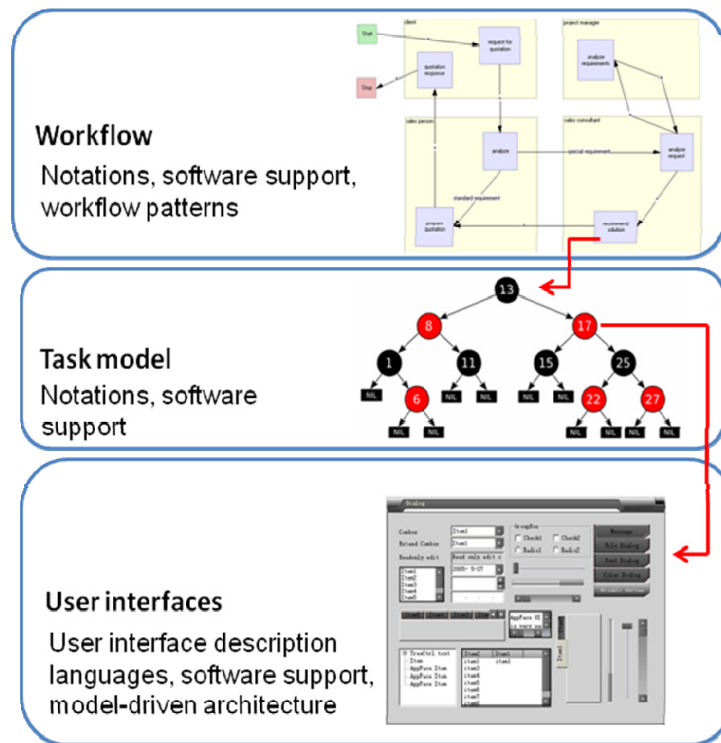


Figure 1-4. An overview of the proposal.

- The method proposed is applied to the automation of business process integrating human and machine base activities, in particular those related with information technology.

Chapter 1. Introduction

- Workflow development is also relevant to various areas of computer science such as, but not limited to: office information systems, computer supported cooperative work field, web engineering, etc. In this last area, Web Modeling Language (WebML) and WebRatio evolve from the support of online content browsing to the management of full-fledged collaborative workflow-based applications, spanning multiple individuals and organizations [Bram07]. Similarly, internet technologies like Active XML documents that are viewed as a data-oriented workflow language for specifying the Web service calls and their interactions [Subr09, Wang05]. The approach described in this dissertation departs from these other approaches in some aspects [Guer09b, Gonz09].
- The goal here is not to provide yet another model of workflows, but to identify a meta-model of concepts that are considered fundamental in order to address challenges posed by UI of these workflows. This could be achieved by identifying existing related models and consolidate them (Figure 1-5).

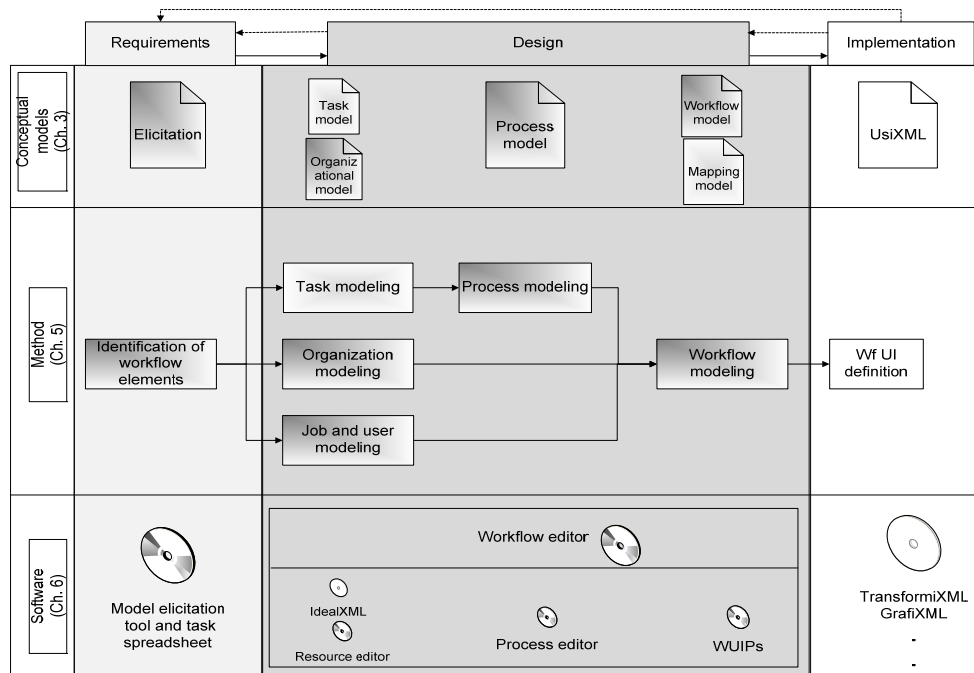


Figure 1-5. Contribution, reuse, and modification of concepts for the methodology.

- The objective of this thesis consists in defining a method that eases the designer's workload when developing workflow UIs. We rely on an existing and complete XML language. Therefore, the question of usability and accessibility of UI resulting from this methodology, although important, will not be addressed in this thesis.

Chapter 1. Introduction

- The target audience of this thesis are: the HCI research community, the professionals involved in the design and development of workflow UIs, and the professionals involved in workflow management systems. The ultimate target is the end user for whom the benefit of this UIs should become obvious.
- There are some user behaviors that could affect the workflow. Further, conflict is always possible and usually presents among participants in any social structure. Despite of this, human behavior is out of the scope of this thesis.
- Workflows may also consist of implementing business processes on document management to route documents from person to person so that they can each complete their document management tasks, such as reviewing documents, approving their publication, or managing their disposition. In order to manage an active document (e.g., an object to manipulate), first the task should be modelled for the UI that will use such document, what actions do you expect to perform over it (e.g., check in, check out, delete). The UI should enable the end user to carry out her tasks along with the active document attached. The logic of the workflow is embedded in the active document itself: when a first person has fulfilled her own task, the document is passed onto another person, resulting in activating the document part that is relevant for this person at this stage of the workflow. And so forth.
- The research environment of this thesis is composed of (Figure 1-6): Information Systems (IS), Human-Computer Interaction (HCI), and Computer-Supported Cooperative Work (CSCW). We do not intent to cover all the aspects of these areas, but to capture the relevant part of each one in order to feed our method.

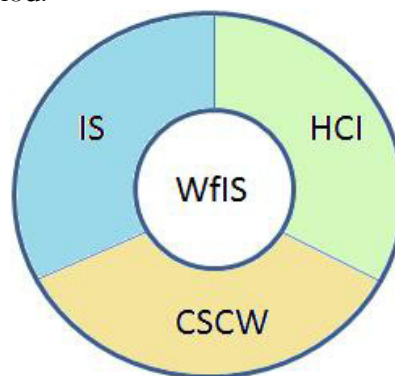


Figure 1-6. Methodological domain crossroad.

We should make it clear that even a description of a business process and the corresponding workflow specification both refer to the same set of activities and their ordering they are simply expressed at different levels of abstraction with different viewpoints. The differences are due to various objectives and perspectives.

Chapter 1. Introduction

A business process description is made by domain experts. It describes the processes from a user perspective. Activities are depicted as active bits that are executed by actors. The objective of a business process description is to provide a basis for communication. The descriptions are used for various purposes. In the everyday life of a company they serve as manuals for process participants or as learning material for newcomers. A business process model provides a basis for discussion in order to detect optimization potential. In preparation for the use of a WfMS they provide a basis for agreeing on the processes to be supported.

The business process description must be understandable for people from very different backgrounds and “knowledge cultures”, e.g. heads of departments, department staff, and Information Technology (IT) experts.

A workflow specification, in contrast, is made by IT experts. It describes the process to be supported from a monitoring perspective. Activities are no longer the active bits but are embedded within tasks. A workflow specification is used as input for a WfMS and must therefore be machine readable. The description here refines representation of behavior for subsequent monitoring. A workflow specification must be unambiguous and may not contain any uncertainties. This is a necessary requirement in order to analyze and simulate the described processes and to monitor their execution at run-time. A workflow specification also contains details that are close to implementation. Whereas it is sufficient for a business process description to cover the set of desired process executions, a workflow specification also determines how these executions are achieved. Modeling workflows, it would be good to enhance existing business process descriptions such that they can be used as inputs for a WfMS.

1.3.2.a Workflow management coalition’s standardization work

The Workflow Management Coalition (WfMC) was founded in 1993 as a non-profit international organization of workflow vendors, users and analysts. WfMC has proposed a framework for the establishment of workflow standards [WfMC95], which includes five categories of interoperability and communication standards and development of a common terminology in a glossary of workflow terms [WfMC99].

WfMC characterize a workflow management system as a system providing support in three areas: build-time functions concern with process modeling, run-time control functions concerned with the management of executing processes and run-time interactions with human users and information technology application tools (Figure 1-7).

WfMC propose a general implementation model of a workflow system which is claimed can be mapped to most products in the marketplace. The main components of the generic workflow product structure are depicted in Figure 1-8, and serves as a basis for WfMC’s development of interoperability scenarios.

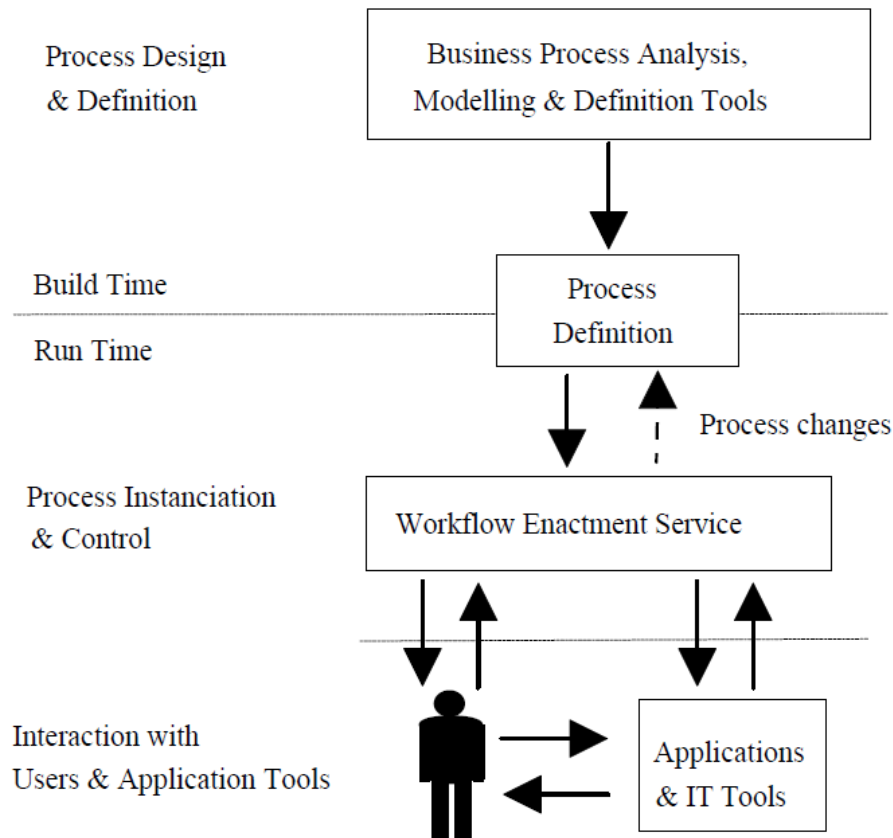


Figure 1-7. Workflow system characteristics [WfMC95].

The distribution of tasks and information among participants may use a variety of underlying communications mechanisms (electronic mail, messaging passing), then is necessary to cover UIs to support interaction with information technology application types and interfaces to allow a complete view of the status of work flowing through the organization (administration and monitoring). The UI is just considered to be a mean for handling the worklist (agenda). When defining a workflow, software rarely supports designers in developing UIs corresponding to this workflow.

1.3.2.b User interfaces for information systems

Today, the importance of ISs, in nearly all companies and organizations, is fundamental. Different types of ISs can be distinguished depending on the operational level they serve in the organization (i.e., strategic, management or operational level) or their major functional area (e.g., sales and marketing, manufacturing and production, finance and accounting, and human resources). An IS constitutes a construction of [Boda89]:

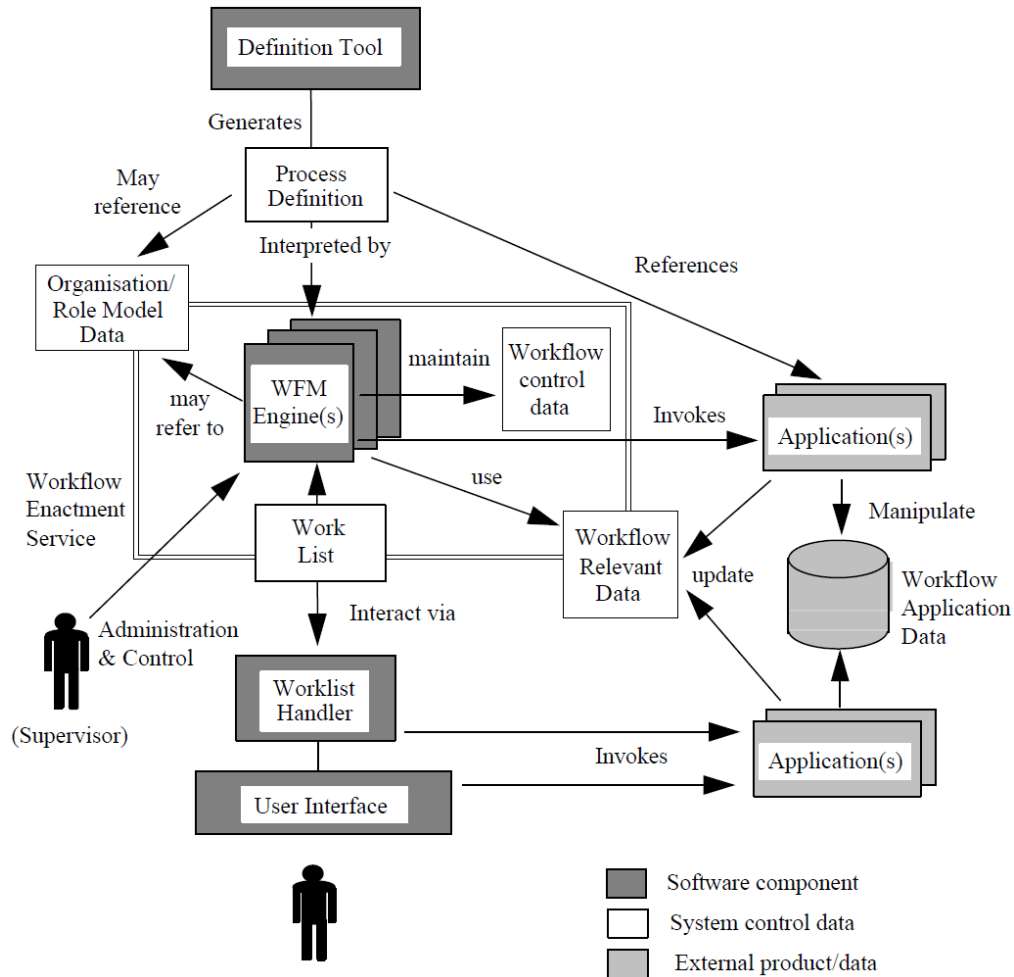


Figure 1-8. Generic workflow product structure [WfMC95].

- *Data*, a partial representation of facts that interest the organization.
- *Processes*, that represent means to acquire, search, store, present, and convey information.
- *Organization rules*, governing the implementation of informational treatments.
- *Human resources and technologies* required for the functioning of ISs.

An IS supports *management tasks* such as those depicted in a classic typology according to [Boda89]:

- *Functioning level*, which ranges from operational, decisional to strategic.
- *Structure level*, which ranges from structured to informal.

In this thesis, we primarily consider tasks that are operational and structured activities defined to deal with routine activities [Boda89]. In this scenario, a context

Chapter 1. Introduction

of use is assumed to be *quasi-constant*: the physical environment is assumed to be an office setup; the user has known skills required conducting these tasks, and a desktop computer is considered as the main computing platform.

Assuming tasks and the context of use, it is crucial to select an *interaction style* that is appropriate. For this purpose, Table 1-1 and Table 1-2 consider interaction styles according to task attributes, respectively to user attributes.

The focus on ISs induces the following values for these task properties:

- *Minimal to maximal prerequisites*, as the amount of knowledge required to the user to properly carry out the task with the intended UI varies. For instance, the prerequisites of an ATM should be minimal, whereas the UI for an air-traffic control system would surely be maximal.
- *Low to high productivity*, as the frequency of use varies depending on the task. For instance, a letter composition in an insurance company is of high productivity for insurance producers, whereas a monthly report is not.
- *Existent objective task environment*, whether an organizational task assumes the presence of domain objects.
- *Feasible environment reproductibility*, it is useful to represent domain objects as handling objects.
- *Low to high task structure*, as the degrees of freedom or constraints that the user has in carrying out the task. For instance, calculating the roots of a second-degree equation is highly structured since a deterministic algorithm governs the process, whereas an advice-giving task for loans may reorder subtasks according to currently available information.
- *Low to high task importance*, whether a task in the organization may be crucial or not. For instance, setting up an alarm in a control room is considered important, whereas editing a simple statistical report is not.
- *Low to high task complexity*, as the complexity degree of a task varies. For instance, a radar-tracking task is highly complex, whereas an advertisement composition is not.

Interaction style	Prerequisites	Productivity	Objective environment	Reproductibility	Task structuring	Task importance	Task complexity
Command language	moderate	high	non existent	unfeasible	low	high	low to moderate
Programming language	maximal	low	non existent	unfeasible	low	low	moderate to high
Natural language	minimal	low	non existent	unfeasible	low	low	low to moderate

Chapter 1. Introduction

Function keys	minimal	high	non existent	unfeasible	low to moderate	moderate	low to moderate
Menu selection	minimal	moderate	non existent	unfeasible	moderate to high	low	moderate
Query language	moderate	moderate	non existent	unfeasible	low	low	low
Questions/ Answers	minimal	low	existent	feasible	high	low	low
Form filling	moderate	moderate	existent	feasible	high	high	moderate
Multi windowing	moderate	moderate	existent	feasible	low to moderate	high	high
Direct manipulation	minimal to maximal	moderate	existent	feasible	low	high	high
Iconic interaction	moderate	high	existent	feasible	moderate	moderate	low
Graphic interaction	moderate to maximal	moderate	existent	feasible	low	low to moderate	low to moderate
Multimedia interaction	minimal	low	existent	feasible	moderate	low to moderate	moderate to high

Table 1-1 . Expression of interaction styles in terms of task parameters [Vand00a].

The focus on ISs induces the following values for the user properties:

- *Task experience* (elementary, regular, rich): this parameter combines syntactic and semantic task knowledge. *Syntactic knowledge* refers to the task allocation and its position in the complete chain, including terminology, whereas *semantic knowledge* refers to domain objects, actions and procedures embedded in the task. If a user integrated these from both an intellectual and practical point of view, then the task experience is said rich.
- *System experience* (elementary, regular, rich): this parameter expresses the experience level required by technological means in order to carry out the task, such as printer facilities, file management, and word processing.
- *Task motivation* (low, moderate, high): this parameter translates the psychological user attitude with respect to the task. If the user is eager to carry out a task, her motivation is high. A constrained user has a low value.
- *Experience with modern interaction devices* (low, moderate, high): this parameter reflects how a user is able to use modern interaction device one at a time or several ones simultaneously. For instance, gesture recognition devices are expected to require some substantive experience from users.

Chapter 1. Introduction

Interaction style	Task experience	System experience	Task motivation	Experience with modern interaction devices
Command Language	moderate to rich	rich	high	moderate
Programming language	rich	rich	rich	moderate
Natural Language	rich	moderate	low	high
Query language	rich	moderate	moderate	high
Questions/Answers	elementary	elementary to moderate	low	moderate to rich
Function keys	moderate to rich	elementary	low	low
Menu selection	elementary	elementary	low	low
Form filling	elementary to rich	elementary to rich	low to moderate	elementary to moderate
Multi-windowing	elementary	elementary	low	low
Direct manipulation	elementary	moderate	low	low
Iconic interaction	elementary to moderate	moderate	low to moderate	low
Graphic interaction	elementary	moderate	low to moderate	moderate
Multimedia interaction	elementary	moderate	low	moderate

Table 1-2. Expression of interaction styles in terms of user parameters [Vand00a].

1.4 Some definitions

Information systems users interact through a UI for pursuing organizational goals. The *User Interface* is the aggregation of means by which people (the users) interact with a particular machine, device, computer program or other complex tool (the system). The UI provides means of: *input*, allowing the users to manipulate a system; and *output*, allowing the system to produce the effects of the users' manipulation [Limb04a].

Model-based interface development is a paradigm for developing interfaces based on constructing a declarative description of how an interface should look and behave (model), and using the description to control the UI execution [Puer97].

Model-Driven Architecture (MDA). The following definition was approved unanimously by 17 participants of the ORMSC plenary session meeting in Montreal on 23-26 August 2004. The stated purpose of these two paragraphs was to provide principles to be followed in the revision of the MDA guide:

Chapter 1. Introduction

"MDA is an OMG initiative that proposes to define a set of non-proprietary standards that will specify interoperable technologies with which to realize model-driven development with automated transformations. Not all of these technologies will directly concern the transformation involved in MDA. MDA does not necessarily rely on the UML, but, as a specialized kind of MDD (Model Driven Development); MDA necessarily involves the use of model(s) in development, which entails that at least one modelling language must be used. Any modelling language used in MDA must be described in terms of the MOF language to enable the metadata to be understood in a standard manner, which is a precondition for any activity to perform automated transformation."

Human Computer Interaction (HCI). ACM defines Human Computer Interaction (HCI) as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them". This definition attempts to cover the wide range of topics HCI covers, including disciplines: psychology, sociology. One of the concerns in this discipline is the development of User Interfaces (UIs).

Management Information Systems is a general name for the academic discipline covering the application of information technology to business problems. The area of study should not be confused with computer science which is more theoretical in nature and deals mainly with software creation, and neither with computer engineering, which focuses more on the design of computer hardware.

Workflow Management (WfM) focus on the automation of processes involving combinations of human and machine-based activities, particularly those involving interaction with information technology applications and tools. A Workflow Management System (WfMS) is "a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications" [WfMC97].

Workflow Information Systems (WIS) cover the application of information technology to business problems. Their primary characteristic is the automation of processes involving combinations of human activities with information technology applications.

1.5 Reading map

This thesis is structured as follows:

Chapter 1 defines the thesis statement based on a set of concerns of workflow and UIs considered important for developing UIs for workflow information systems. We have identified, defined and justified the terminology that will be further employed in this dissertation. In addition, we delimited the scope of this thesis.

Chapter 2 is dedicated to the state of the art with current methods, models, and software dedicated to describe workflow model; considering in addition task

Chapter 1. Introduction

models, UI description languages and Computer-Supported Cooperative Work (CSCW). We conclude with a summary of the state of the art that enables to establish a list of shortcomings of current works. Based on these shortcomings a set of requirements are identified and will further be employed in the validation process of the results provided by our method.

Chapter 3 concerns the aspects of our conceptual model. We present the workflow model and show how UsiXML concepts have been expanded by directly introducing workflow concepts. Then, the composing models are detailed by emphasizing our conceptual contribution. Further, the semantics of the framework is presented along with the supporting syntax and stylistics.

Chapter 4 is dedicated to the method for developing workflow user interfaces employed in the current dissertation. First, we introduce a method for specifying workflow and then the method for developing UIs.

Chapter 5 concerns the implementation aspects of our method. The software supporting our method is introduced.

Chapter 6 address the validation of the method. Besides, we present two case studies with different level of complexity: (1) Order personalized compression stockings over Internet, and (2) Requesting a credit to buy a car. The internal validation consists of reflections that aim to assess the characteristics of our methodology based on the set of considered requirements.

Chapter 7 concludes this thesis by identifying its contribution to the four dimensions of this proposal: models, method, language, and software support. In addition, the chapter presents several possible extension paths for future work and provides an evaluation of the graphical notation.

Since each person, according to her knowledge and what she expects from a document, has different interests on reading the text, we propose a reading map (Figure 1-3) on how to read this thesis depending on the domain of expertise.

Chapter 1. Introduction

	Workflow expert	Task model expert	UI expert
Chapter 1	Introduction		
Chapter 2	State of the art (2.1,2.2,2.3,2.5,2.7)	State of the art (2.1,2.2,2.3,2.4,2.6, 2.7)	State of the art (2.1,2.3,2.4,2.5,2.6, 2.7)
Chapter 3	Conceptual model for WUIs		
Chapter 4	Developing WUIs		
Chapter 5	Software support		
Chapter 6	Validation		
Chapter 7	Conclusion		

Figure 1-9. Reading map.

Chapter 2 State of the art

2.1 Context

Considering the environment in which resides the phenomena of interest, the state of the art in the application domain of the research, and the existing artifacts and processes found in the application domain, Hevner [Hevn07] propose three inherent research cycles. Figure 2-1 shows the *relevance cycle* (it initiates design science research with an application context providing the requirements of the research) and the *rigor cycle* (it provides past knowledge to the research project to ensure its innovation).

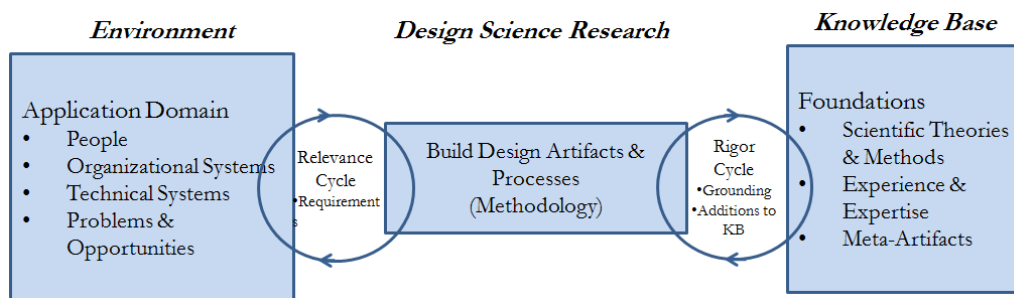


Figure 2-1. Designing science research, relevance and rigor cycles [Hevn07].

In every organization there are workers developing tasks to achieve business goals, some tasks can be supported by a computer system. A UI is the means to interact with the computer system. This chapter presents an overview of the state of the art divided in five related parts: the UI description language to generate UIs, the organizational modeling, the workflow characteristics, the task model concepts, and the cooperative work issue.

Searching the existing literature to provide a meta-model of workflow concepts which are related to UI issues, led us to identify shortcomings and established requirements.

2.2 User interface description language

Interactive computing systems are computer systems allowing a certain level of control by a human agent. This control is operated through a UI. A UI can be defined as any software and/or hardware piece allowing a user to communicate with a computer system. In other words, a UI is a software component, a hardware component, or a series of such components enabling a user to interact. A graphi-

cal UI can be defined as follows: “A user interface is a collection of techniques and mechanisms to interact with something” [Gali97]. A UI Description Language (UIDL) consists of a high-level computer language for describing characteristics of interest of a UI with respect to the rest of an interactive application. There is a wide range of UI description languages that are widely used, with different goals and different strengths. On one hand we have software vendors UIDLs and, on the other hand, there are free license UIDLs to use. Attempting to capture the essence of UIs at various levels of abstraction for different purposes gives birth to many XML markup languages; consequently there is a need to conduct an analysis of features that make all these proposals discriminant and appropriate for any specific purpose. A review of XML UI description languages was produced [Guer09a] that compares a significant selection of various languages addressing different goals, such as multi-platform UIs, device-independence, and content delivery. Below we present some results obtained, detailed information of this review can be found in [ITEA10].

This review and comparison on XML-UIDLs started in [Souc03]; Table 2-1 compares the properties of the different UIDLs according the eight criteria:

- *Component models*: this criterion specifies the different models involved in the UIDL and it gives the aspects of the UI that can be specified in the description of the UIs. The *task model* is a description of the task to be accomplished by the user; the *domain model* is a description of the objects the user manipulates, accesses, or visualizes through the UIs; the *presentation model* contains the static representation of the UI, and the *dialog model* holds the conversational aspect of the UI.
- *Methodology*: this point gives the approaches that are supported by the different UIDLs. Different approaches to specify and model UIs exist: 1) Specification of a UI description for each of the different contexts of use. As a starting point, a UI specification for the context of use considered as representative of most case, the one valid for the context of use considered as the least constrained or finally the one valid for the context of use considered as the most comprehensive is specified. From this starting UI specification, corrective or factoring out decorations (e.g., to add, remove, or modify any UI description) are applied so that UI specifications can be derived for the different contexts of use. 2) Specification of a generic (or abstract) UI description valid for all the different contexts of use. This generic UI description is then refined to meet the requirements of the different contexts of use.
- *Tools*: some of the languages are supported by a tool that interpret or render the specification to a specific language and/or platform.
- *Supported languages*: specify the programming languages to which the XML-based language can be translated.

Chapter 2. State of the art

- *Supported platforms*: specify the computing platform on which the language can be rendered by execution, interpretation or both.
- *Abstraction level*: each UIDL may exhibit the capability to express a runnable UI (instance level), one or many models involved in the development of this UI (model level), how these models are built (meta-model level), and what are the fundamental concepts on which this operation is based (meta-meta-model level).
- *Amount of tags*: to reach the above level of abstraction, each UIDL manipulates a certain amount of tags, which is also highly depending on the coverage of the concepts.
- *Coverage of concepts*: depending on the level of abstraction, each UIDL may introduce some specific vs. generic concepts (e.g., a given presentation model vs. any model, each custom-defined), their properties (e.g., to what extent can a concrete presentation be specified), and their relations.

Table 2-2 compares UIDLs along the following dimensions:

- *Standard* criterion specifies if the UIDL is already a standard language.
- *Specificity* indicates if the UIDL could be used in one or multi platforms or devices.
- *Publicly available*: depending on the availability of the language deep analysis can be done. This category was used to discard many languages that lack on documentation or that is confidential. The possible values are: 0 = no information available, 1 = not available, 2 = poorly available, 3 = moderately available, 4 = completely available and 5 = completely available with meta-models.
- *Level of usage*: depending on the usage of the language we create the following categories: 0 = unknown, 1 = one person, 2 = two or more persons, 3 = one organization, 4 = two or more organizations and 5 = massive usage.
- *Weight of the organization* behind denotes the degree of influence of the organization to which the UIDL belongs.
- *Type* criterion informs whether the UIDL is a research or industry work.

Chapter 2. State of the art

UIL	Models	Methodology	Tools	Supported languages	Supported plat- forms	Abstraction level	Tags	Concepts
DISL [Scha06]	Presentation, dialog and control	Specification of a generic, platform-independent multimodal UI	Rendering engine	VoiceXML, Java MIDP, Java Swing, Visual C++	Mobile and limited devices	Model level	Not specified	Head element, interface classes (structure, style, behavior), state, generic widgets
GIML [Kost04]	Presentation, dialog, and domain	Specification of a generic interface description.	GTK (Generalized Interface Toolkit)	C++, Java, Perl	Not specified	Meta-model	15 tags	Interface, dialog, widget, objects
ISML [Crow03]	Presentation, task, dialog, domain	Specification of a generic UI description	Under construction	Java, Microsoft foundation class, Java swing classes	Desktop PC, 3D screen	Model level	Not specified	Mappings and constraints, action events, meta-objects, display parts, controller parts, interaction definition
RIML [Deml03]	There is no information	Specification of a generic UI description	There is no information	XHTML, XFORMS, XEvents, WML	Smart phone, pda, Mobile, Desktop Pc	Model level	There is no information	Dialog, Adaptation, layout, element
SeesoaXML [Luyt04]	Task, Presentation, dialog	Specification of a generic UI description	CCOM (Beta Version 1.0 2002) PacoSuite MSC Editor	Java AWT, Swing, HTML, java.microedition, applet, VoxML, WML Juggler	Mobile, desktop PC, Palm III	Model level	Not specified	Component, port, connector, contract, participant, blueprint, instance, scenario, platform, user, device
SunML [Pica03]	Presentation, dialog, domain	Specification of a generic UI description	SunML Compiler	Java Swing, VoiceXML, HTML, UIML,	Desktop Pc, pda	Model level	14 tags	Element, list, link, dialog, interface, generic events, synchronization
TeresaXML [Pate03]	Presentation, task, dialog	Specification of a generic UI description	CTTE Tool for task Models	Markup: Digital TV, VoiceXML,	DigitalTV, Mobile, Desktop PC,	Model level	19 tags	Mappings, models, , platform, task, input, output

Chapter 2. State of the art

		tion	Teresa	XHTML/SVG, X+V Programming: C#				
UTML [Helm08, Abra99]	Presentation, dialog, domain	Specification of a generic UI description	UIML.net, VoiceXML renderer, WML renderer, VB2UMIL	HTML, Java, C++, VoiceXML, QT, CORBA, and WML	desktop PC, a handheld device, TV, mobile	Model level	50 tags	interconnection of the user interface to business logic, services
UsiXML [USIX07]	Presentation, task, dialog, domain	Specification of a generic UI description	SketchiXML, GraphiXML, FlowiXML, FlasiXML, QtkiXML, InterpiXML,	HTML, XHTML, VoiceXML, Java3D, VRML, X3D, XAML, Java, Flash, QTK	mobile, Pocket PC, interactive kiosk, a wall screen, pda	Meta-model	118 tags	Task, domain, AUI, AIC, CUI, CIO
WSXL [IBM02]	Presentation, dialog, domain	Specification of a generic UI description	Not specified	HTML	PC, Mobile phone,	Model level	12 tags	CUI=XForms, WSDL, Mapping=XLang Workflow=WSFL, Logic=XML event
XICL [Gome04]	Presentation, dialog,	Specification of a generic UI description	XICL STUDIO	HTML, ECMA Script, CSS e DOM.	desktop PC	Model level	Not specified	Component, structure, script, events, properties, interface
XIML [Eise00, Eise01, Puer02]	Presentation, task, dialog, domain	Specification of a generic UI description	XIML Schema	HTML, java swing, WLM	Mobile, desktop PC, PDA	Model level	32 tags	Mappings, models, sub models, elements, attributes and relations between the elements

Table 2-1. Properties comparison of UIDLs.

Chapter 2. State of the art

UIL	Standard	Specificity	Publicly available	Level of usage	Weight of the organization behind	Type
DISL	No	Multimodal UIs for mobile devices	2	3	Paderborn University	Research
GIML	No	Multimodal	3	2	Technical University of Dresden and Leipzig University of Applied Sciences	Research
ISML	No	GUI, multiplatform, multidevice	2	1	Bournemouth University	Research
RIML	No	Mobile devices	0	3	Industry: SAP Research, IBM Germany, and Nokia Research Center along with CURE, Ubi- Call, and Fujitsu Invia	Industry
Sees- coaXML	No	Multiplatform, multidevice, dynamic generation UI	2	3	Expertise Centre for Digital Media Limburgs Universitair Centrum	Research
SunML	No	Multiplatform	4	3	Rainbow team, Nice University	Research
Tere- saXML	No	Multiplatform, multidevice,	4	3	HCI Group of ISTI-C.N.R.	Research
UIML	No	Multiplatform	4	3	Harmonia, Virginia Tech Corporate Research (OASIS)	Industry
UsiXML	No	Multiplatform	5	3	UCL	Research
WSXL	No	multiplatform, multidevice	4	3	IBM	Industry
XICL	No	Multiplatform	3	3	Federal University of Rio Grande do Norte, Brazil	Research
XIML	No	multiplatform, multidevice	4	3	Redwhale Software	Research

Table 2-2. General features of UIDLs.

2.3 Organizational modeling

Organizational theory examines alternative structures for (business) organizations [Hate06]. The organizational structure discipline refers to the way that an organization arranges people and jobs so that its work can be performed and its goals can be met. In the literature there are various propositions to model organizations, such as:

- *The Enterprise Ontology* of Uschold et al. [Usch98] which provide a framework for enterprise modeling presenting a collection of terms and definitions relevant to business enterprises.

<i>ACTIVITY etc.</i>	<i>ORGANISATION</i>	<i>STRATEGY</i>	<i>MARKETING</i>	<i>TIME</i>
Activity	Person	Purpose	Sale	Time Line
Activity Specification	Machine	Hold Purpose	Potential Sale	Time Interval
Execute	Corporation	Intended Purpose	For Sale	Time Point
Executed Activity Specification	Partnership	Purpose-Holder	Sale Offer	
T-Begin	Partner	Strategic Purpose	Vendor	
T-End	Legal Entity	Objective	Actual Customer	
Pre-Condition	Organisational Unit	Vision	Potential Customer	
Effect	Manage	Mission	Customer	
Doer	Delegate	Goal	Reseller	
Sub-Activity	Management Link	Help Achieve	Product	
Authority	Legal Ownership	Strategy	Asking Price	
Activity Owner	Non-Legal Ownership	Strategic Planning	Sale Price	
Event	Ownership	Strategic Action	Market	
Plan	Owner	Decision	Segmentation Variable	
Sub-Plan	Asset	Assumption	Market Segment	
Planning	Stakeholder	Critical Assumption	Market Research	
Process Specification	Employment Contract	Non-Critical Assumption	Brand	
Capability	Share	Influence Factor	Image	
Skill	Shareholder	Critical Influence Factor	Feature	
Resource		Non-Critical Influence Factor	Need	
Resource Allocation		Critical Success Factor	Market Need	
Resource Substitute		Risk	Promotion	
			Competitor	

Figure 2-2. Overview of enterprise ontology [Usch98].

- *The Agent-Oriented Enterprise Meta-model* [Jure06] provides common terms used to describe an organization, it is divided into five sub-models: organizational sub-model (describing actors, roles, responsibilities and capabilities), goals

sub-model (describing enterprise and business process purposes), conflict sub-model (indicating inconsistencies in the business process), process sub-model (describing how actors achieve goals), and objects sub-model (describing non-intentional entities and assumptions about the environment of the organization).

- The most referred model of organizational configurations was introduced by [Mint82], the famous *Structure in five*. Each configuration contains six components: 1) operating core, the people directly related to the production of services or products; 2) strategic apex, serves the needs of those people who control the organization; 3) middle line, the managers who connect the strategic apex with the operating core; 4) techno structure, the analysts who design, plan, change or train the operating core; 5) support staff, the specialists who provide support to the organization outside of the operating core's activities; and 6) ideology: the traditions and beliefs that make the organization unique.

From all these propositions there are a few basic principles that apply and a number of common terms like: activity (task), process, resource (actor), role, organizational unit, goals. Even that exist more terms used to represent organization structures, we focus on those terms that are relevant to develop UIs to workflow information systems; they are described in next chapter.

2.4 Workflow

This section is about the different notations and languages to represent/describe workflow, tools, and workflow patterns used.

Workflow models focus on how work is done to accomplish organizational goals; it defines how task, information, and documents are passed from one participant to another in the organization [WfMC99]. According to [Mars94, Mars97], the essential workflow characteristics are: “*tasks/activities* that are performed by (role-playing) *persons*, using supporting *tools* that give access to a variety of *shared information resources*”.

A wide range of process modeling techniques has been proposed over the years. In [Rose06] an examination on how process modeling techniques have developed over the last four decades is presented. According to [Carl97], the existing languages for workflow process modeling can be classified in five distinct groups:

- IPO (Input-Process-Output)-based languages, such as the activity networks used in IBM MQSeries Workflow [Leym94]. These languages describe a

Chapter 2. State of the art

workflow as a directed graph of activities, denoting the sequence of their execution.

- Speech-Act-based approaches (sometimes called Language Action approaches) as used in Action Technologies Action Workflow product [Medi92]. These approaches model a workflow as an interaction between (at least) two participants that follow a structured cycle of conversation. Namely the phases: negotiation, acceptance, performance and review are distinguished.
- Constraint-based modeling methods, such as Generalized Process Structure Grammar (GPSG), proposed by [Glan96]. These approaches describe a process as a set of constraints, leaving room for flexibility that is otherwise governed by the restrictions of the IPO- or Speech-Act-based approaches. Constraint-based modeling languages are typically text-based and resemble traditional programming languages, whereas a graphical representation of these models seems difficult.
- Role-modeling based process descriptions, such as Role Activity Diagrams (RADs).
- Systems thinking and system dynamics that are used in conjunction with the concept of learning organizations.

Due the number of existing propositions of process modeling, we may have omitted some significant works, but our goal is not to be exhaustive. Below we present a brief description of some of them, for more details see Appendix A.

- *Petri Nets* is a modeling language, graphically depict the structure of a distributed system as a directed bipartite graph with annotations. They are a technique for modeling and analyzing processes. In [vand98] Petri net theory is applied to process modeling and workflow nets (WF-net) are introduced. A WF-net is a Petri net which has a unique source place (i) and a unique sink place (o).
- *Statechart Diagrams* [W3C05, Wodt97]. A statechart diagram is a graph that represents a state machine describing the response, of an object of a certain class, to the receipt of outside stimuli.
- *Business Process Modeling Notation* (BPMN) [OMG06] is a standardized graphical notation for drawing business process in a workflow. It provides a simple means of communicating process information to the other business users, process implementers, customers, and suppliers.

Chapter 2. State of the art

- *UML Activity Diagrams* [Duma01]. An activity diagram provides a view of the behavior of a system by describing the sequence of actions in a process. UML activity diagrams are intended to model both computational and organizational processes.
- *Yet Another Workflow Language* (YAWL) is a workflow language based on Petri nets. It has been extended with features to facilitate patterns involving multiple instances, advanced synchronization patterns, and cancellation patterns [vand05a].
- The *XML Process Definition Language* (XPDL) is proposed by the Workflow Management Coalition (WfMC) [WfMC08] to interchange process definitions between different workflow products. The keywords of XPDL are based on the terms defined in the WfMC glossary [WfMC95].
- *eXchangeable Routing Language* (XRL) is a workflow language that uses XML for the representation of process definitions and Petri nets for its semantics. XRL/flower is a WfMS to support XRL [Verb02].

In the literature there are some other works like *Business Process Execution Language for Web Services* (BPEL4WS) [Andr03] that is a technical standard used to describe executable process models based on Web Services; *Event-driven Process Chains* (EPCs) [Kelle92] which are a leading standard for modeling business processes, which was developed within the ARIS framework; *Unified Enterprise Modeling Language* (UEML) [Vern02, Peti02] which intent to act as an intermediate language between existing enterprise modeling languages and to facilitate interoperability between a wide variety of enterprise modeling languages and models.

Currently, several models and design methods support the development of complex workflow-based applications providing notations for business process and including tool support. There is a plethora of tools support for process modeling including open source systems and vendors systems, research into all of them is not the objective of this section, and we just present some of them (more details in Appendix A):

- *The Progression Model* [Stav04] has incorporated some of the managing concepts of workflow to increase the flexibility in ISs. It makes explicitly the steps and transactions a user undertakes when using an IS. As the user progress towards accomplishing a task or goal, the progression model infrastructure records each step and the state of the transaction and workflow.
- *Microsoft Windows Workflow Foundation* (WWF) [Espo05] is an extensible framework for developing workflow solutions on the Windows platform. It

provides a single, unified model to create end-to-end solutions that span categories of applications, including human workflow and system workflow.

- *WebSphere® MQ Workflow* (IBM) [IBM06] supports long-running business process workflow as they interact with systems and people. Automates and tracks business process in accordance with business design. Provides integration processes with rich support for human interactions.
- *Workflow on Intelligent and Distributed database Environment (WIDE)* [Casa96] defines an advanced conceptual model for describing both the flow of activities and the organizational environment in which these activities are performed.
- *Business Process Visual ARCHITECT* (BP-VA) [Visu07] is a visual modeling tool that provides support for BPMN; it covers from model specification to graphical notation, including different presentation options.

Due to the large amount of existing workflow products we came to a point where it is very difficult to analyze and compare their capabilities on a common scheme. However, they can be gathered in a collection of workflow patterns that provide the basis for an in-depth comparison of commercially available workflow systems.

- *Control-flow patterns* [vand03] identified useful basic routing constructs such as sequence, parallel split, synchronization, exclusive choice. From a data perspective, there is a series of characteristics that occur repeatedly in different workflow modeling paradigms.
- *Workflow data patterns* [Russ04] are aimed at capturing the various ways in which data is represented and used in workflows.
- *Workflow resource patterns* [Russ05] correspond to the manner in which tasks are allocated to resources, that is any entity that is capable of achieving some work unit.

2.5 Task model

One important characteristic of workflow is the concept of *task*. A common definition for a task is “an activity performed to reach a certain goal” [vanW98]; task models play an important role because they indicate the logical activities that an application should support to reach user’ goals.

In the literature, there are several definitions for task models. Accordingly to [Limb04a], task models describe the various tasks to be carried out by a user in interaction with an interactive system. Task modeling is considered as an important activity in order to conduct and ensure user-centered design of interactive

Chapter 2. State of the art

systems. This is reinforced by the fact that a task model is assumed to capture most elements describing how a task is carried out by a particular user in a given context of use (i.e., a triple user-computing platform-physical environment) or in a given scenario [Limb04b].

While the purpose of task analysis is to understand what tasks should be supported and what are their related attributes, the aim of task modeling is to identify more precisely the relationships among such tasks. Task models are explicit representations of user tasks that can help support certain rigorous forms of task analysis. One of the advantages of task modeling is the characterization of the logical activities that an interactive application must support independently of any underlying technology or implementation.

Task modeling has become today a widely recognized activity in the UI development life cycle. Several task models are precisely defined and are adequately made editable through software; without an explicit understanding of the different attributes of these models, it is difficult to select a specific one to achieve one's goals.

In [Guer08b] we analyze and discuss some well-known and widely used task notations, examining which characteristics they exhibit and which attributes they cover, the discussion was started on a previous work [Limb04b] that we updated and enriched.

The task models analyzed show a variety of concepts and relationships. Differences between concepts are both syntactic and semantic. Syntactic differences cover differences of vocabulary used for a same concept across models. Semantic differences are related to the conceptual variations across models. Semantic differences can be of major or of minor importance. A major difference consists in the variation of entities or relationships definitions and coverage; for instance, a same concept does not preserve a consistent definition across models. A minor difference consists in the variation of expressing an entity or a relationship. For example, constructors in Groupware Task Analysis (GTA) or Task Knowledge Structures (TKS) express temporal relationship between a task and its subtasks (parent-children), although the set of constructors is not identical in all models, while operators in ConcurTaskTree (CTT) are used between sibling tasks (tasks at the same level in the hierarchy of the task tree).

Table 2-3 provides the variations between task models; we compare main features of task models to know which of them are similar and outstanding of taking into account for the construction of a broader task model.

Chapter 2. State of the art

Organizational concepts describe the environment in which the task is developed (for instance a laboratory).

Cooperative aspect, since a UI is designed for a given role; this distinction is useful both for UI design and for coordinating the computer-supported cooperative work.

Role concept indicates classes of actors to whom certain subsets of tasks are allocated. *Users* may have internal representations of their own roles and others' roles. A user can and will take on more than one role and more than one user can and will take on a given role.

Type of task specifies the nature of the task.

Formalization specifies whether a model is based on a formal system or not.

Graphical representation determines if the task model has a graphical notation or not.

Scope of constructors express the scope of the task elements on which the temporal relationships work. The scope can be the *parent* or the *sibling* when any temporal operator constraint affects the ordering, respectively, between a father node in the task decomposition and its children or between siblings of the same father.

Tool determines if the task model is supported with a software environment.

After, we compare the temporal relationships that each task model manages (Table 2-4).

2.6 Computer-supported cooperative work

Computer-Supported Cooperative Work (CSCW) objective is the creation of systems for groups. Knowing groups behavior and user's behavior inside organizations, CSCW consider users that are not alone, that are part of an organization. Software engineering must guide the development of applications to support working groups (know as groupware), increasingly used and needed in the new ways of interaction in our societies. CSCW started as an effort by technologists to learn from economists, social psychologists, anthropologists, organizational theorists, educators, and anyone else who can shed light on group activity. It is also a place for system-builders to share experiences and inform others of technical possibilities and constraints. Applications include desktop conferencing and videoconferencing systems, collaborative authorship applications, electronic mail and its refinements and extensions, and electronic meeting rooms or group support systems.

Chapter 2. State of the art

	AMBOSS [Gies08]	ANSI/CEA [CEA07,Rich09]	CTT [Pate99]	Diane + [Taba9]	GOMS [Card83]	GTA [vand96, vand99]	HTA [Anne67, Anne04]	TKS [John89, John91]	TOOD [Mahf97]
Organizational concepts	+-	X	X	X	X	X	X	X	X
Cooperative aspect	√	X	√	X	X	√	√	√ (FKS extension)	√
Role concept	√	√	√	+-	X	√	+-	√	√
User concept	+-	+-	X	+-	√	√	X	√	√
Type of task	Human, system, abstract (based on the role)	General task	Abstract, user, system, interactive	Manual, automatic, interactive	General task	Complex, unit, basic	General task	General task	Manual, automatic, interactive, cooperative
Formalization	X	X	√	√	X	X	X	X	√
Graphical representation	√	X	√	√	X	√	X	X	√
Scope of constructors	Parent	Multiple levels	Sibling	Sibling	Multiple levels	Parent	Parent	Multiple levels	Sibling
Tool	AMBOSS	DTask	CTTE	TAMOT	GLEAN3, GOMSED, QGOMS	EUTERPE	HTAWin, PI-TAS/Task Architect (potentially available)	ADEPT (partially based on model)	TOOD-IDE ETOOD

√ Supported, +- Partially supported, X Unsupported

Table 2-3. Main features of task models.

Chapter 2. State of the art

	AMBOSS	ANSI/CEA	CTT	Diane +	GOMS	GTA	HTA	TKS	TOOD
Decomposition	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy	√ Hierarchy
Sequence	√ SEQ	+ Ordered = true, information passing (Post condition)	√ Enabling, enab- ling with infor- mation passing	√ Ordered se- quence	√ Sequence	√ Seq	√ Fixed sequence	√ Sequence	√ Sequence
Iteration	X	+ MinOccurs+ MaxOccurs	√ Iteration, finite iteration	√ Loop	+ Loop (If, then, else)	X	+ Stop rules	X	X
Choice	√ ALT	+ Precondition	√ Choice	√ Required choice, free choice	+ Or (If, then, else)	√ Or	√ Selective rule	√ Or	√ Choice
Optionality	+ Barrier	+ MinOccurs+ MaxOccurs	√ Optional	√ Optional	+ Optional (If, then, else)	+ Start condition	X	X	X
Interruption	X	X	√ Suspend- resume, disabl- ing	X	+ Interruption (If, then, else)	+ Stop condition	√ Stop rules	X	√ Interruption
Concurrency	√ SER	+ Ordered = false	√ Concurrent, concurrent communicating tasks, indepen- dence	√ unordered sequence	+ Concurrency (If, then, else)	X	+ Selective rule	X	√ Concurrency
Cooperation	+ Precondi- tion	X	√ Cooperative	X	X	√ Cooperation	+ Teamwork	√ Collabora- tion(FKS exten- sion)	√ Collaboration
Parallel	√ PAR, SIM	X	X	√ Parallel	X	√ And	√ Dual task (time sharing)	√ And	√ Simultaneity

√ Explicit supported, +- Implicit supported, X Not specified

Table 2-4. Comparison of task relationships.

Other application domains that are related include Computer-Assisted Design/Computer-Assisted Manufacturing (CAD/CAM), Computer-Assisted Software Engineering (CASE), concurrent engineering, distance learning [Grud94].

The developers of CSCW systems not only have to design for single users interacting with the system but also for groups interacting *via* the system. CSCW developers often focus on the fine grained organization of features, the design of interfaces, and the way people actually use their systems.

Some CSCW methodologies are:

- AMENITIES (A METHodology for aNalysis and desIgn of cooperaTive systems) [Garr03] is a methodology; see Figure 2-3, centered on the group by covering relevant aspects of its structure and behavior. This methodology follows an iterative process allowing the refinement of the model as a consequence of the continuous analysis, requirement evaluation, etc.

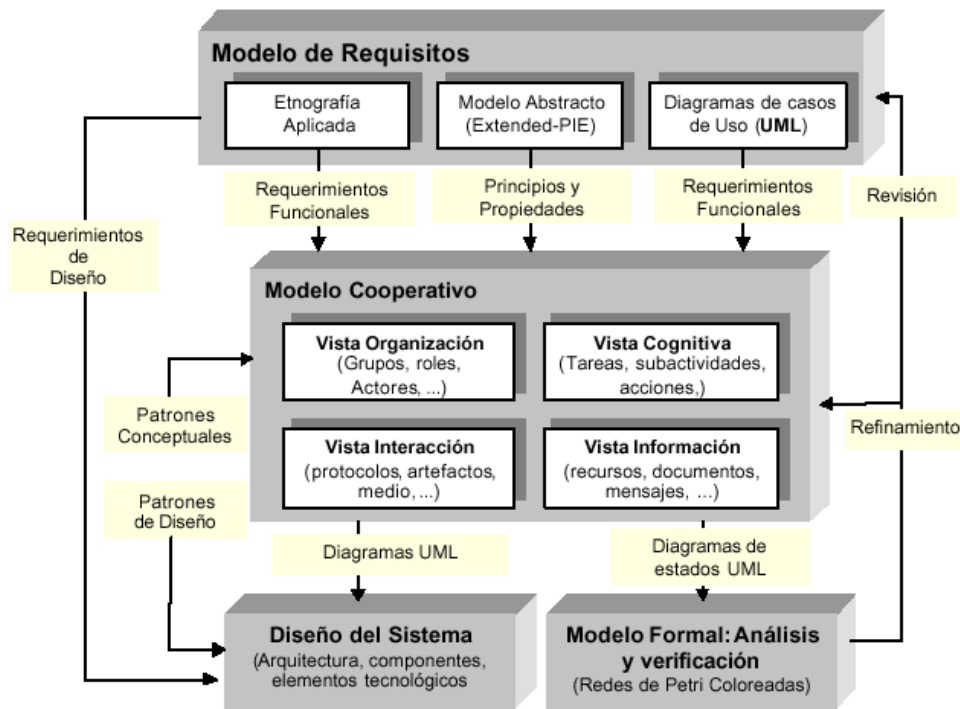


Figure 2-3. Methodological phases AMENITIES.

The general schema of the methodology is composed in four modules:

1. Requirements model (modelo de requisitos) where requirements and specifications are collected.

2. Formal model (Modelo formal) is an automated analysis, using Color Petri Nets (CPN), which provides the semantics.
3. System design (diseño del sistema) is the connection point of the methodology with the software development process.
4. Cooperative model (Modelo cooperativo) is a conceptual model that describes the structure and functionality of a CS. The study of the cooperative facet is divided into four views; see Figure 2-4, with the most relevant aspects of CS.



Figure 2-4. Conceptual views of the cooperative model of AMENITIES.

- i) Group view (vista de grupo); describes the organizations and constraints within. The organization is seen as a group of interrelated users performing tasks. Three concepts are used: users, task and roles. The relationships among users are conditioned by the system collaborative constraints: capacity and laws.
- ii) Cognitive view (vista cognitive) represents knowledge acquired or owned by each member of the group. In this view task analysis is done, first a role interface is defined and then is detailed; a more detail is added later to each task.
- iii) Interaction view (vista de interacción); is a study of the processes that imply a dialog among participants. Dialog modes are established through protocols.

Chapter 2. State of the art

iv) Information view (vista de información) collects all the information that is shared on the collaborative scenario.

■ CIAM (Collaborative Interactive Applications Methodology) [Moli06] is a methodology based on a set of models providing guidelines for the design and development of UIs for interactive applications supporting group work. This methodology is based on a group of views interrelated:

1. Organizational view; considers aspects related to the structure in which CW is performed and the hierarchical relations: organization, role, group, software agent, actor, and work team.
2. Interaction list; model the structure and flow of work, being the task the main concept. In this view concepts such as process CSCW, task event, interdependency, working group task, individual task, collaborative task and flow of work, are considered.
3. Data view; model the data that is manipulated, considers concepts such as: shared context, object, attributes, operations, composed object and simple object.
4. Interaction view; describes tools and interactive aspects of the application. Considering concepts such as tool, tool dependency on the task, tool independent from the task, UI, interactive task, interaction task, application task, and relationship among tasks, UI collaborative and visualization areas.

The phases of the CIAM methodology is centered on the modeling of the group, after on the process model and finally on the modeling of interactive tasks performed by the user.

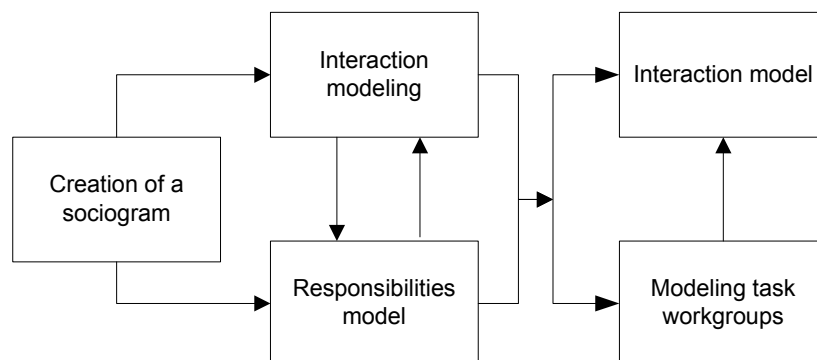


Figure 2-5. Methodological phases CIAM.

Chapter 2. State of the art

The first step is the creation of a sociogram (Figure 2-5) that represents the structure of the organization and the relations that exist among the different members that composed it. Like this, apart from the categories identified in the organizational view (role, actor, software agent, working group), basic relationships among this element can be characterized, for instance, inheritance, performance, and association. The second step is the interaction modeling consists on the specification of the tasks that define the work in groups. Such tasks can be classified as: cooperative, collaborative or individuals. So, after the specification of the working group the set of responsibilities is extended to each role, by adding individual assignments (responsibilities model). The next step, consist on extending the details of the cooperative and collaborative tasks previously identified (Modeling task workgroups). Cooperative tasks are modeled with graphs of responsibility decomposition; each task is decomposed in sub-tasks. Collaborative tasks are modeled including the roles involved on their execution and the data models that they manipulate. Finally, in the interaction model, aspects of the human-computer interaction are modeled. The notation used is the tree decomposition of CTT.

- TOUCHE (Task-Oriented and User-Centered Process Model for Developing Interfaces) proposes a process model and a methodology for the development of UI for CSCW systems centered on the user and driven by tasks [Peni03, Peni07]. The methodology is based on four models, see Figure 2-6.

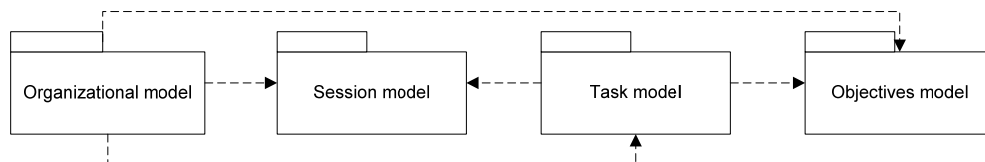


Figure 2-6. Models relationships.

Organizational model; represents the structure of the actors involved in the system, so as the collaborative relationships among them. The concepts identified for such representation are: organizational structure, elements of the organization, organizational relationships, roles, actors, individuals, users, agent, organizational structure relationships, organizational group relationships, performance relationships, aggregation relationship, hierarchical and co-interaction relationships. *Task models*; specify individual or collective actions performed by users to achieve determined objectives. *Objectives model*; represent

group goals and the relevance of performing the task. *Session model*, provides system views at determined times.

The methodology has four steps: requirements elicitation, analysis, design, and implementation.

2.7 Chapter summary

Supporting business process with the help of computer-based systems is a necessary prerequisite for many companies to stay competitive. The interest in how people work, with an eye to understanding how technology could support them, gave birth to approaches like Workflow and Computer-Supported Cooperative Work. CSCW focus on how cooperative activities and their coordination can be supported by means of computer systems, and Workflow focus on the automation of business process, it coordinates documents, information or task among participants.

“Workflow technology can be seen as a specific type of groupware for supporting collaboration that is based on work being planned and articulated in work process models” [Carl97].

We presented the workflow technology which facilitates modeling, redesigning and administration of process in an organization. Different workflow notations have been proposed for the design and graphical representation of it. Some of them have a formal structure (i.e. a mathematical base) like Petri Nets, other ones like UML has an informal base.

In addition, workflow patterns have been identified for assigning tasks to resources, for routing constructs, and for comparing workflow products on a common scheme. Several workflow management systems have been developed to manage the workflow including open source and vendor systems.

An important aspect on workflow is the concept of task. Nowadays there are a lot of proposition to analyzed and model task. We offered a comparison of well-known task models. Although specific task models differ, they have many things in common (e.g. task decomposition and relationships). Almost all task modeling methods also use graphical representation to show the information of the model and they are supported by a tool. Since there is no apparent need to conduct any research for the model part (the task model has gained today a precise and shared definition) or any development for the tool support (excellent software are publicly available for this purpose), we believe that there is still some research to be

conducted for improving the methodological guidance for conducting task modeling.

Workflow modeling languages are often used for describing organizational and group work, while task analysis and task modeling are used for describing and formalizing individual work [Trae99]. Both essentially describe the same domain, but at different levels.

Considering that our purpose is developing UIs for workflow systems, a review of UI description languages was produced that compares a significant selection of various languages addressing different goals, such as multi-platform user interfaces, device-independence, and content delivery. There has been a long history and tradition to attempt to capture the essence of UIs at various levels of abstraction for different purposes. The return of this effort today gains more attraction, along with the dissemination of XML markup languages, and gives birth to many proposals for a new UI description language. Consequently, there is a need to conduct an analysis of features that make all these proposals discriminate and appropriate for workflow modeling. We want to highlight that this review is part of the *Deliverable 1.1 SoTA of User Interface Description Languages* of ITEA 2 -UsiXML project.

From our review of User Interface description language (UIDLs) we have the following observations: XIML [Puer02] has a well formed structure, considers the task, the user, it has a good tool support, it is possible to expand, however by the fact that is available via a non-commercial research license, it is difficult to add new elements. UIML [Abra99] is oriented to multi-devices UIs, no present a task neither a user model, the effort required to extend it is considerable and present the same dilemma as XIML that is available via a non-commercial license. Due to the availability of UsiXML, the structure, the possibility to expand, and all the advantages that it has, we select to be our textual syntax. However, our method for developing UIs can be applied to any other language in the same way.

After this, we have identified some important concepts to be considered for a task-based design of a UI: 1) Goal and task hierarchies are essential. 2) Operators must express temporal constraints between tasks. 3) Role specification in terms of tasks is a minimal requirement for dealing with cooperative aspects. 4) Objects and actions that are performed on them make possible the detailed modeling of presentation and dialog of the UI.

All the literature review helps us to identify shortcomings on exiting works from which we derive a set of requirements for our methodology.

2.7.1 Shortcomings

We identified the following shortcomings that serve as incentives to consider this topic an important, original, yet unsolved and challenging research problem of workflow UI development. These shortcomings confirm our first perception of the problem captured as concerns in the introduction.

1. *Limited support for extensions.* Some methods have a particular concern for extension possibilities of their underlying models.
2. *Limited support of expressivity.* A conceptual model should provide enough details to address problems that motivated the elicitation of its constituent concepts.
3. *Difficulty of obtaining a workflow system evolving with the organization.* When the organizational structure changes, but the workflow does not change, it is complicated to reassign the tasks if a logical workflow has not been defined independently of the organizational structure [Stav05]. The consequence is that the software used for supporting individual or collective tasks need to be reassigned. It is likely that a workflow sustaining an organization cannot be updated instantly each time the organization is evolving, but if at least some support for considering evolution is provided, the impact of the required changes could be minimized.
4. *Limited support for organizational model.* Taking into account the complex structure of the organization and not only individual users performing the tasks. Even most of the workflow models in the literature consider the concept of *role*, we assume that the key elements of an organization are not just its people, but also its structure (representing the places where work is performed), objectives and rules.
5. *Lack of integration of individual interactive systems in the global workflow.* Often, individual users in organizations already have their own software to use to carry out their interactive tasks, as they have been assigned. This is not problematic. But when the time comes to communicate the results of their tasks to their hierarchy or to their colleagues, apart from using traditional e-mail, chatting or instant messaging they do not rely on dedicated software for supporting the communication [Laud06].
6. *Absence of considering real group requirements in workflows.* Typically, a workflow often represents a tiling of existing software with people trying to coordinate their work among themselves by relying on implicit (e.g., manual, verbal, informal) communication schemes.

7. *Lack of guides to identify the points where to start the task modeling and where to stop it.* Until when should we proceed with task modeling such as decomposition and refinement?
8. *Difficulty to model tasks.* People may diverge on their interpretation of what needs to be captured in a task model and what not. In particular, what makes a task and what does not make a task? Different people may produce different, possibly inconsistent, task models for the same design problem because they do not share the same perception or rules; a same person (e.g., a task analyst, a task modeler) may produce task models with different levels of details depending on the design problem; even more, a same person can produce different task models for the same design problem over time.
9. *Limited support for the complete task life cycle.* Task life cycle refers to a series of states that task goes through, from creation to termination.
10. *Limited development of UIs for workflow users.* Developing UIs for a workflow is a challenge because the interaction of the users takes place in two different logical levels synchronously. On the higher level, the workflow manager specifies and monitors the workflow execution. On the other level, the workflow users carry out their assigned tasks.
11. *Software support.* Lack of software to support the complete workflow model with the generation of UIs.

2.7.2 Requirements

The problems captured as shortcomings are the starting point of the issues we want to address in this thesis. This is why we propose a set of requirements to cope with the shortcomings. The methodology is composed of four main axes: models, language, method and the software supporting the method. Thus, we grouped the requirements in these four axes.

Modeling requirements

Modeling requirements have been gathered in terms of: their usage (requirements 1, 2), robustness (requirements 3, 4, 5 and 6). The expected result of the conceptual modeling is a set of models to represent WfIS with at least the following list of characteristics:

1. *The model must be extensible* (shortcoming 1). The model structure should allow the extension to workflow and task concepts.
2. *Expressivity* (shortcoming 2). The model should, at least, provides enough de-

tails to allow an implementation of the system it describes.

3. *Organizational modeling* (shortcomings 3 and 4). Strict separation between the description of the organization and workflow process specification to support the evolution of the organization and the workflow independently.
4. *Support carrying out group tasks* (shortcoming 6). From the individual level continuously throughout the global level: individual, within groups, for the group as a whole, among groups, within organization, and among organizations (Mandviwalla & Olfman [Mand94] criteria for support group interactions).
5. *Support multiple ways to support a group task* (shortcoming 6). In principle, there should not be a unique way to carry out a single group task, but several mechanisms should be offered for this purpose. If a mechanism is no longer available, another one should be selectable.
6. *Support the group evolution over time* (shortcoming 6). When the group evolves over time, the workflow definition should be easily maintained and reflected in the system.

Language requirements

The language specifies the conceptual modeling. It is the machine readable version of the concepts (requirements 7). The expected language must specify all our models and should support extensibility (requirements 8).

7. *Machine readable*. States that the proposed ontology should be legible by a machine.
8. *Extendibility*. It refers to the ease of adapting a conceptual structure to the occurrence of newly elicited concepts. HCI is a vast area covering the definition of multiple types of interfaces, interaction techniques, and interaction contexts. A specification language should be equipped with extension mechanisms to allow its evolution in parallel with the artifact it seeks to model.

Method requirements

The method makes the concepts described in the conceptual modeling coherent when used together to design WfIS. The method should be clear (requirement 9), provide guidance on its use (requirement 10, 11) and provide means to develop UI for WfIS (requirement 12).

9. *Method explicitness*. States that the component steps of our method should define in a comprehensive way their logic and application. This requirement is

Chapter 2. State of the art

motivated by the lack of explicitness of the existing approaches in describing the proposed transformational process.

10. *Identification criteria* (shortcomings 7 and 8) introduced a set of precise criteria that can be used in order to identify a task in a textual scenario.
11. *Consider a complete task life cycle* (shortcoming 9). Considering all the possible states in order to create a complete task life cycle from the beginning to the end.
12. *Flexibility in creating UIs* (shortcoming 10). It is important to generate UIs for workflow in a manner which the platform is independent, customizable, and extensible.

Software requirements

The software supporting the method must provide means to design WfIS (requirements 14), UIs for the WfIS (requirement 15) and consider that the execution engine must include agendas handling (requirement 13).

13. *Management of agendas* (shortcomings 5 and 10). Allow user to optimize distributed tasks providing them the capability to store and manage a flexible personal TODO list of tasks, and provides the manager the capability to manage his own task list.
14. *Support software for method for specifying workflow* (shortcoming 11). It refers to the possibility to develop software to support the method for specifying workflow information systems.
15. *Support software for method for developing user interfaces* (shortcoming 11). It refers to the possibility to develop software to support the method for developing workflow user interfaces.

The set of requirements merely expresses the minimum set of characteristics expected for each component of the methodology. This does not mean that this thesis is only limited to those requirements. In the next chapter, our conceptual model for workflow information systems is presented, having these requirements into account.

Chapter 2. State of the art

Chapter 3 Conceptual model for workflow information sys- tems

3.1 Context

The workflow environment and the existing knowledge base, let us to design and extend four different design artifacts: model, language, method, and software. In this chapter we present the conceptual workflow model (model) associated to a UIDL (language) (Figure 3-1). The conceptual model is desired not for workflows per se, but for workflow concepts that have some impact on the user interface. A *User Interface Description Language* (UIDL) is a formal meta-language used in Human-Computer Interaction (HCI) in order to describe a particular UI independently of any implementation technology. As such, the UI might involve different interaction modalities (e.g., graphical, vocal, tactile, haptic, multimodal), interaction techniques (e.g., drag and drop) or interaction styles (e.g., direct manipulation, form fillings, virtual reality). A common fundamental assumption of most UIDLs is that UIs are modeled as algebraic or model-theoretic structures that include a collection of sets of interaction objects together with behaviors over those sets. A UIDL can be used during:

- *Requirements analysis*: in order to gather and elicit requirements pertaining to a UI of interest.
- *Systems analysis*: in order to express specifications that addresses the aforementioned requirements pertaining to a UI of interest.
- *System design*: in order to refine specifications depending on the context of use
- *Run-time*: in order to execute a UI via a rendering engine

The design process for a UIDL encompasses the definition of the following artifacts:

1. **Semantics.** They express the context, meaning and intention of each UIDL abstraction captured by the underlying UI meta-models. UI Meta-Models are normally represented by means of UML Class Diagrams, OWL or other conceptual schemas. Semantics are usually conveyed using natural language.
2. **Syntax.** *Abstract Syntax* is a syntax that makes it possible to define UI models (in accordance with the UIDL semantics) independently of any representation formalism. *Concrete Syntax*, they are (one or more) concrete representation

formalisms intended to express syntactically UI Models. Many UIDLs has an XML-based concrete syntax. In fact XML has been proven to be extremely useful in the description UIs according to the different CAMELEON abstraction levels. By leaving the names, hierarchy, and meanings of elements/attributes open and definable, XML lays the foundation for creating custom and modular (new formats can be defined by combining and reusing other formats) XML-based UIDLs.

3. **Stylistics.** They are graphical and textual representations of the UIDL abstractions that maximize their representativeness and meaningfulness in order to facilitate understanding and communication among different people. Stylistics are typically used by models editors and authoring tools.

UIDL is a more general term than "User Interface Markup Language" (UIML) which is often defined as [Wikipedia]: "a markup language that renders and describes graphical user interfaces and controls. Many of these markup languages are dialects of XML and are dependent upon a pre-existing scripting language engine, usually a JavaScript engine, for rendering of controls and extra scriptability". Thus, as opposed to a UIML, a UIDL is not necessarily a markup language (albeit most UIDLs are) and does not necessarily describe a graphical user interface (albeit most UIDLs abstract only graphical user interfaces).

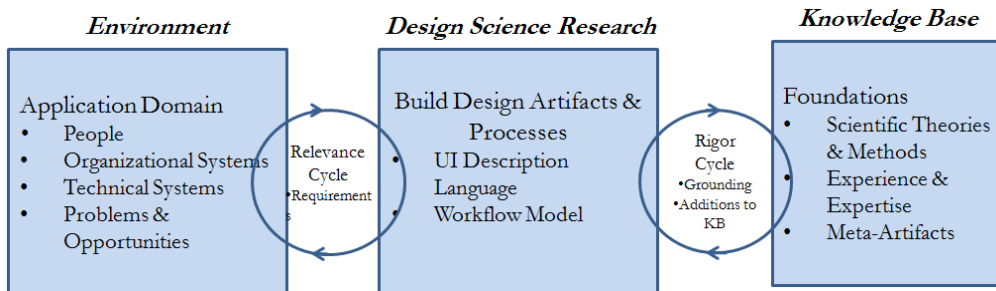


Figure 3-1. Schema for conceptual modeling derivation (adapted from [Hevn07]).

3.2 Semantics: meta-modeling for workflow information systems

After a comparison of the UIDLs in [Guer09a], we have considered that UsiXML is the suitable language that could accommodate workflow concepts in a flexible way. The current section presents our contribution to UsiXML v1.8 [USIX07] which integrates the improvements and the expansions accomplished by the present thesis in order to adapt the UsiXML to the requirements of workflow. As notation to represent this approach, we use the Unified Modeling Language (UML) class diagrams. Figure 3-2 provides a simplified overview of the meta-model.

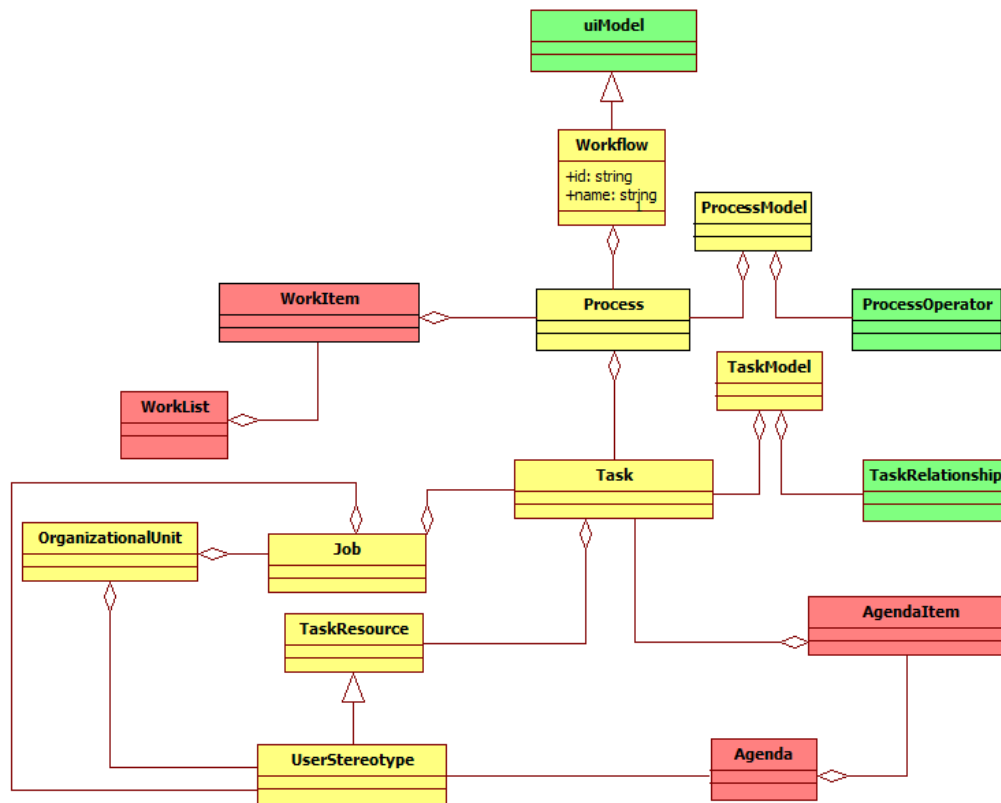


Figure 3-2. Simplified overview of workflow meta-model.

In following sections, the meta-model is fully explained. Color is introduced in order to denote explicitly those models whose instances are potentially highly dynamic (instances of red classes) at run-time. By run-time is meant when the WfIS is in execution. While all concepts are involved during the design of the WfIS, instances of classes are not necessarily modified or used at run-time. For instance, once a task model relationship has been established during the design of the WfIS there is no way to change it at run-time (green classes). No need to focus on this aspect when considering the implementation of the WfIS.

The color distinction proved to be useful for the implementation of a workflow editor, to discard classes that were not needed at all for the design of the WfIS (red classes), and to keep an understanding of those concepts that at run-time are to be implemented (yellow and red classes).

Notice that there are some classes that are both used at design-time and run-time. An instance of this class can change significantly (the agenda is an example as it changes constantly) thus the use of the red color. An instance of a class that changes moderately (the job definition for the execution of the task is an example

as the definition does not normally change on a regular basis) thus the use of the yellow color.

Developers can use this information to imagine potential benefits on the design and implementation of the WfIS such as: giving priority to dynamic information to be more visible or easily accessible.

3.2.1 Workflow model

Our workflow model is called *FlowiXML* (user interfaces to workflow based on *UsiXML*) (Figure 3-3) and is composed of processes and tasks. It describes how the work in organization flows by defining models of: process (what to do?), tasks (how to do it?), and the organizational structure (where and who will perform it?). A workflow model has at least one process and each process has at least two tasks.

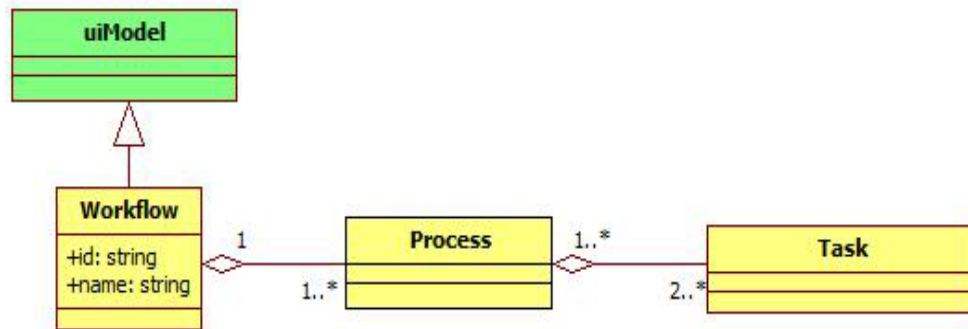


Figure 3-3. Workflow model.

3.2.2 Process model

The definition of a process indicates the ordering of tasks in time, space, and resources. Our model (Figure 3-4) is based on Workflow Nets notation proposed in [vand98, vand02] and is compatible with the workflow resource patterns proposed in [Russ05]. The process model is composed of:

- *Process*. A process consists of a number of tasks and a set of relationships among them. The definition of a process indicate which tasks must be performed and in what order.
 - Rationale: Each process has a unique identifier and a name. *Frequency* is the relative occurrence of execution of a process; it is evaluated on a scale from 1 to 5. A value of 1 means that a process has a low frequency, 5 mean that a process has the highest frequency. *Importance* implies a value judgment of the task with respect to main users's goals. *Category* is evaluated on primary,

Chapter 3. Conceptual model of workflow

secondary and tertiary. Primary processes are known also as production processes, they deal with cases for the customer. As a rule, they are the processes that generate income for the company, and are clearly customer-oriented. Secondary processes are known as support processes, then they are those that support the primary ones. Tertiary processes are the managerial processes that direct and coordinate the primary and secondary processes [Boda89, vand02].

- Example: We shall examine how a travel agency deals with a booking.

■ *Process operators.* They indicate the different ways in which the processes could be executed.

- Rationale: It is not mandatory that processes on the workflow need to be performed in the same order in which they were identified; all process can be modeled using different paths.
- Example: Two or more processes that must be performed in a strict order are called a sequence.
- The process operators are synchronization, exclusive choice, multi choice, sequential, parallel split, and simple merge :
 - Sequential indicates that a number of processes are performed one after the other.
 - Exclusive choice indicates that one of several branches is chosen.
 - Multi choice is used when any of two processes is chosen. However it is also possible that both need to be executed.
 - Parallel split indicates that two or more process can be executed in parallel, thus allowing processes to be executed simultaneously or in any order.
 - Simple merge indicates that two or more alternatives branches come together without synchronization.
 - Synchronization is used when multiple parallel processes converge into one single thread of control.

■ *Work item.* It is the representation of the work to be processed.

- Rationale: It could contain the identifier of the workflow and the identifier of the process to which it belongs, the identifier of the task resource that execute the task and the identifier of the organizational unit where the task is performed; the actual status of

Chapter 3. Conceptual model of workflow

the task (for instance: not started, in progress, in progress with delay, suspended, cancelled, and finished). It also considers the date when the task begins, the deadline (i.e. due Date), the date when the task could be assigned or delegated, and the date when the task was completed.

- Example: The task *fill-in form* that belongs to *claims control* process, is been executed (is in progress) by the secretary.

■ *Work list*. It is a list of work items related with a given workflow.

- Rationale: The work list allows *workflow manager* to guide, support, and coordinate tasks, but not overly constraint it.
- Example: Workflow manager can offer a task to those resources that have the shortest queue of work.

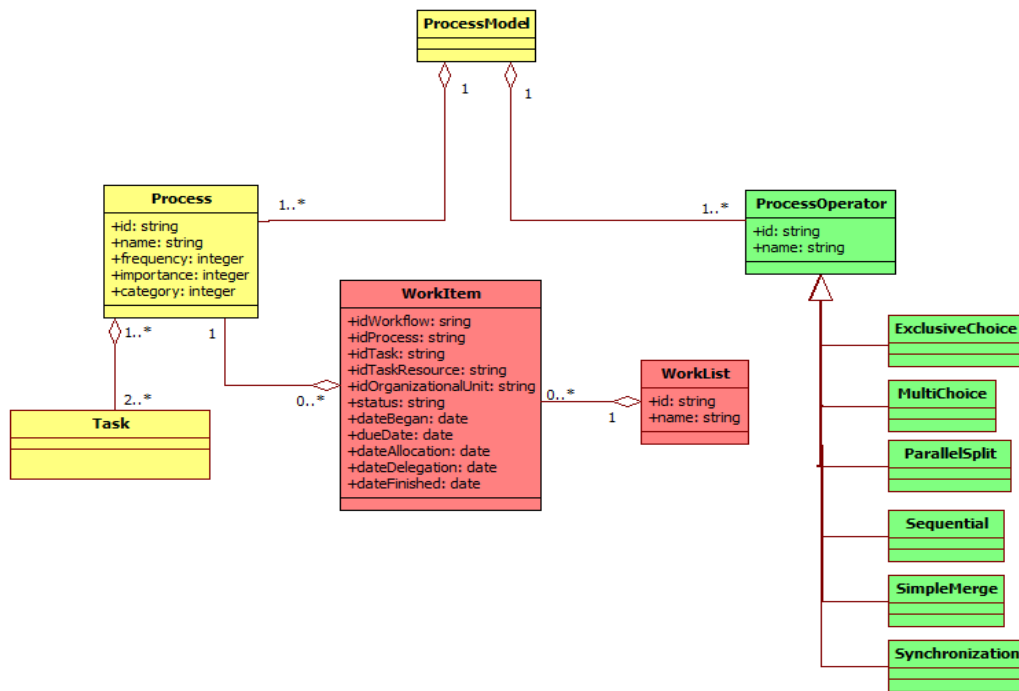


Figure 3-4. Process model.

3.2.3 Task model

The existing Task Model defined in [USIX05] is an extended version of Concur-TaskTree notation defined in [Pate97]. A task is an activity that has to be performed by users to reach a given goal [vanW98]. Introducing task models description to the workflow models is motivated by the following reasons:

Chapter 3. Conceptual model of workflow

- Task models describe, opposed to process models, end users' view of interactive tasks while interacting with a computer-based system.
- It is true that in a process model we can describe how tasks are performed adding the detail desired with process hierarchies; however, we consider that specific temporal operators like *iteration* can be more naturally defined in a task model despite that they can also be used in process model.
- Task models have been used in the domain of Model-Driven Architecture to develop user interfaces.

The full task model is depicted in Figure 3-5.

3.2.3.a Existing task model

The task model describes the interactive tasks to be performed by the user of an application through the application's user interface. It is composed of *tasks* and *task relationships*. *Tasks* are, notably, described with attributes such as *name* and *category*. The *name* of the task is generally expressed as a combination of a verb and a substantive (e.g., take pictures). The *category* attribute identifies one of the four basic task types: user, interactive, system or abstract.

Leaf tasks are described by two additional attributes (i.e., *taskType* and *taskItem*) that enable a refined expression of the task nature. The taxonomy is twofold: a verb describes the type of activity at hand and an expression designates the type of object on which the action is operated. By combining these two dimensions a derivation of interaction objects that are supposed to support a task becomes possible. The *taskType* attribute refers to verbs that indicate the actions required to perform the task, while the *taskItem* attribute refers to an object type or subject of an action.

Tasks repetitions are measured with a *frequency*, *importance* implies a value judgment of the task with respect to main users' goals; the level of structuration of a task, i.e. the interrelation of its parts in an organized whole, is evaluated by *structurationLevel* on a scale from 1 to 5. A value of 1 means that a task is not structured, 5 means that a task is very structured. The *complexityLevel* measures the difficulty in developing the task, the *criticality* attribute indicates an absolute criticality evaluation of a task, a critical task is essential for accomplishment of goals (i.e. the actual importance of a task for a manager has nothing to do with whether it is critical or not); the *centrality* attribute assesses if the task is the primary importance or it is peripheral. *TerminationValue* attribute is a condition to be satisfied in order to verify that the task is completed.

Chapter 3. Conceptual model of workflow

Task relationships are of two main types: decomposition and temporal; *decomposition* enables representing the hierarchical structure of a task tree, *temporal* allows specifying a temporal relationship between sibling tasks of a task tree.

LOTOS operators are used here [Pate97]. *UnaryRelationships* are temporal relationships that connect several instances of the same task, they are: *optional*, *iteration*, and *finiteIteration*. *BinaryRelationships* are a type of temporal relationships that connects several instances of two different tasks; they are: *enabling* relationship specifies that a target task cannot begin until source task is finished; *disabling* relationship refers to source task that is completely interrupted by a target task; *suspendResume* relationship refers to source task that can be partially interrupted by a target task and after the target task is completed the source task will be concluded; *orderIndependence* relationship refers when two tasks are independent of the order of execution; *concurrencyWithInformationPassing* relationship is a type of temporal relationship where two tasks are in concurrency execution and passing information between them; *independentConcurrency* relationship is a type of temporal relationship where two tasks are executed concurrency but are independent one to each other and there is no information interchange; *enablingWithInformationPassing* relationship specifies that a target task cannot be performed until the source task is performed, and that information produced by the source task is used as an input for the target task; *deterministicChoice* relationship refers to two source tasks that could be executed but once that one task is initiated the other cannot be accomplished anymore; *undeterministicChoice* relationship defines the relation between two source tasks in which both task could be started but once one task is finished the other cannot be accomplished anymore.

3.2.3.b Extended task model

Our contribution to the task model described in [USIX05] consists mainly in added task attributes and relationships. The condition is an attribute not considered in previous version and essential to the occurrence of the task.

- *preCondition*. Before a task can be performed it must fulfill certain conditions.
- *postCondition*. A task can be considered finished if it fulfills the necessary requirements.

The binary relationships that were introduced are:

- *Cooperation*. This relationship specifies the cooperation that could be between two or more tasks.

- *Inclusive choice.* This relationship specifies two tasks that: both could be executed or just one of them or neither of them.
- *Disabling with information passing.* This relationship occurs if one task is completely interrupted by another task; and the information produced in the first task is used as an input for the second task.

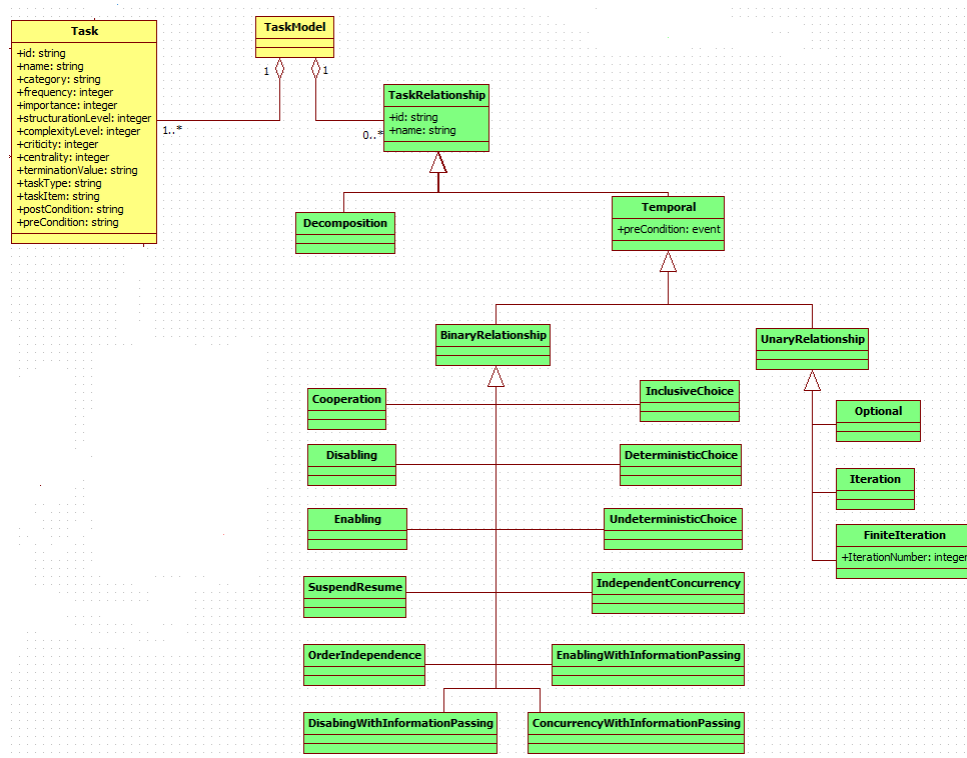


Figure 3-5. Task model.

3.2.4 Organizational model

Although ISs are an integral part of organization; the key elements of an organization are its people, structure, business processes, politics, and culture [Usch98] [Mint82]. We propose an organizational model (Figure 3-6) representing the places where work is performed and the users that perform it. This part contributes to UI adaptation to different categories of users and security of IS by blocking access to UIs when the user does not have the permission to perform the task.

Chapter 3. Conceptual model of workflow

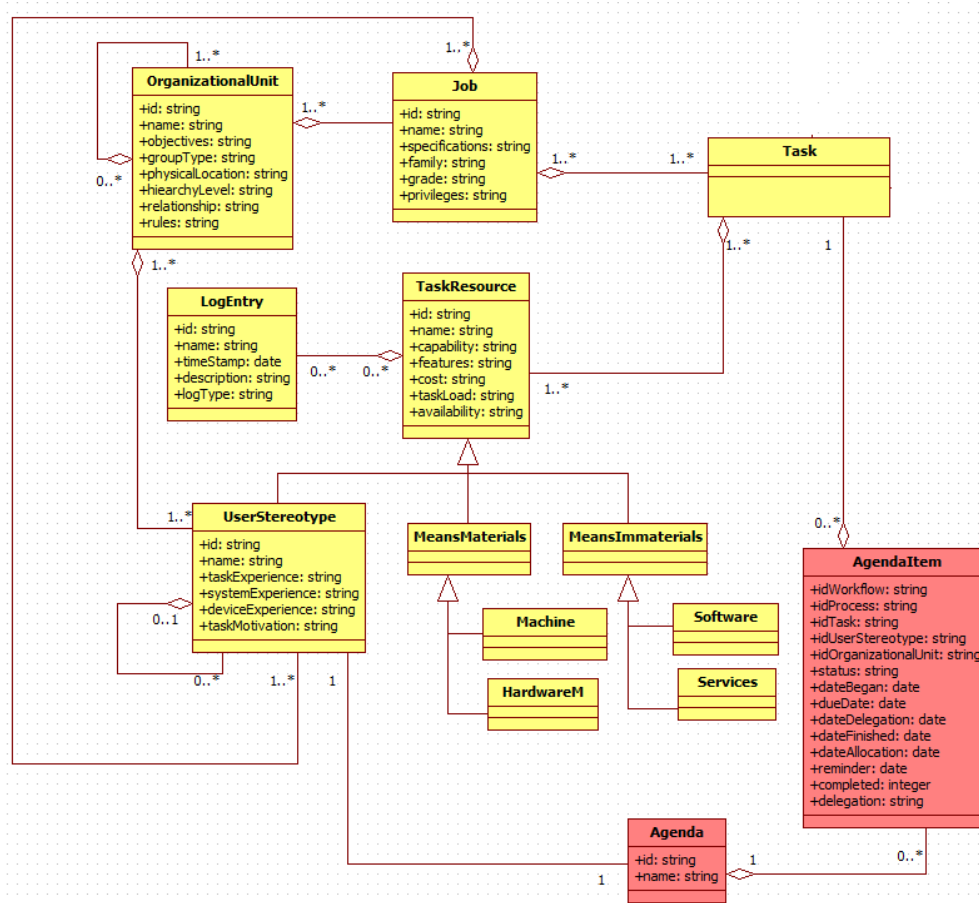


Figure 3-6. Organizational model.

The attributes of this model are:

- *Organizational unit.* An organizational unit is a formal group of people working together with one or more shared goals or objectives. It could be composed of other organizational units.

 - Rationale: Normally an organization has a *name* and *objectives*, i.e. business benefits that an organization expects to achieve as a result of injecting project product(s) into itself or its environment [Mint82, Hatc06]. The *rules* describe the way and criteria that govern the execution of the process. *GroupType* represent the different groups that exist inside an organization as units (permanent groups that undertake work items relating to a common set of business objectives), teams (similar to organizational unit but not necessarily permanent in nature), branches (which defines a grouping at specific physical location), divisions (which defines a large

Chapter 3. Conceptual model of workflow

scale grouping persons either along a regional geographic or business purpose lines). *PhysicalLocation* indicate the place in the organization where an employee works. *HierarchyLevel* indicates the position of a worker within the organizational hierarchy. In terms of the organizational hierarchy, each worker may have a number of specific *relationships* with the other worker. It could be direct report, subordinates, delegates, etc.

- *Task resource.* A task resource is an entity that is directly or indirectly involved in carrying out the work.
 - Rationale: Each taskResource has a unique identifier and name. This class includes *capability* (it refers to the attributes of a resource as his qualifications, skills, and abilities), *features* (they further describe specific characteristics that the resources may possess that could be of interest when allocating tasks [Russ05]), *cost* (it indicates the monetary value that the resource represent for the organization), *task load* (it refers to the load of task that is allocated to a resource), and *availability* (it refers to the availability of the taskResource in order to carry out a task) as attributes.
- *User stereotype.* This class represents the set of users sharing the same values for same describing parameters so as to create a hierarchical decomposition of the user population into user stereotypes. Each stereotype may in turn be decomposed into sub-stereotypes.
 - Rationale: User stereotypes are, notably, described with attributes such as *name*, *task experience* (it describes the experience level the users of the stereotype have in order to accomplish the task independently of any supporting system. It can range from 1 (low) to 5 (high)), *system experience* (it describes the experience level the users of a stereotype have in order to manipulate the interactive system. It can range from 1 (low) to 5 (high). This attribute is different than the task experience in the sense that it expresses the skills the user may have with the system independently of the particular task. It could be with another task.), *device experience* (it refers to the experience level users of a stereotype have in manipulating advanced interaction devices with the system. Advanced interaction devices include: stylus, pen-based interaction, gesture-based interaction, foot mouse, etc; while traditional devices include touch screen, keyboard, and mouse. It

Chapter 3. Conceptual model of workflow

can range from 1 (low) to 5 (high)), and *task motivation* (it specifies the level of motivation raised by the task. Can range from 1 (low) to 5 (high). For instance, a tedious and repetitive task can be considered of low motivation while a game can be considered as highly motivating).

- *Means materials*. It is a type or resource that is physically tangible and is a non human resource.
- *Means immaterials*. It is a type of resource that is physically intangible; it does not have a material form or substance.
- *Log entry*. LogEntry describes specific activities that resources may possess. A log is a way to store the history of actions over a WfIS. Each resource may have zero or more log Entry associated with it.
 - Rationale: Each log entry has a *name* and a *description*. *Time stamp* indicates the time and the date from which the work item is available. *Log type* indicates the type of work items that a userStereotype has, it could be past (work items completed), present (work item that are executing) or future (work items committed to undertaking at a specified future times).
- *Job*. Jobs are the total collection of tasks, duties, and responsibilities assigned to one or more positions which require work of the same nature and level.
 - Rationale: The job concept allows assembling tasks under a same umbrella in a way that is independent of individual resources in the workflow. In this way, several individuals could play a particular job, and jobs could be interchanged dynamically. Its attributes are: *name*, *specifications* (it defines what worker characteristics are required to perform the job for it to be carried out competently), *family* (it refers to a group of jobs having the same nature of work but requiring different levels of skill, effort, responsibility, or working conditions), *grade* (one of the classes, levels or groups into which jobs of the same or similar value are grouped for compensation purpose), and *privileges* (rights or special advantage that enjoys a userStereotype by concession of his organizational level).
 - Example: *Administrator*, *assessor*, and *executive* are examples of jobs. Job *assessor* consists of the workers *Annie*, *Tom* and *Jack*; but *Tom* could be a member of both the *administrator* and *assessor* categories.
- *Agenda item*. Agenda items are the tasks that a *userStereotype* has to perform.

Chapter 3. Conceptual model of workflow

- Rationale: It could contain the identifier of the workflow, process, and task to which it belongs, the identifier of the userStereotype which is responsible for it, the actual status of the task (for instance: not started, in progress, in progress with delay, suspend, cancel, finished). Also, it indicates the date when the task begins, the deadline (i.e. due date), the date when it is delegated, and the date and time in which the alarm send a message to the userStereotype for reminding the task.
- Example: The task *fill-in form* belongs to *claims control* process, was assigned on March 15th, and must be done on March 18th.

■ *Agenda.* The agenda is a list of agendaItem that are assigned to *userStereotypes*. A userStereotype has one and only one agenda and an agenda belongs to one and only one userStereotype.

- Rationale: The agenda displays the tasks in a tabular format, it enables users to organize and manage their personal work. It provides a single place for userstereotypes to manage the whole range of their tasks.
- Example: Selecting a task, *administrator* could delegate it to one *assessor* in particular.

3.2.5 Mapping model

Model integration is well-known issue in transformation driven development of UI [Puer99]. A mapping model contains a series of related mappings between models or elements of the models. It serves to gather a set of inter-model relationships that are semantically related. For instance, the mapping *allocated to* associates a task with a *resource*. The existing Task Model defined in [USIX05] was extended for expressing the resource allocation patterns and mappings for the tasks, such extensions are detailed below.

3.2.5.a Existing mapping model

It consists of one to more *interModelRelationships*, a part of them being used throughout the steps of the transformational approach for UI generation. This can be useful for enabling the derivation of the system architecture (mappings between domain and user interface models, see Appendix B), for traceability in development cycle (reification, abstraction, and translation), for improving the preciseness of model derivation heuristics. We briefly define them below, it is not in the scope of this work to get into the details, and more information about their usage can be found in [Limb04a]:

Chapter 3. Conceptual model of workflow

- *Manipulates*: maps a task to a domain concept, i.e. a class, an attribute, a method or any combination of these types. This relationship is useful when it comes to find the most appropriate interaction object to support a specific task.
- *Updates*: is a mapping defined between an interaction object (an entity (e.g., window, push button, text field, check box) that can be perceived and/or manipulated by the users.) and a domain model concept (specifically, an attribute). It describes the situation where the attribute of an object in the domain model must be synchronized with the content of a UI object.
- *Triggers*: is a mapping defined between an interaction object and a domain model concept (specifically, an operation). This mapping describes that a UI object is able to trigger a method from the domain model.
- *IsExecutedIn*: maps a task to an interaction object (a container or an individual component) allowing its execution. This relationship is notably useful for deriving a dialog control component, for ensuring that all tasks are supported appropriately by the system.
- *IsReifiedBy*: maps a task to an interaction object (a container or an individual component) allowing its execution. This relationship is notably useful for deriving a dialog control component, for ensuring that all tasks are supported appropriately by the system.
- *Observes*: is a mapping defined between an interaction object and a domain model concept (either an attribute, or an output parameter of a method).
- *IsAbstractedInto*: indicates that an abstract object is the reification of a concrete one through an abstraction transformation.
- *IsTranslatedInto*: indicates that an interaction object (abstract or concrete) is adapted into another one as a result of an adaptation transformation.
- *HasContext*: maps any model element to one or several contexts of use, where a context of use is defined as a triple <user, platform, environment>.
- *IsShapedFor*: allows associating a plasticity domain to a concrete user interface. Plasticity refers to the ability of a user interface to mould itself to a range of contexts of use [Coll08].

3.2.5.b Extended mapping model

As pointed out early, of particular interest from a resource perspective is the manner in which tasks are advertised and ultimately bound to specific resources for execution. At this point however, we have introduced the concepts of task and resource, the next step is establishing a relation between these concepts. Based in [Russ05] we use the workflow resource patterns to capture the various ways in

Chapter 3. Conceptual model of workflow

which tasks are allocated. A full description with examples of these workflow resource patterns is presented in chapter 4.

Figure 3-7 shows the complete mapping model; in this case we do not apply a class color due to the concept of each class is not object of changes.

We extend the mapping model by adding:

- *Is grafted on.* It grafts a task on another one. This relationship is useful when a task (T_j) has been executed, and a complementary task (T_i) is defined to realize the first task. T_i is completely autonomous to T_j [Pet94].
- *Is defined by.* It refers to a task defined by a workflow designer.
- *Is allocated to.* It corresponds to a task that is assigned to a resource. We define several allocation relationships for this assignment:
 - *Assignment.* It is the way in which tasks are advertised to specific resources for execution. It could be: direct, deferred, authorization-based, separation of duties, case handling, retain familiar, capability-based, history-based, hierarchy level-based, role-based.
 - *Distribution.* It is the way in which newly created tasks are proactively offered or allocated to resources by the workflow system. It could be: offer single-resource, offer multiple-resources, allocation single-resource, early distribution, distribution on enablement, late distribution.
 - *Allocation principles.* It corresponds to the way in which tasks are allocated to resources by the workflow system. They could be: random allocation, round robin allocation, shortest queue.
 - *Managing.* It is the way in which the tasks are initiated by individual resources. It could be: resource-initiated allocation, resource-initiated execution- allocated item, resource-initiated execution - offered item, system-determined work queue content, resource-determined work queue content, selection autonomy.
 - *Deviation.* It corresponds to situations in which the normal sequence of state transitions for a task is changed. It could be: escalation, deallocation, stateful reallocation, stateless reallocation, suspension/resumption, skip, redo, pre-do.
 - *Auto start.* It corresponds to situations where the execution of task is triggered by specific events in the lifecycle of the task or the related

Chapter 3. Conceptual model of workflow

process definition. It could be: commencement on creation, commencement on allocation, piled execution, chained execution.

- *Visibility*. It represents the scope in which task availability and commitment are able to be viewed by resources. It could be: configurable unallocated work item visibility, configurable allocated work item visibility.
- *Multiple resources*. It corresponds to situations where there is a many-to-many correspondence between the resources and work tasks in a given allocation or execution. It could be: simultaneous execution, additional resources.

■ *Is delegated to*. A userStereotype (called the delegator) who is assigned to a task delegates it to another userStereotype (called the delegate) provided that this user stereotype fulfills the skill conditions for carrying out the delegated task. An exception could be introduced where the skills are close enough to the skills required.

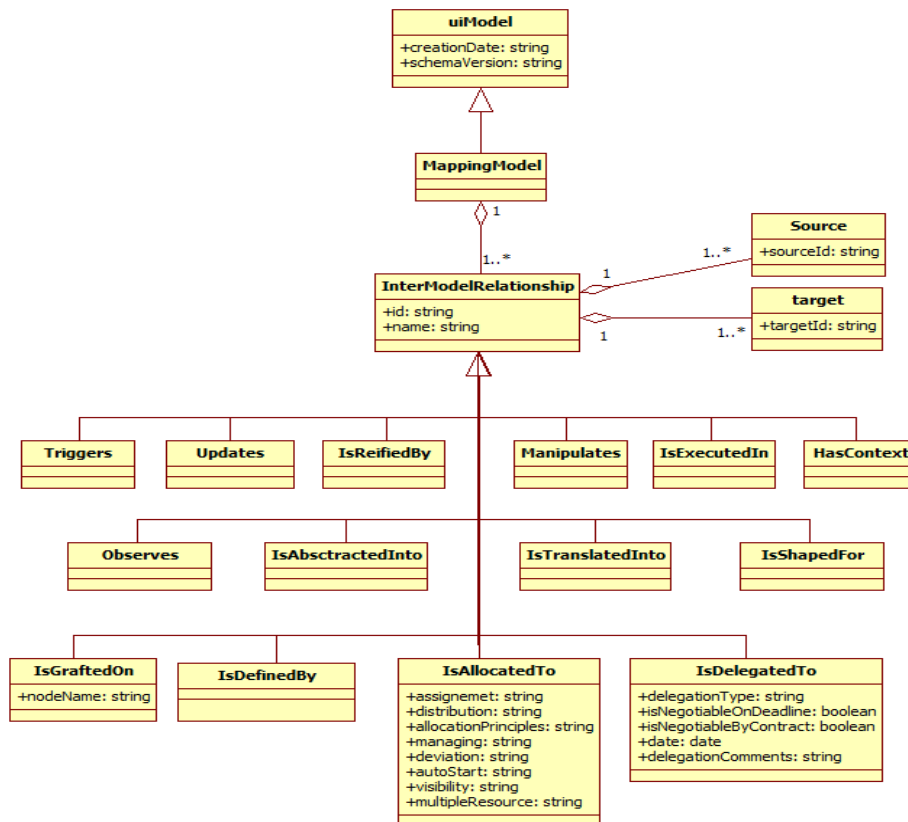


Figure 3-7. Mapping model.

Chapter 3. Conceptual model of workflow

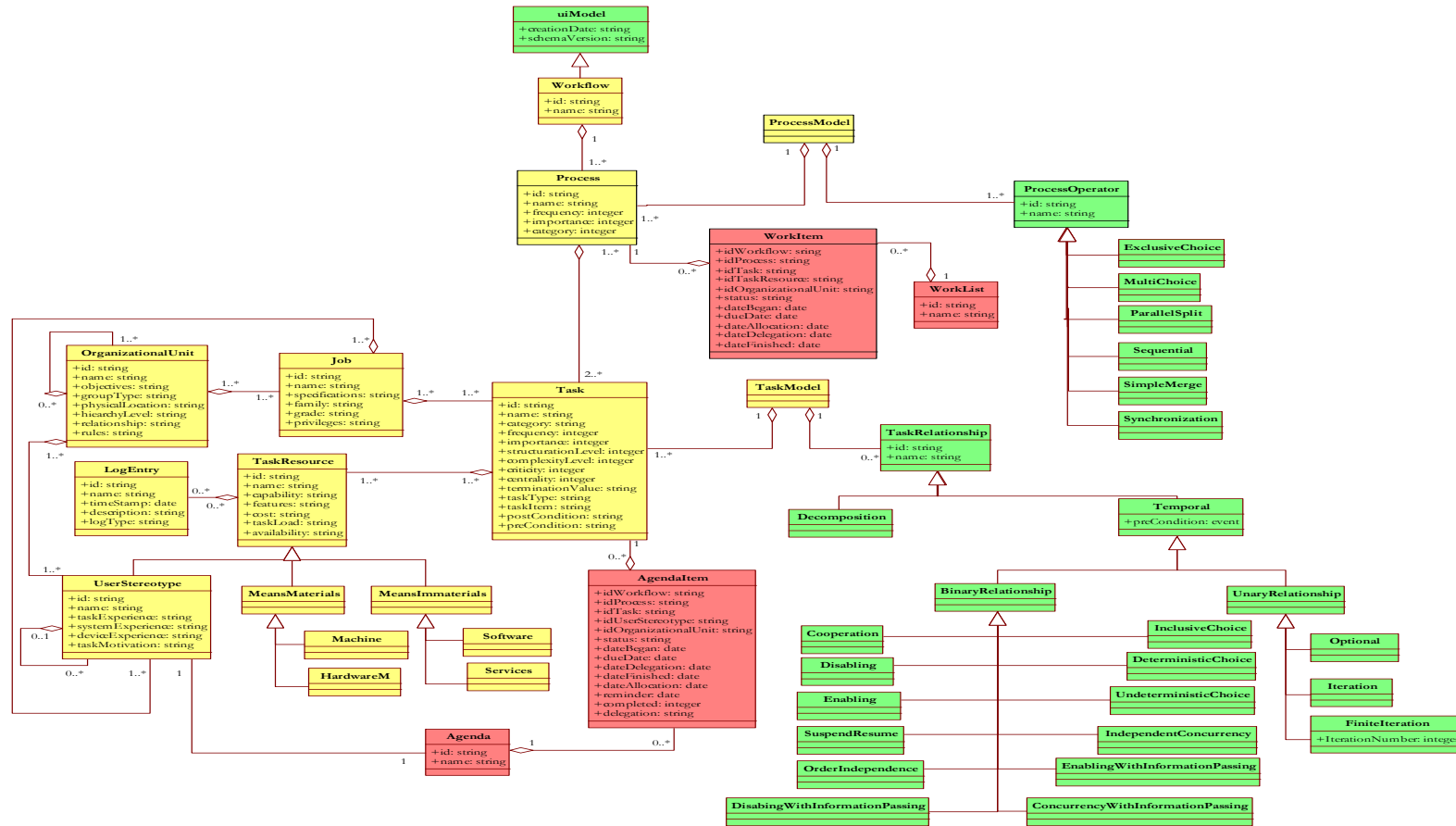


Figure 3-8. Workflow meta-model.

3.3 Syntax

Syntax refers to the ways symbols may be combined to create well-formed sentences (or programs) in the language. Syntax defines the formal relations between the constituents of a language, thereby providing a structural description of the various expressions that make up legal strings in the language [Slon95]. Syntax deals solely with the form and structure of symbols in a language without any consideration given to their meaning.

The syntax of the UsiXML language has an XML-based format structure, which allows describing sets of data with a tree-like structure [Stan08]. Figure 3-9 illustrates how the ontological concepts defined in the previous section are transformed in a UsiXML specification, which considers XML Schemas [W3C01] for the definition of valid XML elements. For this purpose manual transformations (T1) are applied in order to produce UsiXML XML Schemas from the UML class diagram description. Objects resulting from the instantiations of class diagram concepts are further transformed (T2) into UsiXML specification. Finally, the UsiXML specification is validated by the corresponding XML Schema.

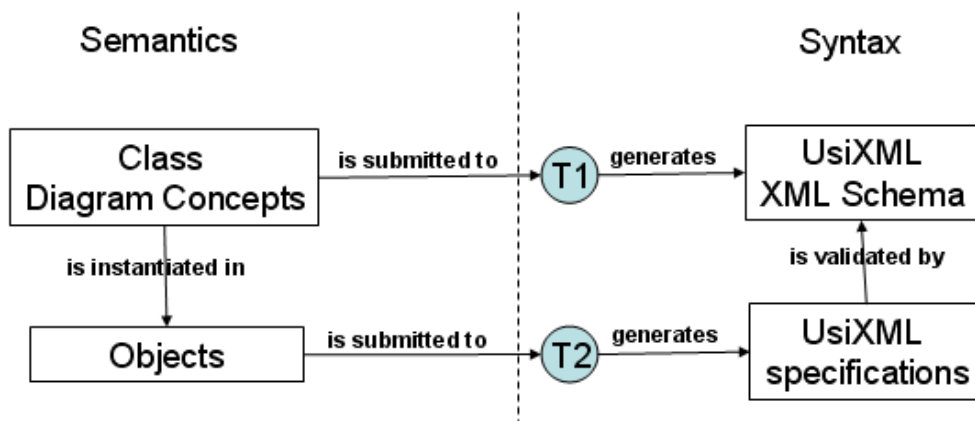


Figure 3-9. Generation of UsiXML specification.

In the following figures we illustrate how instances of a set of class diagram concepts are submitted to transformations T2 in order to obtain UsiXML specification based on the transformations catalog of UsiXML [USIX07]. Defining all the transformations needed for every single aspect related to the UI development life cycle, including even documentation generation, is one of the goals of UsiXML. In order to give the reader an understanding of how this process is done we present the mappings:

- A class becomes an XML element and class attributes become XML attributes: Figure 3-10 exemplifies how an instance of the *UserStereotype* class is mapped into an XML element with the associated attributes.

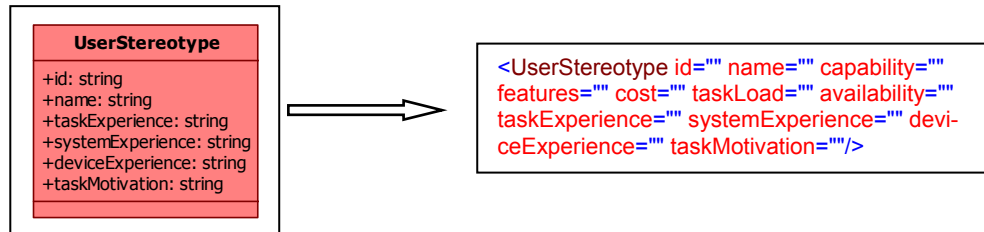


Figure 3-10. From UML class to UsiXML specification.

- *Aggregate relationship* corresponds to an XML structure where the client class and the supplier class are transformed into XML elements according to the example provided in Figure 3-10. The XML element generated from the client class embeds the XML element generated from the supplier class. Figure 3-11 exemplifies how an instance of a client class (i.e., *Process*) and two instances of a supplier class (i.e., *Task*) are transformed into XML elements. The *Process* element will embed the two *Task* elements. UsiXML takes advantage of the XML document structure and allows to derive implicit relationships between objects.

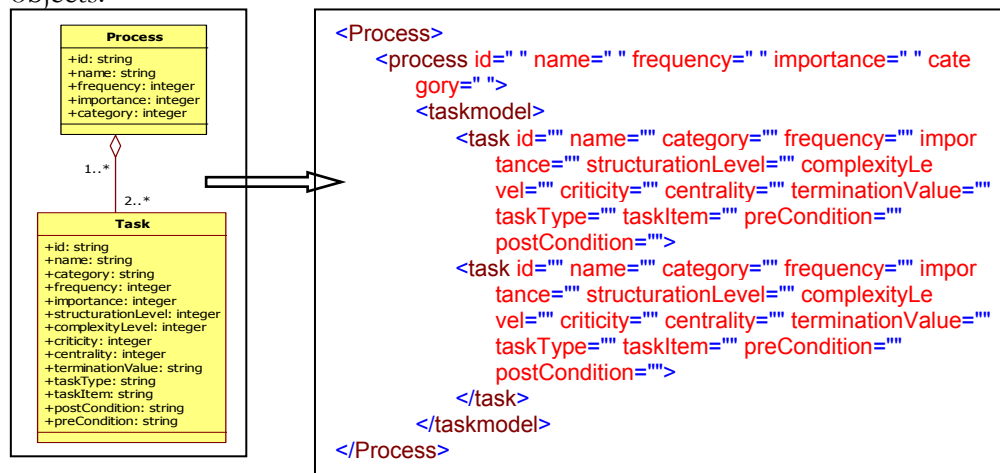
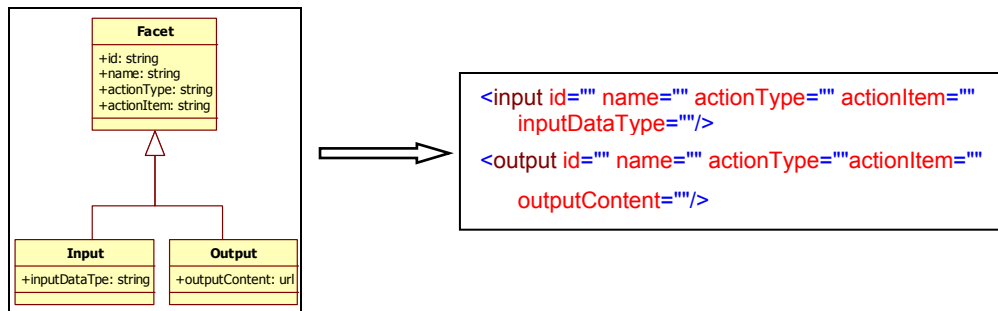


Figure 3-11. Transformation of the aggregation relationship into UsiXML specification.

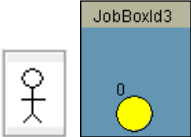
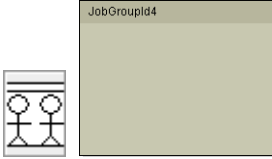
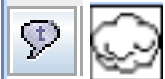
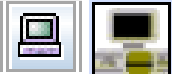
- Inheritance relationship class is transformed into an XML element for which the value of the type attribute takes the name of the subclass. In addition, the attributes of the subclass become XML attributes of the created element.

- Figure 3-12 presents two objects of two different classes (i.e., input and output) that inherit attributes from the same superclass (i.e., facet). For each object an XML element named facet is created. The attributes of the subclass instances (i.e., the inputDataType and outputContent) become XML attributes of the corresponding facet element.



Model	Graphical representation	Rationale
Process	Place 	The process model is represented using Petri Nets notation (Chapter 4). Place representation conserves the original icon used in Petri Nets.
	Transition 	It is used to model a Petri Net, it conserves the original icon.

Chapter 3. Conceptual model of workflow

	Arc 	It is used to model a Petri Net, it conserves the original icon.
Organizational	Organizational unit 	It represents the organizational place where the task is developed, the depiction is twofold: a container with the 3 elements of the Petri Net for the tool bar, and a container to group the different parts of the process (transitions and arcs) which are executed in the same organizational unit.
	Job 	It represents a function performed by a resource, the representation is twofold: a human drawing for the toolbar and a container showing the number of resources available for executing the task for the drawing area.
	Job group 	It represents the group of various jobs, the representation it is twofold: two humans drawing for the toolbar and a container to group the jobs for the drawing area.
Task	Abstract task 	It represents the abstract type of the task's category, it conserves its original design defined in [Mont06] [Pate99]
	System task 	It represents the system type of the task's category, it conserves its original design defined in [Mont06] [Pate99]
	User task 	It represents the user type of the task's category, it conserves its original design defined in [Mont06] [Pate99]
	Interactive task 	It represents the interactive type of the task's category, it conserves its original design defined in [Mont06] [Pate99]

Chapter 3. Conceptual model of workflow

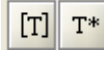
	Cooperative task 	It represents the cooperative type of the task's category, it conserves its original design defined in [Mont06] [Pate99]
	Operators: choice, order independency, concurrent, concurrent with information passing 	They represent the relationships among tasks; they conserve its original icon from [Pate99].
	Operators: disabling, suspend/resume, enabling, enabling with information passing 	They represent the relationships among tasks; they conserve its original icon from [Pate99].
	Operators: optional, iterative 	They represent the relationships among tasks; they conserve its original icon from [Pate99].

Table 3-1. Stylistics for interaction objects.

3.5 UIDL for the conceptual model

As overviewed into the previous chapter, there is a plethora of user interface description languages that are widely used. One objective of the current dissertation is to offer designers the capability of developing user interfaces for workflow information systems. In software engineering, model-based approaches rely on the power of models to construct and reason about information systems.

The goal of these approaches is to propose a set of abstractions, development processes and tools that further enable an engineering approach for UI development. In order to achieve this goal a UIDL is desirable. After a comparison of the UIDLs in [Guer09a], even that the conceptual model describe above can be incorporated to UIML, XICL or XIIML languages, we have considered that UsiXML is the suitable language that could accommodate those new concepts (i.e. workflow concepts) in a flexible way because:

- Its documentation is available including its meta-models and deep analysis can be done.
- It has a unique underlying abstract formalism represented under the form of a graph-based syntax.

Chapter 3. Conceptual model of workflow

- UsiXML allows reusing parts of previously specified UIs in order to develop new applications. This facility is provided by the underlying XML syntax of UsiXML which allows the exchange of any specification. Moreover, the ability to transform these specifications thanks to a set of transformation rules increases their reusability.
- It is structured according to the four basic levels of abstraction defined by the Cameleon reference framework [Calv03]. The framework represents a reference for classifying UIs supporting multiple target platforms and multiple contexts of use and enables to structure the development life cycle into four levels of abstraction: task and concepts, abstract UI (AUI), concrete UI (CUI) and final UI (FUI). The identification of the four levels and their hierarchical organization is built upon their independence with respect to the context in which the FUI is used. Thus, the Task and Concepts level is computational-independent, the AUI level is modality-independent and the CUI level is tool-kit-independent.

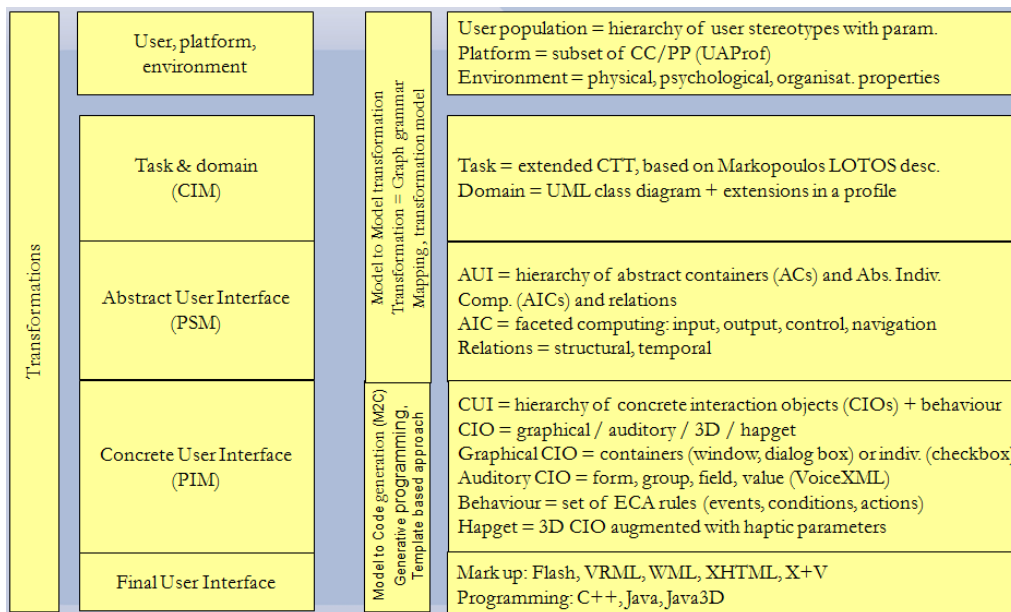


Figure 3-13. UsiXML profile based on Cameleon UIDL layer.

- UsiXML relies on a transformational approach that progressively moves from the Task and Concept level to the FUI.
- UsiXML supports the incorporation of new interaction modalities thanks to the modularity of the framework where each model is defined independently

Chapter 3. Conceptual model of workflow

and to the structured character of the models ensured by the underlying graph formalism.

- It is supported by a collection of tools that allow processing its format.
- It allows cross-toolkit development of interactive application thanks to its common UI description format.
- UsiXML ensures the independence of modality thanks to the AUI level which enables the specification of UIs that remains independent of any interaction modality.

3.6 Chapter summary

The current chapter presented how UsiXML, a UIDL to support our model-based approach, has been extended in order to respond to the requirements of workflow systems. A time line shows the evolution of UsiXML and the contribution from this dissertation (Figure 3-14).

FlowiXML [Guer08a] supports workflow information systems, which are advocated to automate processes, following a model-driven engineering based on requirements and processes of the organization. It applies to: 1) integrate human and machines based activities, in particular those involving interaction with IT applications and tools; 2) to identify how tasks are structured, who perform them, what their relative order is, how they are offered or assigned, and how tasks are being tracked. A workflow is recursively decomposed into processes that are in turn decomposed into tasks; each task gives rise to a task model.

In particular, a set of workflow attributes and the relationships between them have been introduced along with their semantics and stylistics.

With workflow model is possible to make a rough sketch of the task in order to show up the “What to do?” that is to say the main treatments, the main data, and the resources (human and system) which are necessary in order to accomplish the task. The relevant importance is the manner in which tasks are assigned to resources; we present a method for developing workflow user interfaces in next chapter.

Chapter 3. Conceptual model of workflow

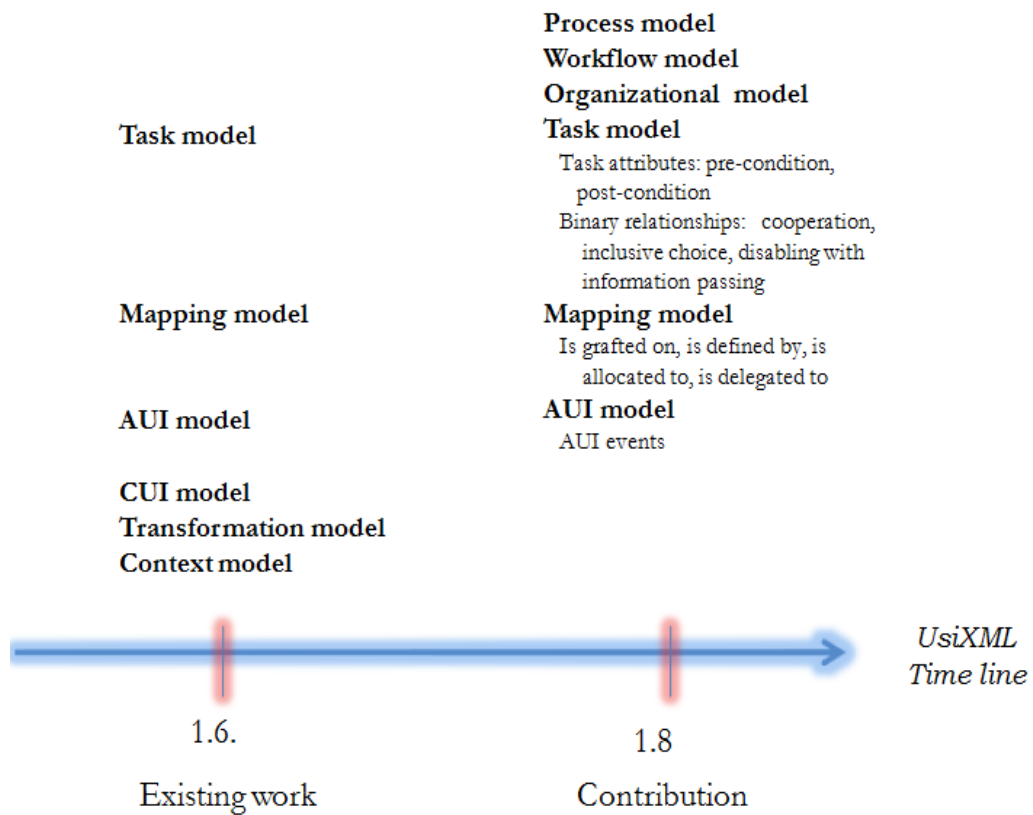


Figure 3-14. Time line of contributions to UsiXML.

Chapter 3. Conceptual model of workflow

Chapter 4 Developing workflow user interfaces

4.1 Context

Once we have presented the model and the language design artifacts, this chapter focus on describing the method for developing workflow user interfaces (Figure 4-1).

Developing UIs for WfIS represents many challenges because the user interaction takes place in two different logical levels synchronously. At the higher level, the workflow manager needs user interfaces to monitor (user interface for the work-list) the workflow execution and to allocate work (user interfaces for resource allocation patterns). At the lower level, the workflow users are carrying out their allocated interactive tasks (they need user interfaces for that purpose) whose current status (user's agenda) is then communicated to the workflow manager (user interface for the worklist).

These results into two UI categories: UIs for the end user (referred to as the *userStereotype*) and UIs for the workflow manager. There is some dependence between these two categories: any change of state of the worker UI should be reflected into the workflow manager UI. For instance, when a worker has finished a task, this information is propagated to the workflow UI. Similarly, when a task should be offered to a single or multiple resources, or delegated, the user's agenda UI is informed.

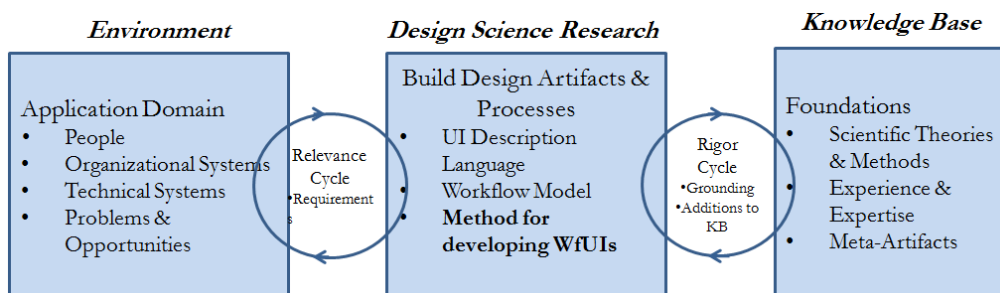


Figure 4-1. Schema for method derivation (adapted from [Hevn07]).

4.2 Method for specifying workflow information systems

Our method provides mean to specify a WfIS and some guidance on how to derive its corresponding user interfaces. It is composed on the following major steps:

1. Workflow information system requirements. This is the result of the elicitation of the organization. We assume that there are means such as: interviews, direct observation, to collect information that will serve as input to identify workflow element. This step corresponds to the requirements of the problem.
2. Workflow information system design. This step includes modeling of: workflow, organizational units, jobs, user stereotypes, processes, workflow allocation patterns and tasks. Mapping the workflow specification into a workflow information system.
3. Workflow information system development. We consider the development of UI for: task models, allocation patterns, agendas, worklist. Even that it is not explicitly defined we considered that the implementation of a workflow manager is possible based on the workflow designed in the previous step.

In Figure 4-2, forward and backward arrows denote the propagation of information from one model to another. For instance, a new task model must make available a task for a process model and vice versa, a new task in a process model might be detailed with a task model. Jobs, user stereotypes and organizational modeling just affect the workflow model. Then the workflow model makes them available for process modeling and task modeling. This particular aspect of concepts propagation was significantly useful for the software tools that support our methodology (see next chapter).

The system design is an activity that can start from any model except for the task allocation (dash lines model) because it needs tasks and resources already defined. The design of a workflow permits designers to identify concepts freely and to start to detail based on their preferences. One designer must prefer to get into details of task modeling before describing a process model. Once the task models are ready then it can model the processes that then are arranged to represent the workflow. Another designer might have a better understanding of the problem with the workflow model (more abstract view of the problem) and then start to refine by adding process models and finished with task models. There is no constraint on the starting and end point for modeling just to be sure keep the tracea-

bility of the concepts that are shared in different models (task model is part of process and a process model is part of a workflow model).

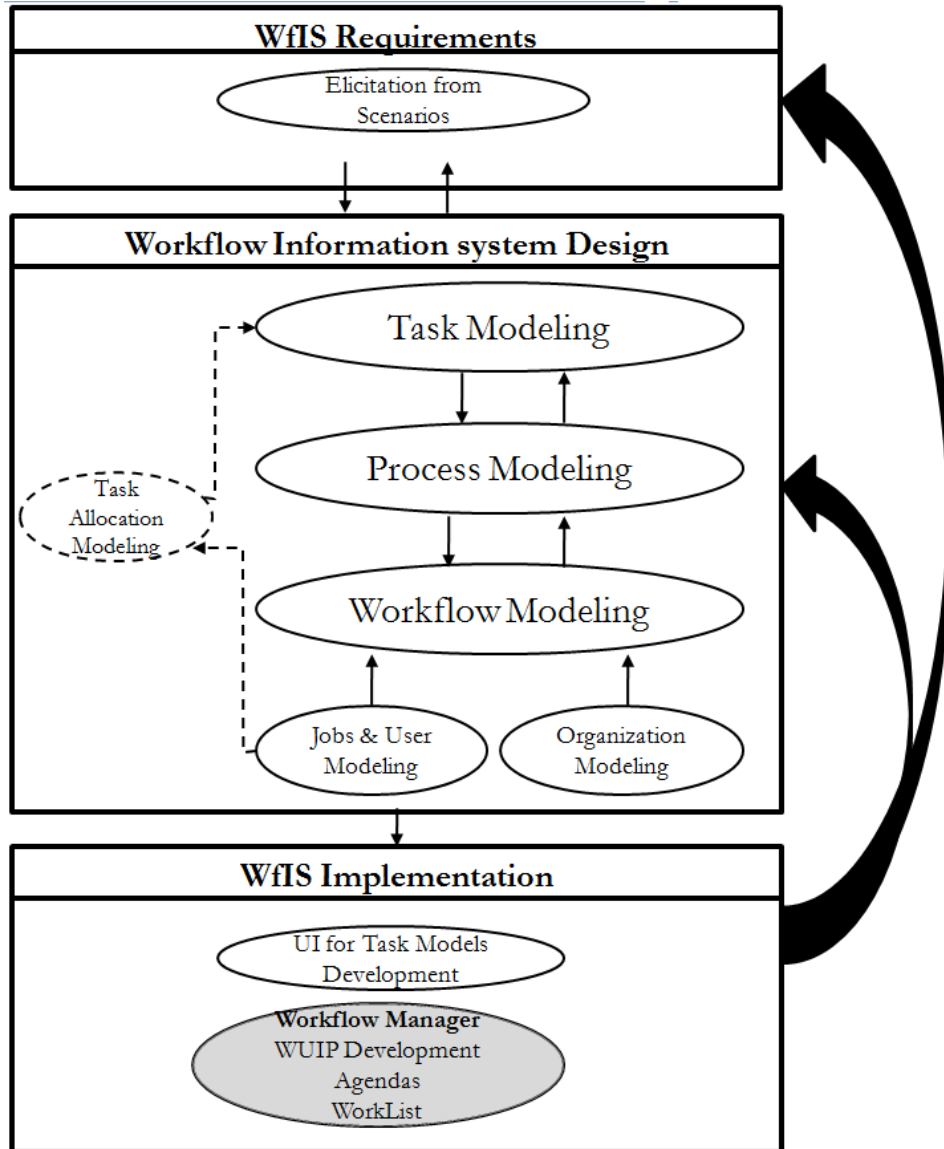


Figure 4-2. Method overview.

The implementation of the system refers to the UI generation for tasks. We rely on UI generation from task models techniques [Vand05b] for deriving UI. As these techniques uses task model, among other models, as a source to develop UIs, we model the allocation patterns with task models to derive Workflow User Interface Patterns (WUIPs). Finally we identify potential information relevant for the implementation of the workflow manager with the UI flow.

Chapter 4. Developing workflow user interfaces

The process is iterative; once the implementation is done the system can be refined starting from the requirements or the design. The modeling activities also are iterative. It is important to mention the evaluation of the final implemented system is not in the scope of this dissertation. As many methods exist for this purpose and depending of the context of use they are used. We just assumed that in some way the system will be tested and if there is a need to iterate then there is a point back into the requirements definition and the design of the workflow information systems.

Finally, a complete specification of the method using the Software Process Engineering Meta-model (SPEM) notation (Figure 4-3), promoted by the OMG [OMG07], is presented in this chapter. This notation, apart from being formal, provides guidance on the use of the method. This notation has been recommended to express methods that adhere to the Model-Based User Interface Development Approach.

				
Package	Task Definition	Role Definition	Guidance	Work Product
It contains elements as Work Product, Task Definition, Role Definition, etc.	It defines work being performed by Roles Definition instances.	It defines a set of related skills, competencies, and responsibilities. Roles are used by Task Definitions to define who performs them as well as to define a set of Work Product Definitions they are responsible for.	It provides additional information related to Describable Elements. It should be classified as guidelines, checklists, reports, etc.	It is used, modified, and produced by Task Definition

Figure 4-3. Some SPEM concepts.

4.2.1 Workflow information systems: elicitation from scenarios

In a user-centered design it is expected to identify the end-user and get their description for the work they do. This early understanding of the work, which is carried out in the organization, requires constant meetings with end user to understand their tasks in their context (environment, available resources). During the stage of system requirements gathering, model elicitation is aimed at identifying in textual scenarios elements that are relevant for building a first version of models

that will be further exploited in our method.

Scenarios have the advantage to describe UI requirements from captured or imagined user interactions through concrete examples [Garl01] of the user carrying out her task. This form is much more representative and evocative for an end user to validate UI requirements than models that are mainly used by software engineers. Models, e.g., domain models, user models, are expressed in a way that maximizes desirable properties such as completeness, consistency, and correctness [Vand05b]. But their expression is significantly less understandable for end users who are often in trouble of validating their UI requirements when they are confronted to models. Consequently, both types of descriptions, scenarios and models, are needed interchangeably in order to conduct a proper Requirement Engineering (RE) process that will effectively and efficiently feed the rest of the UI development life cycle. We introduce model elicitation as the activity of transforming textual scenarios into models that are pertaining to the UI development.

The WfIS requirements step (Figure 4-4) is concerned with the understanding of a problem by studying an existing organizational setting; the emphasis is put on identifying the elements involved in the business process description following identification criteria. The output of this phase is an organizational model, i.e., lists for: task, job, organizational unit, resource, which includes relevant actors and their respective tasks. This step uses as guidance three documents: task-meta-models definitions if there a need to look for a definition of a concept; task identification criteria that to identify tasks in the scenarios; and the most relevant work describing this step has been described in [Lema08].

On practical bases (Figure 4-5), the domain expert (end-user) describes the business process, from this scenario the workflow designer identifies the tasks, the resources in charge to develop them, the unit where they are executed, and so on. After, he produces a classification of these concepts, which will be validated for the domain expert (end-user, wf manager, and supervisor).

4.2.1.a Identification criteria

One important and recurrent element that is of our interest is the concept of task. We identified [Guer08g] a set of criteria to identify the concept of: task, process and workflow. During the practical experience of using the methodology, we identify and provide a solution to the following question: How from a textual scenario a task can be identified? We looked at four dimensions surrounding the task execution (i.e., time, space, resources, and information). Any variation of any of these four dimensions, taken alone or combined, thus generates a potential identification of a new concept.



Figure 4-4. Requirements elicitation package.

We focus on the task identification using the following identification criteria [Guer08g]:

1. Change of space (or change of location): when the scenario indicates a change of location of the operations, a change of task occurs. Therefore, any scenario fragment like “in the headquarters, the worker does ..., and then in the local agency, the worker does...” indicated a change of space, therefore a change of task.
2. Change of resource: when the scenario suggests that new or different resources are exploited, a change of task occurs. We distinguish three categories of resources:
 - a. Change of resource of type “User stereotype”: when another user stereotype appears in the scenario. For example, “a clerk does ..., then an employee files the results of ...” The two different names for two different

users indicate a change of the user in charge of executing the task. Maybe they are collaborating but we need explicitly to state their role in such interaction.

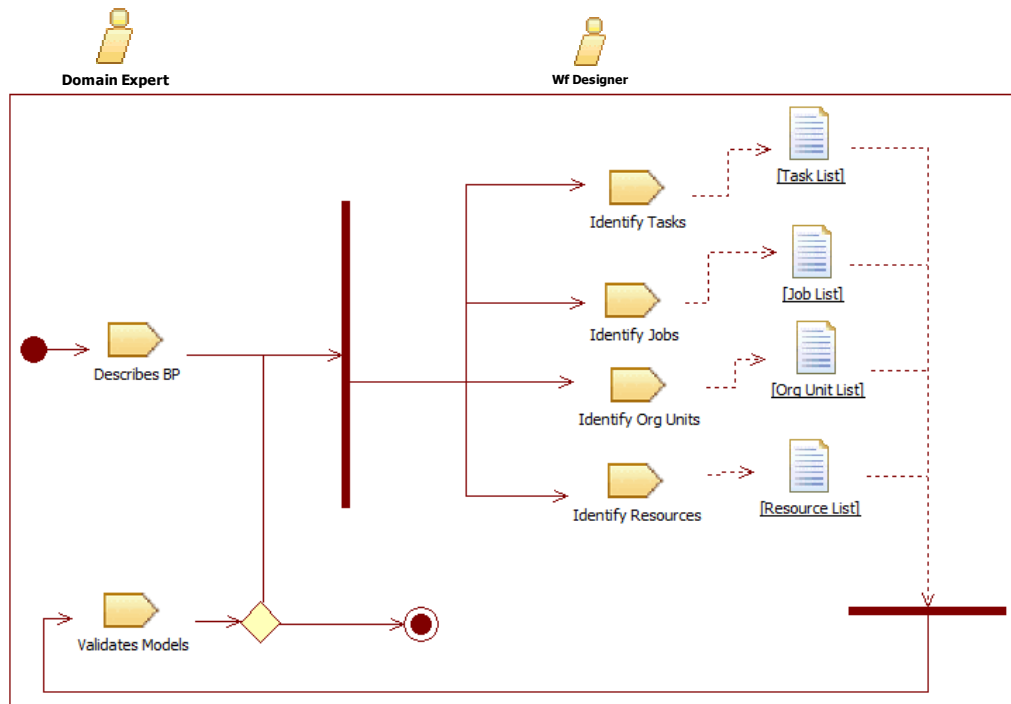


Figure 4-5. Requirements elicitation activity diagram.

- b. Change of resource of type “material”: when another material resource appears in the scenario. For example, “a clerk enters the customer’s data on a Pocket PC, and then takes a picture with a mobile phone camera” indicates two tasks resulting from the usage of two different resources, here a Pocket PC and a mobile phone. This should not be confused with a task that is performed on different computing platforms, like in the context of a multi-device UI.
 - c. Change of resource of type “immaterial”: when another immaterial resource appears in the scenario. For example, “a network administrator uses specific software to check network status; s/he uses other software to update the computers of the network”. The two different types of software involved indicate a change of task.
3. Change of time: when the scenario indicates a different time period in which the task is performed. We differentiate four criteria:
 - a. Existence of an interruption: when the task is interrupted by an event that changes the time period. For instance, “an employee registers every incom-

ing complaint. After registration a form is sent to the customer who returns the form within two weeks”.

- b. Existence of a waiting point: when in the development of a task there is a moment where is necessary to wait that something occurs for continuing. We have two types of waiting points:
 - i. Waiting point of type “decision”: when a determination arrives at after consideration. For example, “after the preparation of a flight plan, the pilot will take the decision to fly”.
 - ii. Waiting point of type “accumulation”: when there is necessary to create a waiting list. For instance, “due to a car accident, more complaints arrived yesterday at the insurance agency and the employee had to register all incoming complaints to send as a group to directors”.
 - c. Permanence of execution unit: when the task execution depends of the results of at least two previous asynchronous tasks. For instance, “the results of an insurance complaint are delivered to the client when the complaint manager provides whether the complaint applies or not and when the evaluator provides the estimated cost”.
 - d. Periodicity of execution: when there is a periodicity established to execute tasks. For example, “every Monday the employee does a backup of the information”.
4. Change of nature: when the scenario represents a change of category a change of task occurs. For instance, “first a secretary types a letter in the computer (interactive), after a printer prints the text (automatic) and finally the manager signs the letter (manual)”.

	Time	Space (location)	User Stereotype
Workflow	Series of time periods	Different locations	Different groups of resources
Process	Series of time periods	Same location	One resource or a group of resources
Task	Same time period	Same location	Same resource

Table 4-1. Identification criteria.

At first everything is classified as tasks, and then comes the problem of *How to group tasks, processes and workflow? Where do they belong?* A task could be part of a process model or a task model. Existing knowledge on task identification criteria is again relevant to make such separation. Table 4-1 summarizes these identification criteria:

1. A task model is composed of tasks performed by the same resource, in the same location and in the same time period. The reader must not be confused by the fact that during the execution of the task one of these properties might be changed. For instance, the task “insert client information” could be suspended during the execution of the task, for instance, while answering the phone, consequently when the user resumes the execution of the task the time period changes, it is no longer the same. However, this does not affect nature of the task time periodicity this kind of external events are part of the task life cycle (Figure 4-9). Location and change in user stereotypes is something evident to identify but the time periodicity might be tricky to discover. If the designer confronts difficulties we propose the use of a time line to gather tasks just to be sure that they are executed in the same time series.
2. A process model is composed of tasks performed by one resource, or one or more group of resources; the location is the same; the time periodicity changes.
3. Finally a workflow is composed of processes performed by one or more group of resources; located in different organizational units (within the same or different organizations); and the time periodicity of the processes changes.

4.2.2 Workflow information systems design

The WfIS design (Figure 4-6) is composed of several models (workflow, process and task) and several actors are involved (Wf designer, end-user, domain expert and Wf manager). The first activity corresponds to the consolidation of the concepts identified in the elicitation of the scenario.

The Wf designer is able to start the modeling (Figure 4-7) following:

- Top-down approach, i.e., starting with modeling the high level view of the problem (workflow model), then detailing the workflow models by adding process into the different organizational units. Finally a process model is refined by adding a task model on each task.
- Bottom-Up approach, i.e., starting with low level details (task modeling) and from that, starts to build process models and workflow models.
- Middle out approach, i.e., starting with middle level details (process modeling) then going to details with task modeling, and high level description with workflow.

Chapter 4. Developing workflow user interfaces

The end-user is the responsible of validating task models. Notice by end-user we understand the person who actually is in charge of performing this task, i.e., the most qualified to say something about it. The process is validated by the domain expert. This is due to the fact that a process requires a higher level of understanding of the problem. In this view it could be any person in the organization that through the requirements analysis has been identified to be most familiar with the whole process modeled.

Finally the workflow model is validated by the Wf manager, who is the person or group of persons with an understanding of the whole workflow when processes are grouped. It is also, the Wf manager who is in charge of allocating tasks in a process to resources. The final result is the WIS design. All these design activities are also accompanied with guidance for the modeling activities (Figure 4-6). The next sub-sections details the modeling activities, as depicted in Figure 4-7 and already stated before, the ordering of the next sub-section, describing the modeling activities, does not follow any recommended order for the modeling activities. We aimed at answering with the modeling activities to the questions related to the work done in the organization: what to do? How to do it? Where to do it? Who will carry out it? Whom?

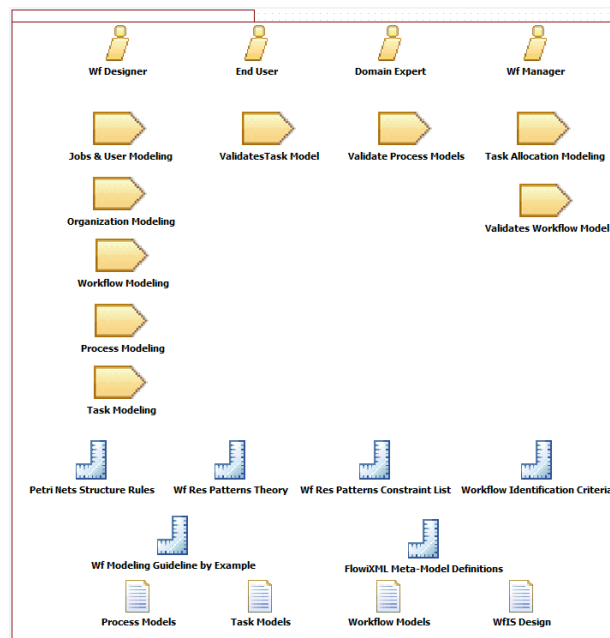


Figure 4-6. Workflow information system design package.

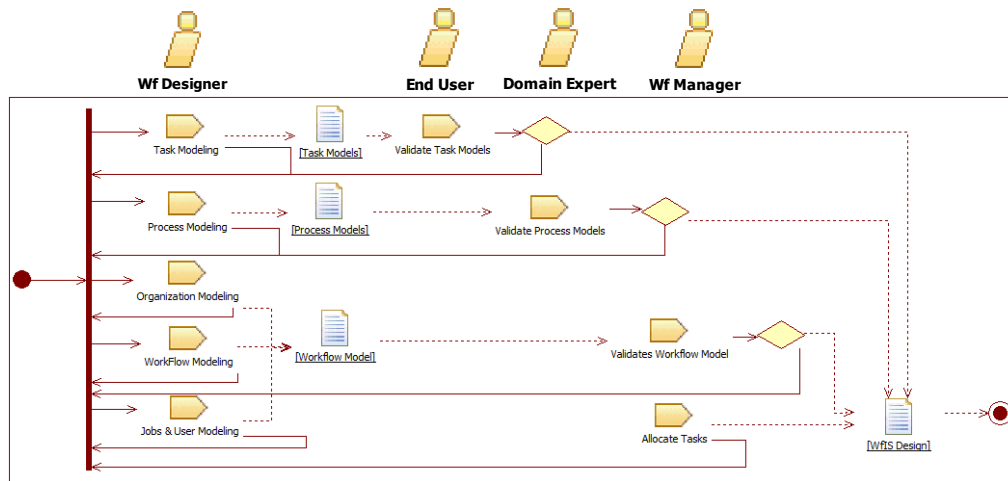


Figure 4-7. Workflow information system design activity diagram.

4.2.2.a What to do? Processes specification

The definition of a process indicates which tasks must be performed and in what order. Thus answering the question what to do? After having identified tasks that are part of a process then they have to be related to each other by means of process operators.

We propose the use of Petri Nets notation for modeling processes. Accordingly with [vand02], the use of Petri Nets has a number of major advantages when used for modeling process: (1) ambiguities and contradictions are prevented; (2) the formalism can be used to argue about process; (3) it thus becomes possible to establish certain patterns over the operators.

Three concepts are relevant for a process model: place (process state), transition (task), and process operators, this is known as a WF-net [vand98]. It has a unique source place (i) and a unique sink place (o). This corresponds to the fact that any case handled by the process description is created if it enters the WFMS and is deleted once it is completely handled by the WFMS. A *token* in the source place Y corresponds to a case which needs to be handled, a token in the sink place β corresponds to a case that has been handled. The process state is defined by the marking. In addition, a WF-net requires all nodes (i.e. transitions and places) to be on some path from i to o . This ensures that every task (transition) or condition (place) contributes to the processing of cases.

When doing process modeling, it is important to specify some starting and stopping points:

- The process starts when an input (time, human or message) triggers the execution of the process.
- An output can be sent when the process finishes (completed, aborted or terminated) its execution.

As guidance for process modeling, designers must rely on: Petri Nets Structure Rules [Vand98], Identification Criteria (4.2.1.a), the WF Modeling Guideline by Example material available through FlowiXML website, and FlowiXML Meta-Model definitions are useful for attributes definition, process operators, and any other theoretical aspect of interest.

4.2.2.b *How to do it?* Tasks specification

For each task in a process a task model can be specified, not necessarily, to describe in detail how the task is performed, answering the question of how to do it? By exploiting task model descriptions different scenarios could be conducted. Each scenario represents a particular sequence of actions that can successfully be performed to reach a goal. Task models do not impose any particular implementation so that user tasks can be better analyzed without implementation constraints. This kind of analysis is made possible because user tasks are considered from the point of view of the users need for the application and not how to represent the user activity with a particular system.

Our task model (Figure 4-8) represents a decomposition of tasks into sub-tasks linked with task relationships. It is an extended version of UsiXML task modeling [Guer08a] and compliant with the graphical notation of CTT [Pate00].

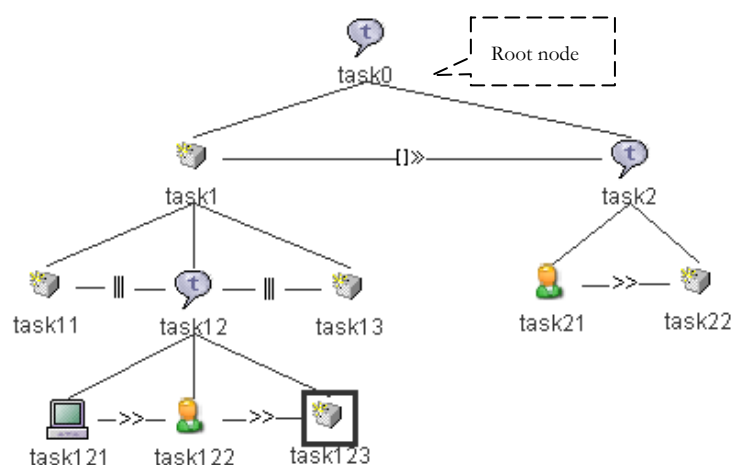


Figure 4-8. Task model representation.

Introducing task models description to the process model correspond, but is not limited, to the following reasons:

- Task models describe, opposed to process models, end users' allowing describing how a task is performed.
- It is true that in a process model we can add the detail desired, with process hierarchies, to represent a detailed task description. Nevertheless, we consider that specific temporal operators (iteration, suspend/resume) applied to task, can be more naturally defined in a task model rather into a process model, that implies the creation of dummy transitions.

When doing task modeling, it is important to decide how far the decomposition of tasks is to proceed. This depends of course of the context and purpose of task modeling; however some stopping criteria based on the task life cycle (Figure 4-9) are:

- For *horizontal stopping*: when the task is finished, or the task is canceled or the task failed.
- For *vertical stopping*: when a task can be performed in a simple and well-determined way (i.e. the task cannot be decomposed in sub-tasks), when the task is executed by a software system and we do not intend to replace this system with anything else.

4.2.2.c *Where to do it?* Organizational units specification

We introduced an organizational unit concept; it describes the places where work is carried out. It can be a physical chamber in a builder (e.g. an operating room), but it can also be a complete floor in a hospital. The elements corresponding to the actual organizational structure in a specific organization are specified during the organization design. This part contributes to UI adaptation to different categories of users and security of IS by blocking access to UIs when the user does not have the permission to perform the task.

4.2.2.d *Who will carry out it?* Job and user specification

This step consists in the description of the resources that are capable of doing work. In addition, a resource may have one or more associated jobs. Jobs serve as another grouping mechanism for resources with similar roles or responsibilities. A resource is a member of an organizational unit; he is owner of an agenda which using to manage the whole range of his tasks.

4.2.2.e Whom? Applying workflow resource patterns

Actual assignment of tasks to resources is performed according to workflow resource patterns (Appendix E). They are applied to different steps of the task life cycle: *creation*, *allocation*, *in execution*. Without considering the moment in which they are applied, designers must define how to allocate work to available resources.

We extended Russell [Russ05] task life-cycle (Figure 4-9) from the time that it is created to its completion (or failure or cancel).

Tasks have been defined before trying to allocate a resource to it. The task life-cycle determines what pattern can be used depending on the task status, thus the interest of presenting the task life cycle. Once a task has been properly defined, it comes into existence in the *created* state. At this point, a task could be effectively *offer* or *allocate*; a task is said to be *started* when the human resource to which the task has been allocated has initiated its execution. A task may be allocated to such resource, but it starts later on. If this allocation is not straightforward, the task can be offered to a single resource or to multiple resources. Once a task is allocated, it could be *delegated* to another resource (e.g., due to unavailability). If the resource which delegated the task wants to receive the results in return, the task is then returned. Otherwise, it can start directly. Subsequent states in the task distribution are *started*, which indicates that a resource has commenced executing it; *suspended* which denotes that the resource has chosen to cease execution of the task for a period, but does intend to continue working on it at latter time; *failed* which identifies that the task cannot be completed; *cancel* which identifies that the resource, by any reason, will not work on it any further; *completed* which identifies a tasks that has been successfully executed; and *finished* which identifies when the goal is reached.

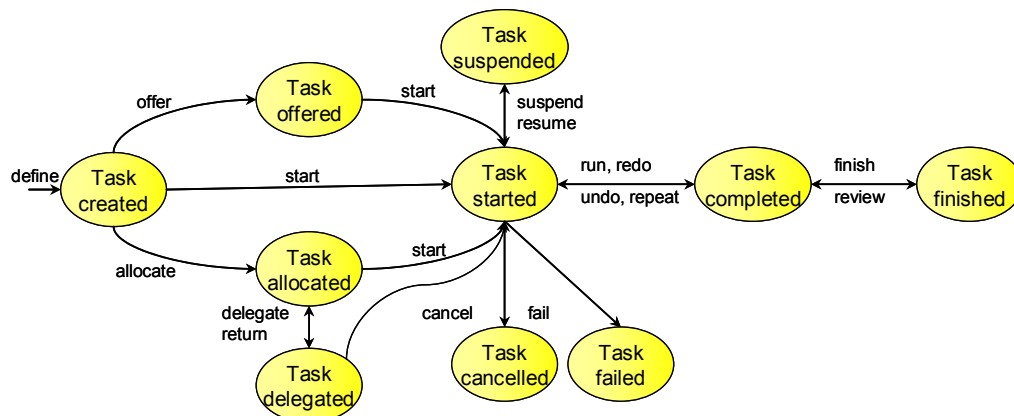


Figure 4-9. Task life cycle.

both, so these patterns are mutually exclusive. The column named $\mathcal{A}/\mathcal{O}$ on the table refers to this values, $\mathcal{A}$ for allocation and $\mathcal{O}$ for offering.

Table 4-3 and Table 4-4 show the possible constraints that we can confront when a pattern is chosen.

About the notion of design and run time makes the distinction when the pattern will be applied or assigned, this is depicted on the table with the column named *Design-Time*, where there is no X means that the pattern is applicable just at run-time. The table is read as follows: if a pattern Y is going to be applied, then you have to check if there is no other pattern Z assigned that constraints its use. For instance, if we consider *direct allocation* pattern (1) being assigned, this pattern just can co-exist with patterns: 22, 30, 32, and 34 when it refers to offering (Table 4-3); and with patterns 3, 4, 5, 6, 7, 8, 9, 22, 30, 33, 34, 35, 36, 41 and 42 when it refers to allocation.

Chapter 4. Developing workflow user interfaces

	Pattern	Design-Time	Run-Time	User Agenda	WF Server	Properties of the Agenda	A / O
1	Direct Allocation	X					A
2	Deferred	X					
3	Authorization based	X					A/O
4	Separation of duties	X					A/O
5	Case handling	X					A/O
6	Retain familiar	X					A/O
7	Capability based	X					A/O
8	History based	X					A/O
9	Hierarchy level based	X					A/O
10	Distribution by offer single resource	X					O
11	Distribution by offer multiple resource	X					O
12	Distribution by allocation single resource		X				A
13	Early distribution		X				A
14	Distribution on enablement		X				A
15	Late distribution		X				A
16	Random allocation	X					A/O
17	Round robin allocation	X					A
18	Shortest queue	X					A
19	Resource initiated allocation		X	X			
20	Resource initiated execution - allocated task		X	X			
21	Resource initiate execution - offered task		X	X			
22	System determined agenda queue content	X					
23	User determined agenda queue content		X				
24	Selection autonomy		X	X			
25	Escalation		X		X		A/O
26	Deallocation		X	X			O
27	Stateful reallocation		X	X			O
28	Stateless reallocation		X	X			O
29	Suspension / resumption		X	X			
30	Skip	X	X	X			
31	Redo		X	X			
32	Pre-do		X	X			
33	Commencement on creation	X	X	X			
34	Commencement on allocation	X					
35	Piled execution	X					A/O
36	Chained execution	X					A/O
37	Configurable unallocated task visibility					X	
38	Configurable allocated task visibility					X	
39	Simultaneous execution						
40	Additional resources		X	X	X		A/O
41	Delegation	X	X	X			A/O
42	Role based	X					A/O

Table 4-2. Classification of WF resource pattern.

Chapter 4. Developing workflow user interfaces

Pattern	Design-Time	A / O	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
1 Direct Allocation	X	A	X																					X								X			X	X								
2 Deferred	X		X																					X								X			X	X								
3 Authorization based	X	O	X	X	X	X	X	X	X	X	X							X						X								X			X	X	X	X				X	X	
4 Separation of duties	X	O	X	X	X			X	X	X	X							X						X								X			X	X	X	X				X	X	
5 Case handling	X	O	X	X		X	X	X	X	X	X							X						X								X			X	X	X	X				X	X	
6 Retain familiar	X	O	X	X	X	X	X	X	X	X	X													X								X			X	X	X	X				X		
7 Capability based	X	O	X	X	X	X	X	X	X	X	X							X						X								X			X	X	X	X				X	X	
8 History based	X	O	X	X	X	X	X	X	X	X	X							X						X								X			X	X	X	X				X	X	
9 Hierarchy level based	X	O	X	X	X	X	X	X	X	X	X							X						X								X			X	X	X	X				X	X	
10 Distribution by offer single resource	X	O										X												X								X			X	X								
11 Distribution by offer multiple resource	X	O											X											X								X			X	X								
12 Distribution by allocation single resource		A																																										
13 Early distribution		A																																										
14 Distribution on enablement		A																																										
15 Late distribution		A																																										
16 Random allocation	X	O		X	X	X		X	X	X								X						X								X			X	X	X	X				X	X	
17 Round robin allocation	X	A																	X					X								X			X	X								
18 Shortest queue	X	A																		X				X								X			X	X								
19 Resource initiated allocation																																												
20 Resource initiated execution - allocated task																																												
21 Resource initiate execution - offered task																																												
22 System determined agenda queue content	X		X	X	X	X	X	X	X	X	X	X	X					X	X	X				X								X			X	X	X	X				X	X	
23 User determined agenda queue content																																												
24 Selection autonomy																																												
25 Escalation		O																																										
26 Deallocation		O																																										
27 Stateful reallocation		O																																										
28 Stateless reallocation		O																																										
29 Suspension / resumption																																												
30 Skip	X		X	X	X	X	X	X	X	X	X	X	X					X	X	X				X								X			X	X	X	X				X	X	
31 Redo																																												
32 Pre-do																																												
33 Commencement on creation	X		X	X	X	X	X	X	X	X	X	X	X					X	X	X				X								X			X	X	X	X				X	X	
34 Commecement on allocation	X		X	X	X	X	X	X	X	X	X	X	X					X	X	X				X								X			X	X	X	X				X	X	
35 Piled execution	X	O						X	X	X	X							X						X								X			X	X	X				X			
36 Chained execution	X	O					X	X	X	X								X						X								X			X	X		X				X		
37 Configurable unallocated task visibility																																												
38 Configurable allocated task visibility																																												
39 Simultaneous execution																																												
40 Additional resources		O																																										
41 Delegation	X	O	X	X	X	X		X	X	X								X						X								X			X	X	X	X				X	X	
42 Role based	X	O	X	X	X	X	X	X	X	X								X						X								X			X	X	X	X				X	X	

Table 4-3. Constraints to choose offering patterns.

Chapter 4. Developing workflow user interfaces

Pattern	Design-Time	A / O	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
1 Direct Allocation	X	A	X	X	X	X	X	X	X	X	X	X												X								X			X	X	X	X				X	X	
2 Deferred	X		X																				X									X			X	X								
3 Authorization based	X	A	X	X	X	X	X	X	X	X	X	X						X	X	X				X								X			X	X	X	X					X	X
4 Separation of duties	X	A	X	X	X			X	X	X								X	X	X				X								X			X	X	X	X					X	X
5 Case handling	X	A	X	X	X	X	X	X	X	X								X	X	X				X								X			X	X	X	X					X	X
6 Retain familiar	X	A	X	X	X	X	X	X	X	X														X								X			X	X	X	X					X	
7 Capability based	X	A	X	X	X	X	X	X	X	X								X	X	X				X								X			X	X	X	X					X	X
8 History based	X	A	X	X	X	X	X	X	X	X								X	X	X				X								X			X	X	X	X					X	X
9 Hierarchy level based	X	A	X	X	X	X	X	X	X	X								X	X	X				X								X			X	X	X	X					X	X
10 Distribution by offer single resource	X	O										X											X									X			X	X								
11 Distribution by offer multiple resource	X	O											X										X									X			X	X								
12 Distribution by allocation single resource		A																																										
13 Early distribution		A																																										
14 Distribution on enablement		A																																										
15 Late distribution		A																																										
16 Random allocation	X	A		X	X	X		X	X	X								X						X								X			X	X	X	X					X	X
17 Round robin allocation	X	A		X	X	X		X	X	X									X					X								X			X	X	X	X					X	X
18 Shortest queue	X	A		X	X	X		X	X	X										X				X								X			X	X	X	X					X	X
19 Resource initiated allocation																																												
20 Resource initiated execution - allocated task																																												
21 Resource initiate execution - offered task																																												
22 System determined agenda queue content	X		X	X	X	X	X	X	X	X	X	X	X					X	X	X				X								X			X	X	X	X					X	X
23 User determined agenda queue content																																												
24 Selection autonomy																																												
25 Escalation		A																																										
26 Deallocation		O																																										
27 Stateful reallocation		O																																										
28 Stateless reallocation		O																																										
29 Suspension / resumption																																												
30 Skip	X		X	X	X	X	X	X	X	X	X	X	X					X	X	X				X								X			X	X	X	X					X	X
31 Redo																																												
32 Pre-do																																												
33 Commencement on creation	X		X	X	X	X	X	X	X	X	X	X	X					X	X	X				X								X			X	X	X	X					X	X
34 Commecement on allocation	X		X	X	X	X	X	X	X	X	X	X	X					X	X	X				X								X			X	X	X	X					X	X
35 Piled execution	X	A					X	X	X	X								X	X	X				X								X			X	X	X					X		
36 Chained execution	X	A					X	X	X	X								X	X	X				X								X			X	X		X				X		
37 Configurable unallocated task visibility																																												
38 Configurable allocated task visibility																																												
39 Simultaneous execution																																												
40 Additional resources		A																																										
41 Delegation	X	A	X	X	X	X		X	X	X								X	X	X				X								X			X	X	X	X					X	X
42 Role based	X	A	X	X	X	X	X	X	X	X								X	X	X				X								X			X	X	X	X					X	X

Table 4-4. Constraints to choose allocation patterns.

4.2.3 Workflow information systems implementation

The last step (Figure 4-11) consists on the implementation of the WfIS. We are primary interested in the UIs that are needed for the system rather than the functionality that are:

- Support user's tasks. These UIs are the result of a specification of user' tasks using task models. Thus this generates most of the UIs for the WfIS.
- Support user's communication with UI for user's agendas and manager's worklist that must be updated accordingly as users received a new item or when they finish a task. This category of UIs even that its design is not complex their functionality it is and communication protocols are out of the scope of this work.
- Support tasks allocation, resource patterns, by different means such as: allocation, offering, delegation. We draw attention on these UIs and provide a solution on how to develop them. We called this UIs workflow user interface patterns (WUIPs).

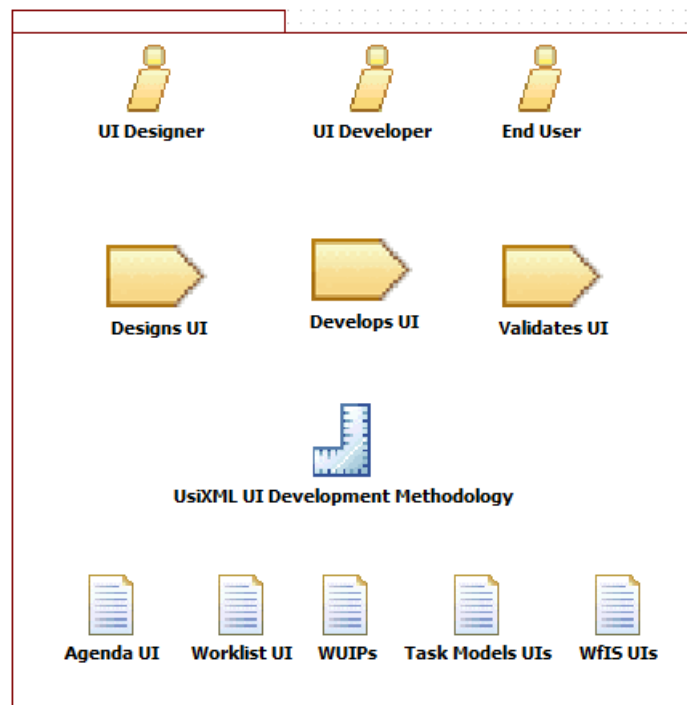


Figure 4-11. Implementation phase.

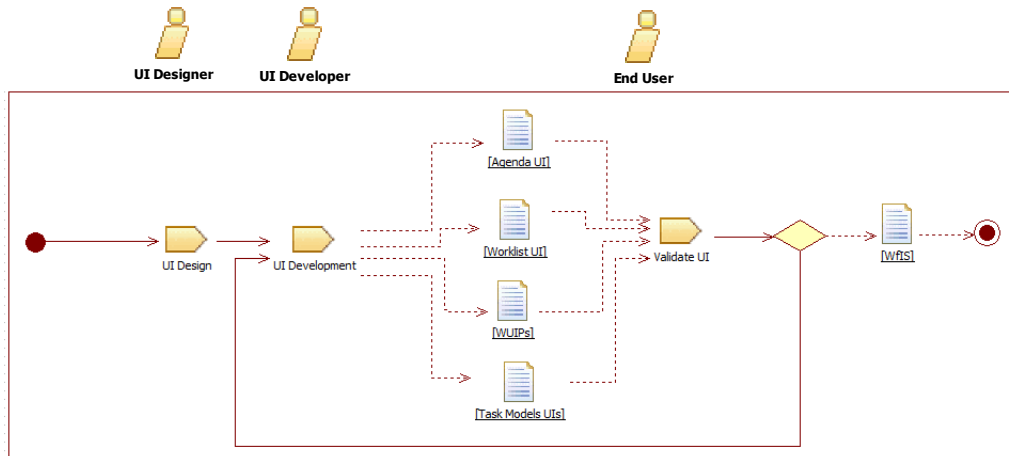


Figure 4-12. Implementation steps.

As all the UIs are expected to be the result of the use of UsiXML methodology we consider its literature as guidance. UIs can be developed from a task model definition. So, we provide task models for the resource allocation patterns in order to design their UIs based on UsiXML methodology. The set of activities (Figure 4-12) starts with the UI designer sketches the UI design that is then developed by a UI developer. Each UI is evaluated by the end-user. When the evaluation satisfies end-user expectations then it becomes part of the WfIS. The same set of activities is carried out for every single UI of the corresponding set. We included UI evaluation as part of the activity set but we do not provided any mean to perform such evaluation. UI evaluation has been largely addressed in the literature thus any method for UI evaluation can be used. Even more, UsiXML literature addresses usability aspects and guidelines evaluation in their methodology.

4.2.3.a User interfaces for task models

To develop UIs from task models, we rely on UsiXML method [Vand05b] that is compliant with the Cameleon Reference Framework [Calv03]. This method is divided in four steps: task model, abstract user interface model, concrete user interface model and final user interface. UsiXML uses a set transformation rules to pass from one development step to another. Transformations are encoded as graph transformations performed on the involved models expressed in their graph equivalent. In addition, a graph grammar gathers relevant graph transformations for accomplishing the sub-steps involved in each step. A complete overview of UsiXML can be found in [Vand05b].

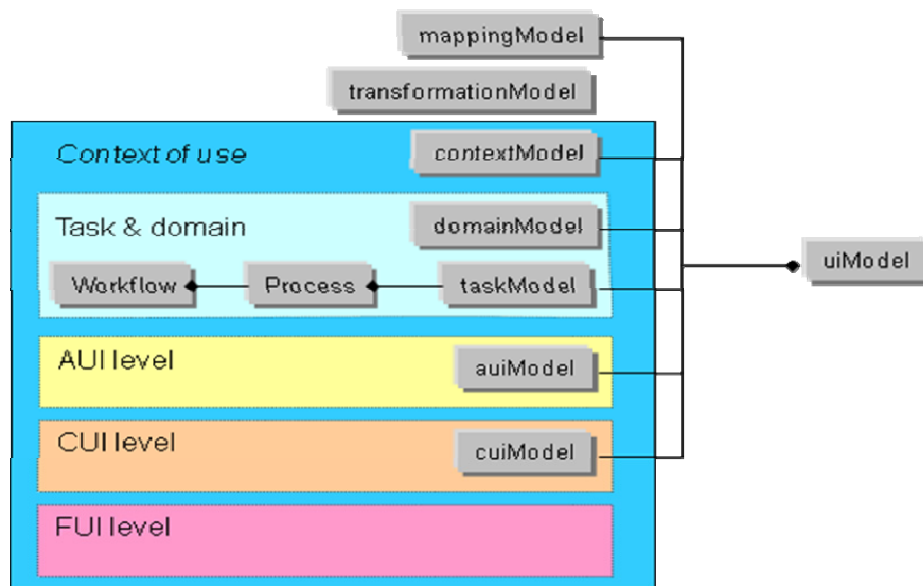


Figure 4-13. Cameleon Reference Framework and UsiXML.

Model-based user interfaces design often starts with a task model that is evolved through an incremental approach to the final UI. In Figure 4-13 the simplified view of UsiXML compliant with the Cameleon reference framework and where our concepts (workflow, process) are included.

UsiXML is a collection of models [USIX07] for specifying a UI, although not all have been explicitly considered in our research. For instance the domain and context models but this does not exclude you to use them, thus we list all UsiXML models:

- *Task Model:* is a model describing the interactive task as viewed by the end user interacting with the system.
- *Domain Model:* is a description of the classes of objects manipulated by a user while interacting with a system. We do not use explicitly this information in our method. Although it can be used.
- *Mapping Model:* is a model containing a series of related mappings between models or elements of models. In this model resource allocation patterns are stored.
- *Transformation Model:* Graph Transformation (GT) techniques were chosen to formalize explicit transformations between any pair of models, except from the FUI level.

Chapter 4. Developing workflow user interfaces

- *Context Model*: is a model describing the three aspects of a context of use in which an end user is carrying out an interactive task with a specific computing platform in a given surrounding environment. Consequently, a context model consists of a user model, a platform model, and an environment model. We do not use explicitly this information in our method. Although it can be used.
- *Abstract User Interface (AUI) Model*: is the model describing the UI at the abstract level as previously defined.
- *Concrete User Interface (CUI) Model*: is the model describing the UI at the concrete level as previously defined.
- *User Interface Model*: is the topmost super class containing common features shared by all component models of a UI.

The different transformational steps for the generation of UI are described in the next sub-sections.

4.2.3.a.1 Step 1: Construct the task model

The initial development step consists of specifying the *Task Model* which requires, first, the identification of the interactive tasks along with their associated attributes and, second, the specification of the relationships between the tasks.

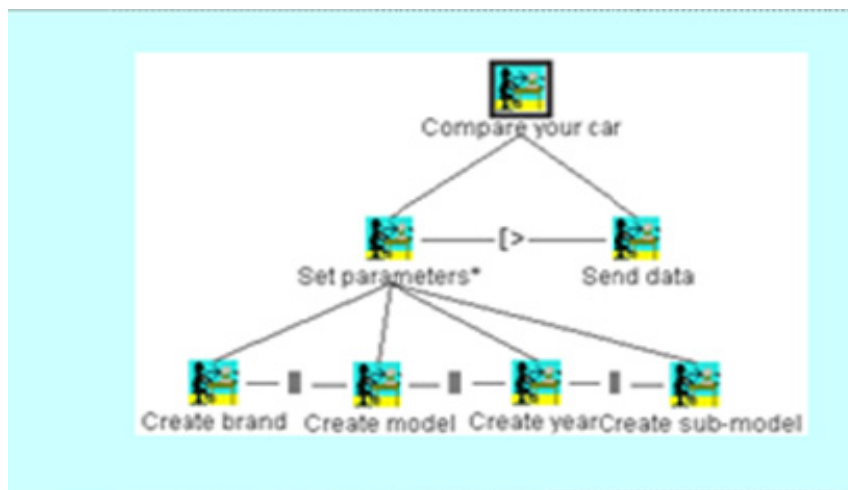


Figure 4-14. Task model.

4.2.3.a.2 Step 2: From task models to abstract user interface model

In next step, an AUI model can be generated automatically, using transformation engine or produced manually from a task model following a set of heuristics. An AUI is defined as the abstraction of any CUI with respect to interaction modality.

Chapter 4. Developing workflow user interfaces

According to Model-Driven Engineering (MDE), it is platform-independent model (PIM). Various set of heuristics may fit this purpose depending on the type of AUI to be obtained: an AUI that reflects the task structure, an AUI minimizing navigation, an AUI compacting input/output. Although this level is independent of any modality, some guidance is still desired on how AUI might be structured considering further reifications into concrete objects.

Abstract Interaction Object (AIO) may be of two types: an Abstract Container (AC) is an entity allowing a logical grouping of other abstract containers or abstract individual components. AC are said to support the execution of a set of logically/semantically connected tasks; and an Abstract Individual Component (AIC) is an abstraction that allows the description of interaction objects in a way that is independent of the modality in which it will be rendered in the physical world. An AIC may be composed of multiple facets. Each facet describes a particular function an AIC may endorse in the physical world. Four main facets are identified:

1. An *input* facet describes the input action supported by an AIC.
2. An *output* facet describes what data may be presented to the user by an AIC.
3. A *navigation* facet describes the possible container transition a particular AIC may enable.
4. A *control* facet describes the links between an AIC and system functions i.e., methods from the domain model when existing.

A single AIC may assume several facets at the same time or just one (first row in the table below).

Input	Output	Navigation	Control
Input + Output	Output + Navigation	Navigation + Control	
Input + Navigation	Output + Control		
Input + Control	Output + Navigation + Control		
Input + Output + Navigation			
Input + Output + Control			
Input + Navigation + Control			
Input + Output + Navigation + Control			

Table 4-5. AIC facets.

Chapter 4. Developing workflow user interfaces

Transformation rules describing the mapping from model to model are numerous in UsiXML and has been reported in different pieces of work [Limb04b, Stan08]. Although, in practice these transformations are hidden for the designers when using UsiXML family tools (more details in section 5.3.1). The transformation rules informally are described for this mapping from task to AUI model. For each task that has task children an AC will be created. For instance the root task *compare your car* has two task children (*set parameters*, *send data*) so an AC (*compare your car*) is created. For each leaf task an AIC will be created. For instance for each leaf task, *create brand*, *create model*, *create year*, *create sub-model*, and *send data* an AIC will be created.

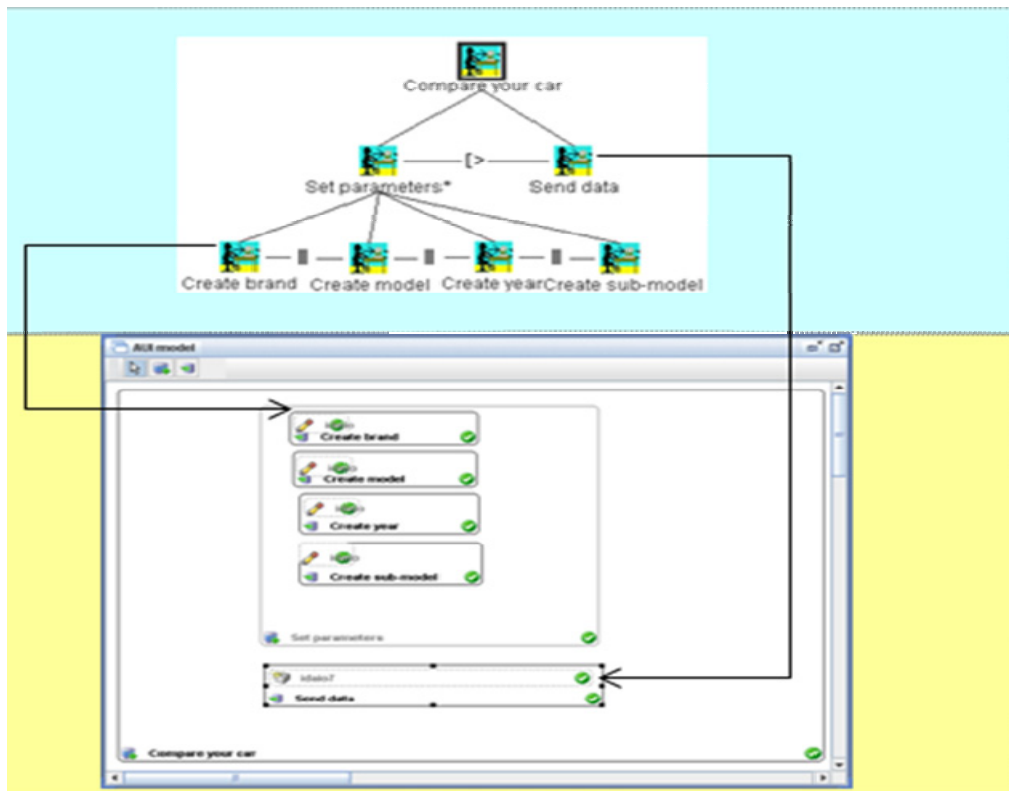


Figure 4-15. From task model to AUI model.

For each parent task that has children tasks, if parent task is associated to an AC (called parent AC) and child task is associated to an AC (called child AC), then, create an association relationship that will ensure the containment of the child AC into the parent AC. For instance the task *set parameters* has four task children (*create brand*, *create model*, *create year*, *create sub-model*) that are respectively associated to AC (*set parameters*). An association relationship is created to ensure the containment of AC *set parameters* into AC *compare your car*. For each parent task that has children

Chapter 4. Developing workflow user interfaces

tasks, if parent task is associated to a parent AC and child task is associated to a child AIC, then, create an association relationship that will ensure the containment of the child AIC into the parent AC. For instance the task *create brand* is associated to AC *set parameters*. An association relationship is created to ensure the containment. The resulting AUI is shown in Figure 4-15.

4.2.3.a.3 Step 3: From abstract user interface model to concrete user interface model

The third transformation step consists of the transition from the Abstract UI Model to the Concrete UI Model (Figure 4-16). A CUI is defined as the abstraction of any FUI with respect to computing platforms, but with the interaction modality given, i.e., GUI, vocal UI, haptic. According to MDE, it is a platform-specific model (PSM). A CUI is made up of Concrete Interaction Objects (CIO), which are abstractions of widgets found in those platforms. Any CIO may be associated with any number of behaviours. A behaviour is the description of an Event-Condition-Action (ECA) mechanism that results in a system state change.

For instance:

ON (Event)	IF (Condition)	THEN (Action)
Click (Button, Mouse1LeftBut)	sourceCurrency<>targetCurrency AND sourceCurrency_Changed	Currency.Convert(sourceCurrencyValue, targetCurrencyValue)
keyTyped (KeyBRD1, null)	controlKeyPressed AND keyBRD1.KeyChar='v'	Paste()
Click (Button, Mouse1LeftBut)	systemDataSave	Exit()

Table 4-6 shows some mappings between AUI events and their concretization in terms of CUI events used in ECA rules, a complete list of examples is presented in Appendix G. ECA rules are used indifferently at both the AUI and the CUI levels. This table shows that for a same abstract event in a dialog model for AUI, several different mappings can be ensured with concrete events in a dialog model for CUI depending on the context of use, particularly the various interaction modalities, in this case: graphical and vocal (see [Stan08] for more details on vocal examples). Although that exist a lot of CUI events, we focus on those that are frequently used.

Chapter 4. Developing workflow user interfaces

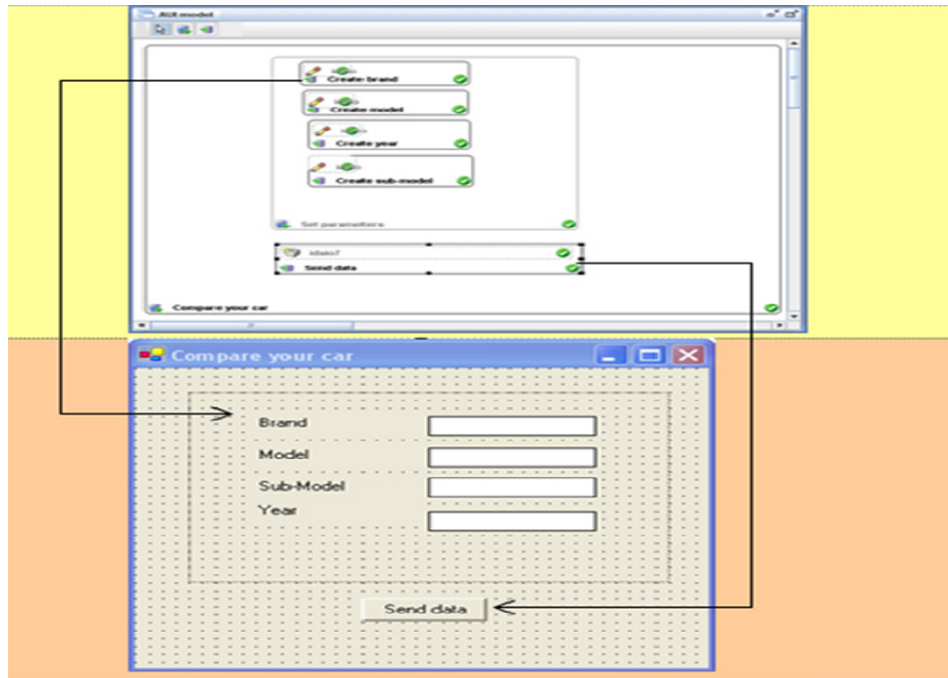


Figure 4-16. From AUI model to CUI model.

Also, notice that certain facets or combinations of them are not applicable as they do not make any sense or are contradicted, they are:

- Activate an AIC with navigation facet. In this case is irrelevant to exemplify.
- Activate an AIC with control facet & activate an AIC with navigation + control facet. Theoretically possible but violate usability guidelines “every control should produce a feedback”.

AUI event	Example of CUI events		
	Graphical		Vocal
	Graphical UI	Web	
Activate an AIC with input facet	To fill out a form, type or modify text start date Due Date subject 	To fill out a web form, type o modify text Username Password 	vocalInput or record
	to select a task ☐ Call to Mrs. Scott ☐ Meeting in Brussels	to select an option Color ☒ Red ☐ Yellow ☐ Green	

Table 4-6. Mappings between concrete and abstract events.

4.2.3.a.4 Step 4: From concrete user interface model to final user interface

From each CUI, a corresponding FUI can be produced by automated model-to-code generation. A FUI is hereby referred to as any UI running on any computing platform with any interaction modality, whether it is rendered by interpretation or by code generation.

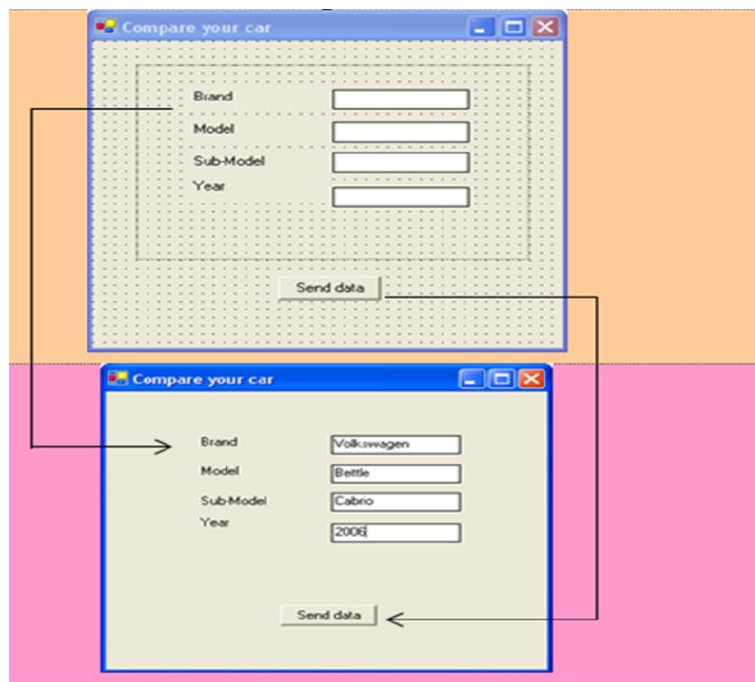


Figure 4-17. From CUI model to FUI.

4.2.3.b Workflow user interface patterns

Task allocation, i.e. the manner in which tasks are advertised and ultimately bound to specific resources for execution needs also care when developing WfIS. To capture the different manners in which resources are presented and used in workflows we rely on *workflow resource patterns* [Russ05].

We adopted the following steps for defining the Workflow User Interface Patterns (WUIPs):

- *UI pattern definition*: from each workflow resource pattern a WUIP is created and defined.
- *Incorporation in the model-driven engineering method*: for each initial pattern definition resulting from the previous step, a task model has been specified using CTT notation [Pate99] in order to depict the pattern.

Chapter 4. Developing workflow user interfaces

- *Final WUIPs*: from the task models resulting from the previous steps, abstract UIs and, consequently, concrete UIs have been defined in terms of the UIDL (here, UsiXML) so as to form corresponding abstract and concrete UI models.

Applying the above methodology resulted in 42 WUIPs (complete list is presented in appendix E). We give below only a snapshot of some of these patterns for facilitating the understanding.

<i>Name:</i>	Direct allocation
<i>Identifier:</i>	R-DA
<i>Synopsis:</i>	The ability to specify at design time the identity of the resource that will execute a task
<i>Strengths:</i>	To prevent the problem of non-suitable allocation
<i>Weakness:</i>	No opportunity to change the resource if he is not available to perform the task
<i>Opportunities:</i>	To ensure task is routed to specific resource
<i>Problem:</i>	This pattern effectively defines a static binding of tasks to a single resource
<i>Solution:</i>	Probably the use of deadline and escalation mechanisms when the resource becomes overload and cannot deal with his assigned workload in a reasonable timeframe
<i>Example:</i>	“Ask reviewers preferences” task must only be undertaken by “Joshua Brown”

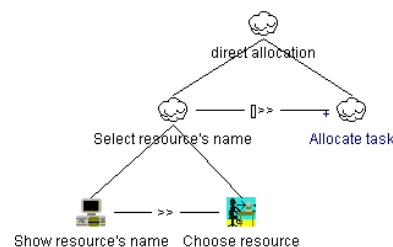


Figure 4-18. Direct allocation pattern.

4.2.3.c User interface flow

After having defining the UIs involved in the workflow, we need now to link all the UIs: the one for the workflow manager and the ones for the workflow tasks. This will be achieved thanks to the *user interface flow*.

During the execution of work, information passes from one resource to another as tasks are finished or delegated; in *FlowiXML* we use an *agenda* assigned to each resource to manage the tasks that are allocated/offered to him. The manager uses the worklist to view and manage tasks that are assigned to each resource.

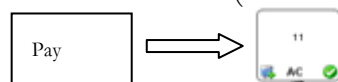
By linking UIs we expect to solve the problem of synchronizing the communication among them. We introduce some rules that can be applied to facilitate the modeling of the UIs flow that is relevant for the implementation of a WfIS.

4.2.3.c.1 Rules

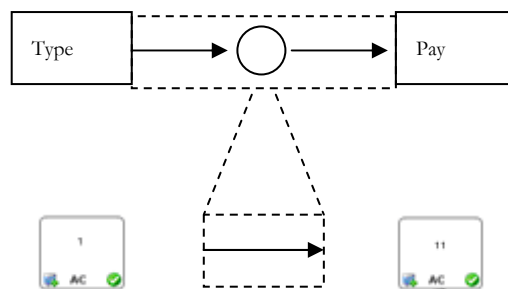
 *Rule 1.* For each userStereotype create a swim-lane

U ₁	
U ₂	
U ₃	
Wf Manager	

 *Rule 2.* For each task (transition) create an Abstract User Interface (AUI)

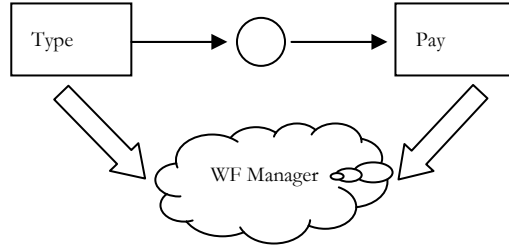


 *Rule 3.* For each workflow transition create an AUI transition



Chapter 4. Developing workflow user interfaces

- *Rule 4.* For each task create a feedback for the workflow manager



- *Rule 5.* For every couple of consecutive tasks which execute in turn one after the other, create a relationship of type “sequence” between these tasks.
- *Rule 6.* If there is a split into two or more branches which can execute tasks parallel, create a relationship of type “parallel split” among these tasks.
- *Rule 7.* If there is a convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled, create a relationship of type “synchronization”.
- *Rule 8.* If there is more than one branch but just one can be enabled, create a relationship of type “exclusive choice”.
- *Rule 9.* If there is a convergence of two or more branches into a single subsequent branch such that each enablement of an incoming branch results in the thread of control being passed to the subsequent branch, create a relationship of type “simple merge”.
- *Rule 10.* When the divergence of a branch in two or more branches such that when the incoming branch is enabled, the thread of control is immediately passed to one or more of the outgoing branches, create a relationship of type “multi-choice” that selects one or more outgoing branches.

4.2.3.c.2 Defining flow of UIs

The flow of User Interfaces is an octuple $UIF (A, \Sigma, U, T, \delta, \omega, ai, ao)$ where

- A is a nonnegative finite set of Abstract Containers (AC)
- Σ is a set of input events [set of events occurring in AC]
- U is a nonnegative set of user stereotypes, such that $\forall a \in A: \exists! u \in U \uparrow$ is used by (a, u) [unique] or $\exists u_1, u_2 \dots u_n \in U \uparrow$ is used by $\{a, u_1, u_2 \dots u_n\}$ [a is shared among $u_1, u_2 \dots u_n$]

Chapter 4. Developing workflow user interfaces

- T is a set of output transitions [output transitions means a navigation from starting AC to a final one, we do not want to commit ourselves to a particular type or representation]
- δ is a transition function, $\delta : A \times \Sigma \rightarrow A$ [a transition is AC + abstract event occurring in one AC]
- ω is an output function, $\omega : A \rightarrow T$
- ai is the initial AC [$ai \in A$]
- ao is the final AC [$ao \in A, ao \neq ai$]

Hypothesis. The target AC after a transition is activated.

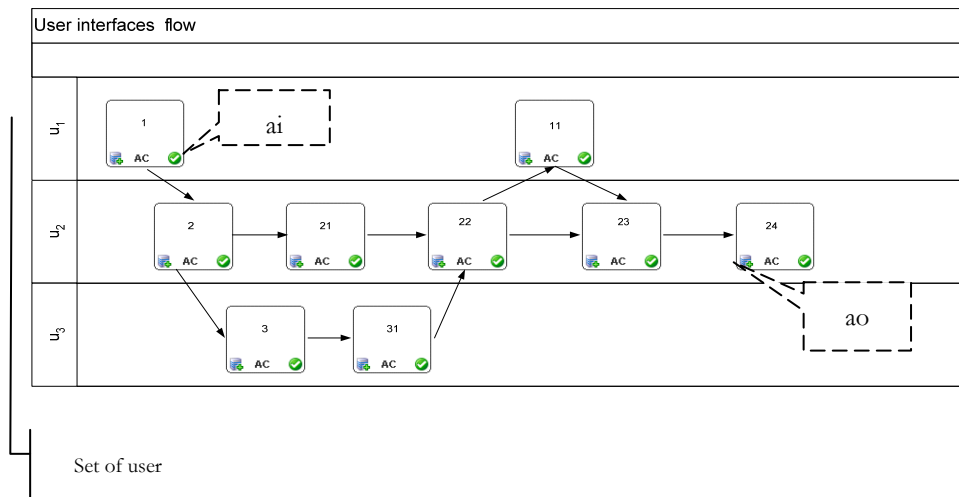


Figure 4-19. User interfaces flow.

4.2.3.d Error recovery

During the execution of the workflow errors occur, at any level. Even that we have not an explicit mechanism to consider error handling we consider that task model and process model can be used for this purpose.

In a task model it is possible to model error capture and detail its recovery steps. There are two options for doing that. First, detailing the task model including tasks capturing errors and recovery mechanism, all these tasks would be optional. They are not part of the natural flow of the task model execution. Second, being not part of the task model in nature, error recovery tasks can be modeled separately and then be related to the task, which could be object of interest in case of error, using the mapping *is grafted on*.

Another way to provide users means to recover from error is the definition of patterns on tasks in a process model. Particularly, the *redo workflow resource pattern*

specifies that a task can be redone it for any reason. This second scenario expresses that the whole task must be repeated, i.e., the complete task model.

4.3 Chapter summary

In this chapter, we introduced a method that integrates: tasks, process, and workflow model. The original UsiXML task model was expanded with additional attributes and relationships pertaining to the domain of workflow. The domain model remains unchanged.

We identify the problem of identifying the boundaries between task, process and workflow models. Due to the correlation of the concepts that they involved there was a need to clearly specify, to our understanding, where each model starts and finishes. We propose a set of criteria for this purpose that includes cross-model verification: once one aspect is modeled in some part, one should ensure that the rest is properly connected and that the three aspects are consistently modeled. This is somewhat related to the question of what is in the model, what is not. In addition, we present an overview of the method to develop UIs based on UsiXML and Cameleon Reference Framework. In order to coordinate the UIs, we included some rules and a definition to user interface flow.

Error recovery is one important aspect when modeling interactive systems. Although we have not considered fine-grained error recovery mechanisms we provided some design options for modeling errors.

There are usability guidelines [Pala99] that we would like to incorporate. In principle, we will use them for evaluation of UIs in order to respect cognitive and sensory-motor capabilities of users. However, these ergonomic rules will be considered for a future work.

In the next chapter we present the software to support this method.

Chapter 5 Software for supporting the method

5.1 Context

In order to support the method for developing WfIS, two software modules are needed: one for the design of the WfIS and one to support the execution of the WfIS. They are respectively called *workflow tools* and *workflow manager*. They incorporate a design artifact to our schema (Figure 5-1).

The *workflow tools* allows modeling the general WfIS defining workflow, process and task models, organizational units, jobs and resources involved, and allocation of task to resources. The *workflow manager* is the run-time engine, this is just partially developed.

The software was developed as a master thesis [Lema07] based on the conceptual modeling presented in [Guer06]. It is the result of constant improvements based on informal evaluations, interviews with the users after modeling a WfIS. Although supports our methodology the software usability independent of the method presented in chapter 3. This aspect is discussed more in details when the methodology is evaluated in next chapter.

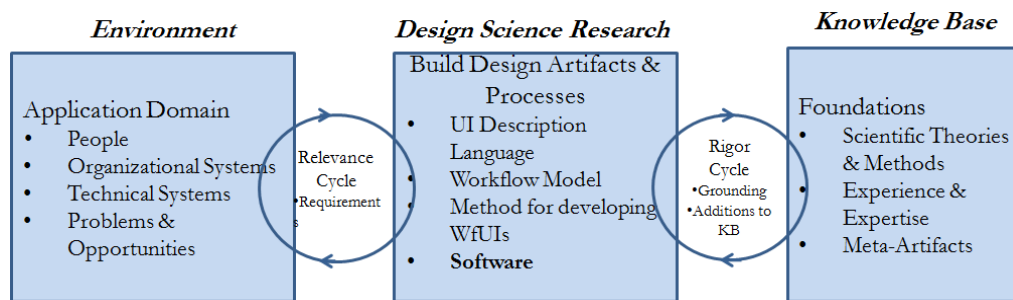


Figure 5-1. Schema for software derivation (adapted from [Hevn07]).

Most of the tools for the design are new (depicted as CD with black marks in Figure 5-2). The set of tools that we reused are depicted a white CD in Figure 5-2 (basically for task modeling IdealXML and UsiXML software family of tools).

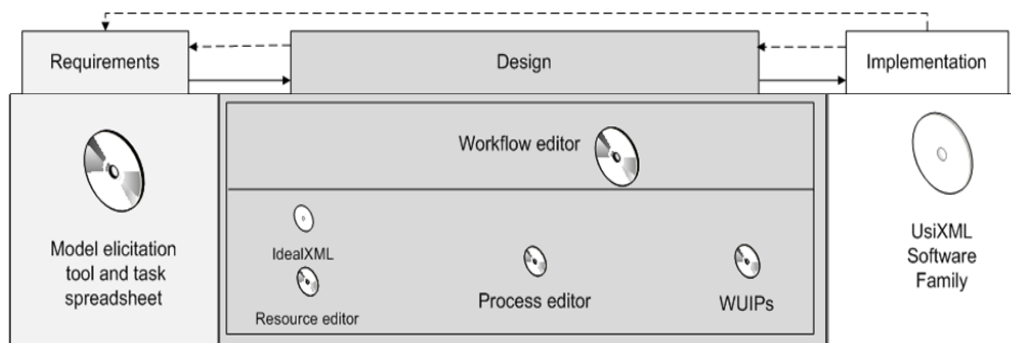


Figure 5-2. Software tools.

5.2 Workflow tools

Primarily we will discuss the elements to which the user has access during the design of the WfIS. In Figure 5-3 the main editors available: for requirements elicitation the model elicitation tool and the task spreadsheet. The design of the system is done with the jobs and user stereotypes editors, and the workflow editor. The workflow editor is used for modeling workflow, task, process, and resource allocation patterns.

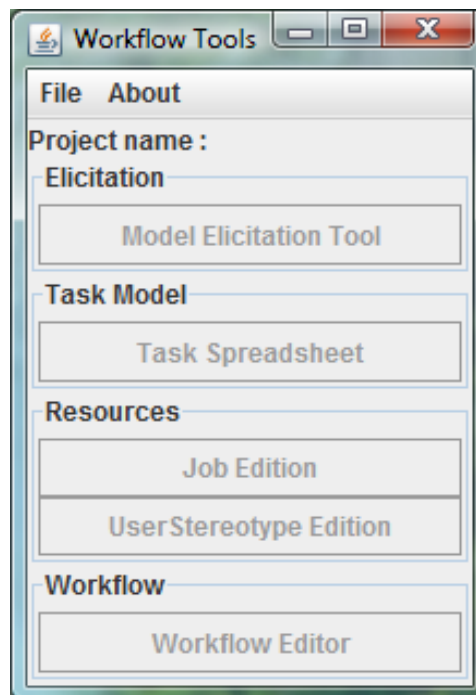


Figure 5-3. Workflow editor.

5.2.1 Elicitation from textual scenario

The main goal of elicitation is to handle the textual statement from the beginning to the end and to ensure that all textual fragments that should be transformed into model elements are indeed elicited. In particular, the graphical highlighting in colors allows designers to quickly identify to which model type the element is relevant and to check in the end that the complete scenario has been exhausted, that no term remains unconsidered. For instance, a scenario sentence like “An accountant receives taxes complaints, but she is also in charge of receipts perception” should generate: a task “Receive taxes complaint”, a task “charge of receipts perception”, both being assigned to the user stereotype “Accountant”, and a concurrency temporal operator between those two tasks because no specific term is included to designate how these tasks are actually carried out by an accountant. We may then assume the most general temporal operator, like a concurrency temporal operator. In order to reach this goal, this level attempts to identify possible terms in a syntactical structure (e.g., a set, a list, a sequence) that depicts a pattern for inferring for instance a task, another task with a temporal constraint, etc.

For more information about this elicitation see [Lema08].

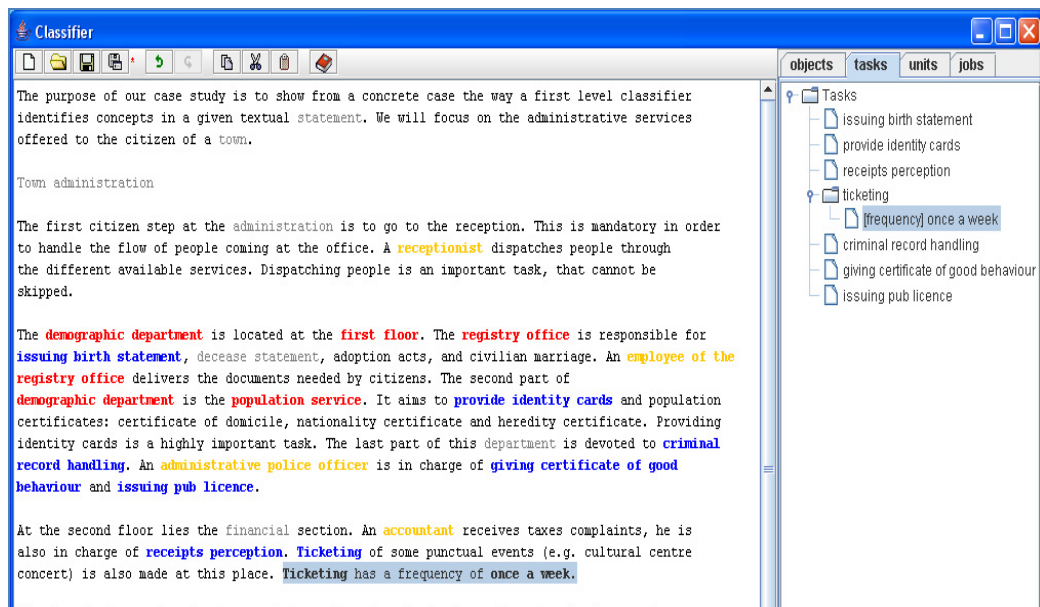
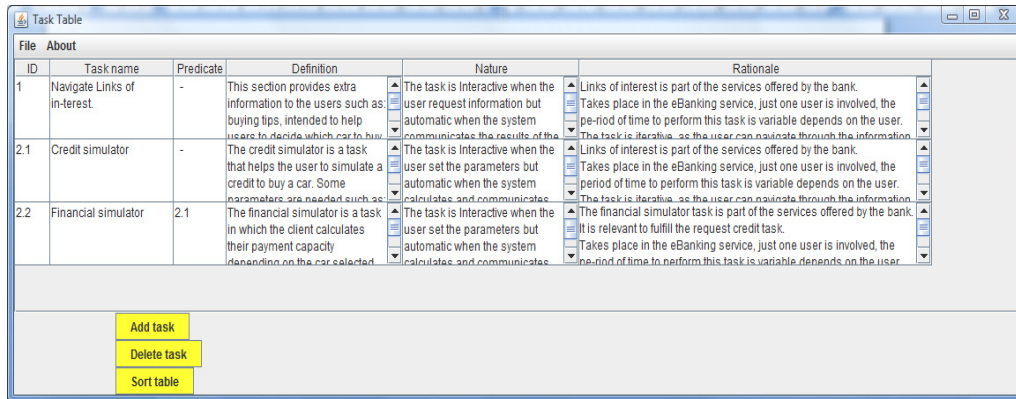


Figure 5-4. Elicitation from textual scenario.

Chapter 5. Software for supporting the method

5.2.2 Task identification

After the elicitation, a task spreadsheet is semi-automatically generated. This table contains the task ID, name, a brief description, the nature of the task, and the justification based on identification criteria.



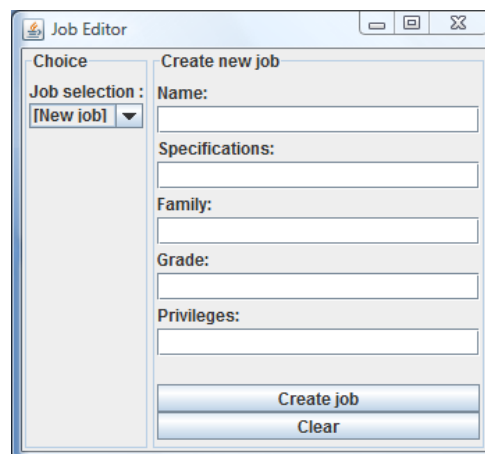
ID	Task name	Predicate	Definition	Nature	Rationale
1	Navigate Links of in-terest	-	This section provides extra information to the users such as buying tips, intended to help users to decide which car to buy.	The task is Interactive when the user request information but automatic when the system communicates the results of the user set the parameters but automatic when the system calculates and communicates	Links of interest is part of the services offered by the bank. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user can navigate through the information.
2.1	Credit simulator	-	The credit simulator is a task that helps the user to simulate a credit to buy a car. Some parameters are needed, such as:	The task is Interactive when the user set the parameters but automatic when the system calculates and communicates	Links of interest is part of the services offered by the bank. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user can navigate through the information.
2.2	Financial simulator	2.1	The financial simulator is a task in which the client calculates their payment capacity depending on the car selected	The task is Interactive when the user set the parameters but automatic when the system calculates and communicates	The financial simulator task is part of the services offered by the bank. It is relevant to fulfill the request credit task. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user.

Buttons: Add task, Delete task, Sort table

Figure 5-5. Task table.

5.2.3 Resource specification

This part is dedicated to the description of the resources involved in the performance of tasks. Jobs are ways to structure the crew of people inside the organization. It involves the complete collection of knowledge and practices needed by a definite human resource to perform a task. The first selection of worker capability is about job qualification. We define a job by its attributes: name, specification, family, grade and privileges.



Job Editor

Choice

Job selection : [New job]

Create new job

Name: _____

Specifications: _____

Family: _____

Grade: _____

Privileges: _____

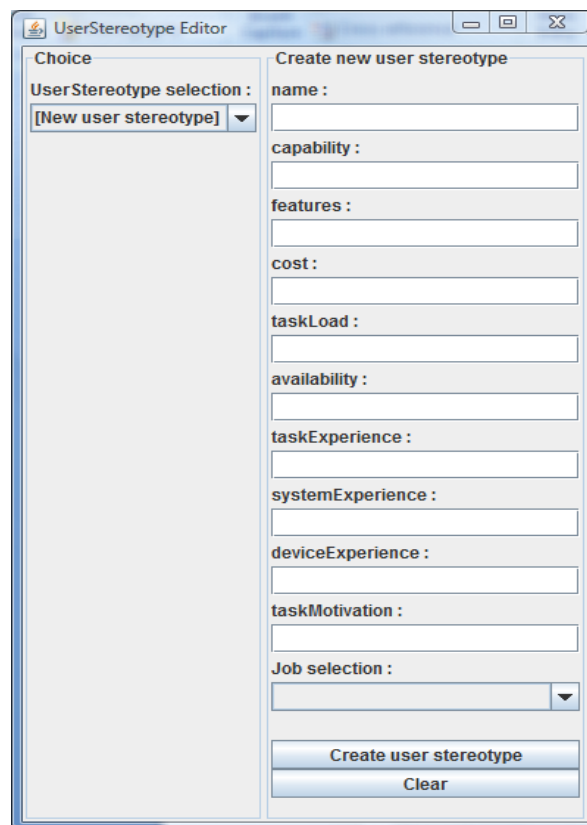
Create job

Clear

Figure 5-6. Job editor.

Chapter 5. Software for supporting the method

The userStereotype refers to the real people in charge of work requiring human resources. A user is able to carry out one or more jobs. Jobs and users can be specified inside the editor by the mean of the resource menu. The job handler and workers editors allow the workflow designer to edit the related attributes. Each modification of a textual field must end by a confirmation by pushing on the enter touch. New jobs and workers can be created, existing ones can be edited.



The screenshot shows a window titled "UserStereotype Editor". It is divided into two main sections. The left section, labeled "Choice", contains a "UserStereotype selection :" label and a dropdown menu currently showing "[New user stereotype]". The right section, labeled "Create new user stereotype", contains a series of input fields for the following attributes: "name :", "capability :", "features :", "cost :", "taskLoad :", "availability :", "taskExperience :", "systemExperience :", "deviceExperience :", and "taskMotivation :". Below these fields is a "Job selection :" dropdown menu. At the bottom of the right section are two buttons: "Create user stereotype" and "Clear".

Figure 5-7. userStereotype editor.

5.2.4 Workflow modeling

The representation is based on Petri Nets, and has been enriched with other elements. The program has an arrangement of elements similar to any graphics editor.

Chapter 5. Software for supporting the method

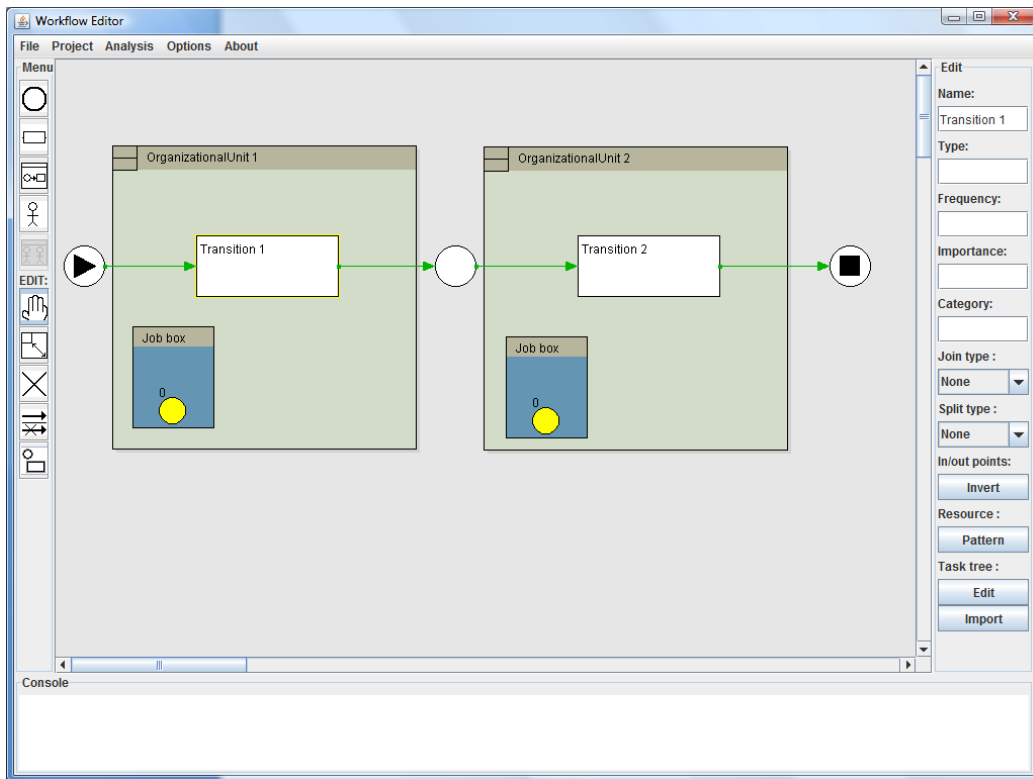


Figure 5-8. Workflow editor.

- On the left a toolbar to insert the graphic elements and to select, move, resize, delete and connect them with an arrow.
- At the top is the menu for access options.
- The right menu allows editing the selected item.
- At the center, the workflow graph which elements are accessible depending on the options identified by the toolbar.
- Finally, a console for communicating messages to the user.

As pointed before, the stylistics is depicted with graphical objects (Figure 5-9).

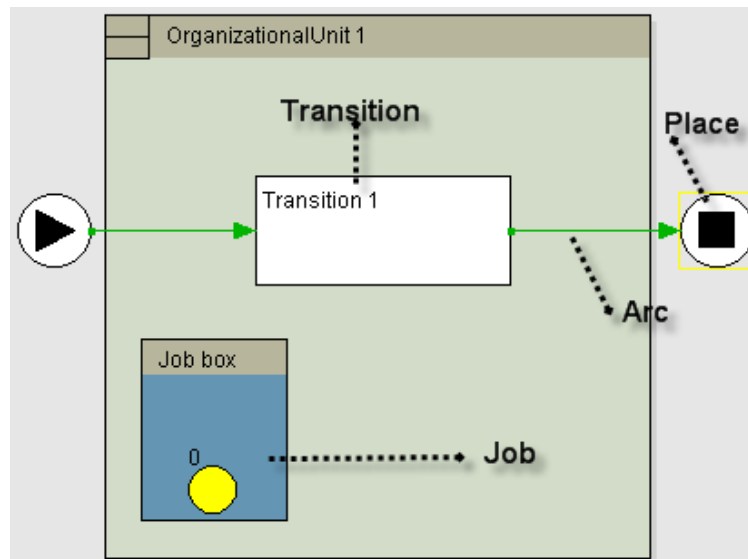


Figure 5-9. Workflow components.

- ▀ *Places*: There are three types of places: the beginning (start), ending (final) and normal. The place of beginning determines the entry point of the workflow. The place of ending identifies the end of the process.
- ▀ *Transitions*: These atomic processes, having only task, feature a pattern routing entry and exit in another. It is through their edition that is accessed both tree decomposition of the task and that we can assign the patterns of resource associated with the task.
- ▀ *Arcs*: Through the use of arrows is the link between places and transitions.
- ▀ *Job*: This box contains a number of workers performing the same job. This allows locate resources.
- ▀ *Organizational unit*: These units can accommodate a part of the workflow to determine specifically where it will take place. An organizational unit has resources, indicated by boxes resources. Specifically, an employee can be attributed to a task it must be located either in the same organizational unit or in a parent unit.

5.2.5 Managing resources

This part is in charge of the workflow resource patterns application. This feature allows determining the right person for the right task at the right moment. For each process it is possible to define one or several allocation or offering relationships.

Chapter 5. Software for supporting the method

The way in which tasks are advertised to resources is essential; therefore we need to consider the different states that task goes through, from creation to termination. There are different approaches to task life cycle [Peti94] [Russ05], from those that contain the basic states to those that contain all the imaginable states.

The window for allocating work for resources that considers all this design characteristics is presented in Figure 5-10.

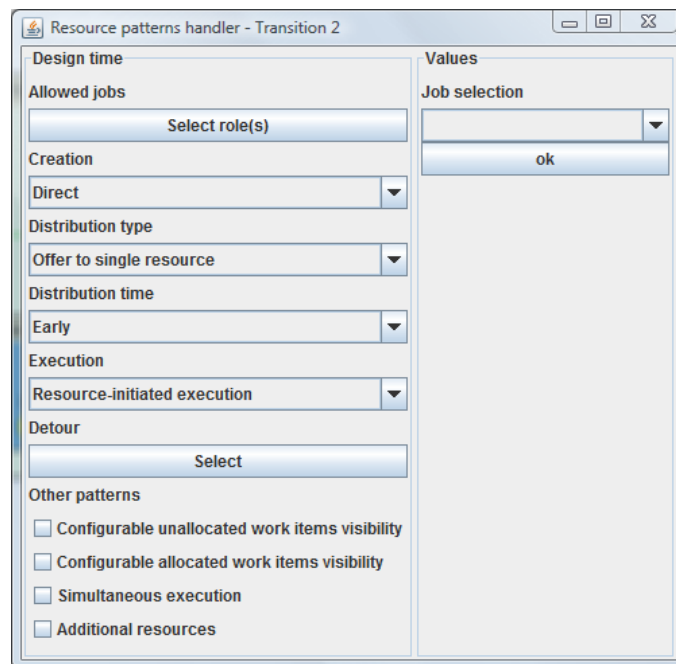


Figure 5-10. Managing workflow resource patterns.

5.2.6 Task model editor

For each transition a task model is specified to describe in detail how the task is performed. This part of the software is completed with a plug-in of IdealXML [Mont05, Mont6]. IdealXML (Interface Development Environment for Applications specified in UsiXML) is a tool to design user interface based on the task. It can easily make prototypes from the specification using a graphical tool. It also generates the AUI model. This software tool is the one that serves as basis for the UI generation as it is UsiXML compliant.

Chapter 5. Software for supporting the method

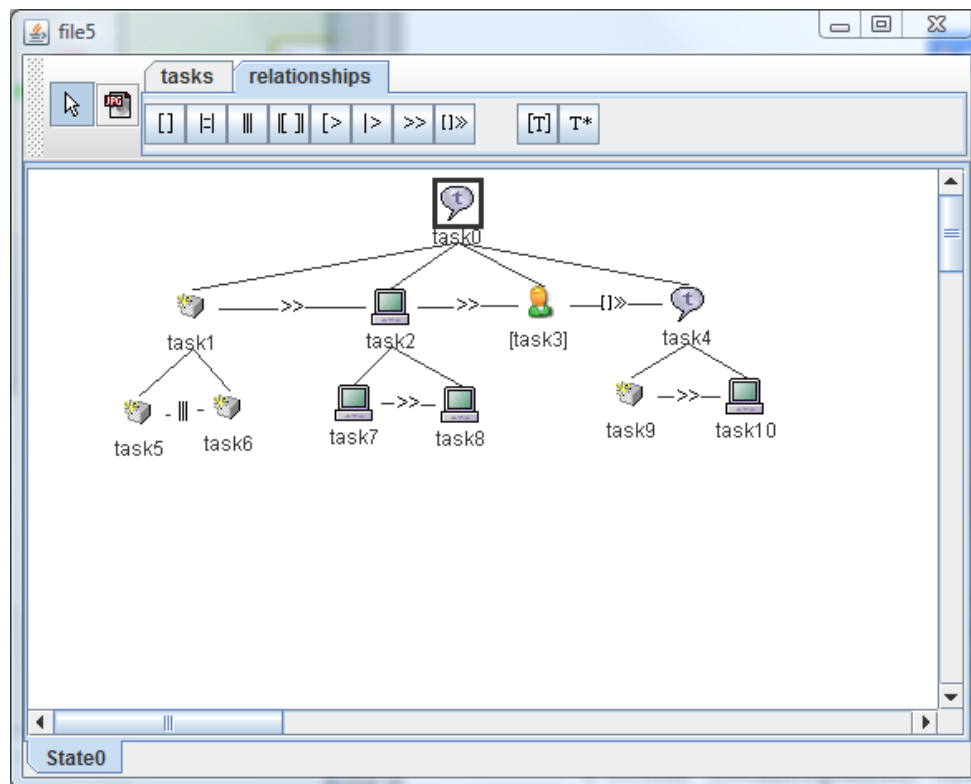


Figure 5-11. Task editor.

The figure shows a 'task properties' dialog box with a table of attributes for 'task10'. The table has two columns: 'name' and 'value'.

name	value
id	st0task10
name	task10
type	application
frequency	
importance	
structuration level	
complexity level	
user action	
task item	
criticity	
centrality	
optional	
iterative	
link OK	
link KO	

Figure 5-12. Task properties.

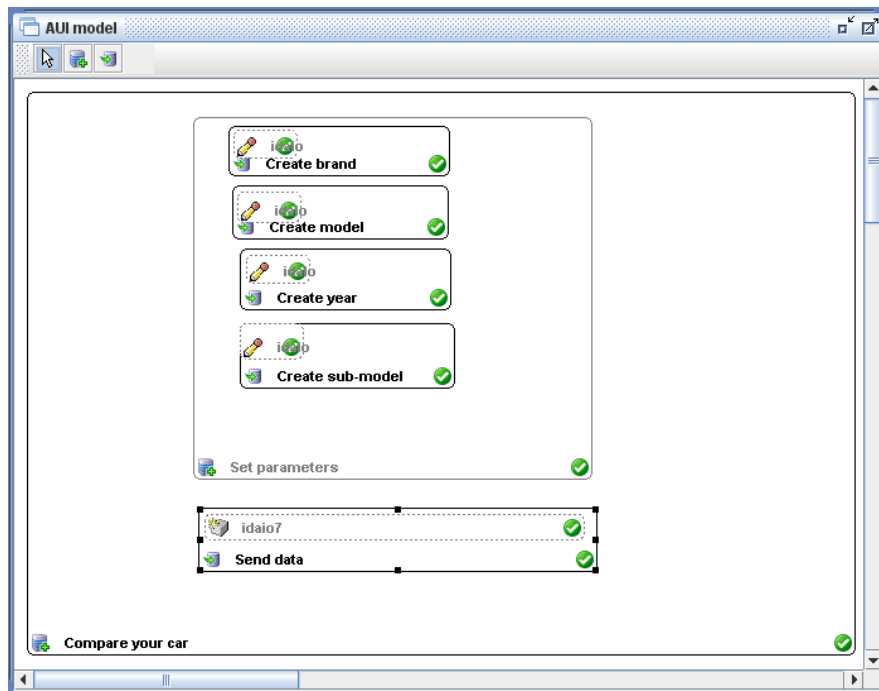


Figure 5-13. AUI model.

5.2.7 Managing files

The editor allows to:

- Define a new workflow
- Open/Save an existing workflow
- Export the workflow specification to UsiXML
- Take screenshot (.jpg)
- View project information, milestones, and organizational tree
- Check star/end, reachability, join/split features
- See an overview map
- Highlight components
- Show/hide highlights components
- Exit from the application

5.3 Workflow manager

The workflow manager allows the simulation of the operation of a service of management of workflow. This second part of the program makes it possible to trace the work throughout their advance in the workflow.

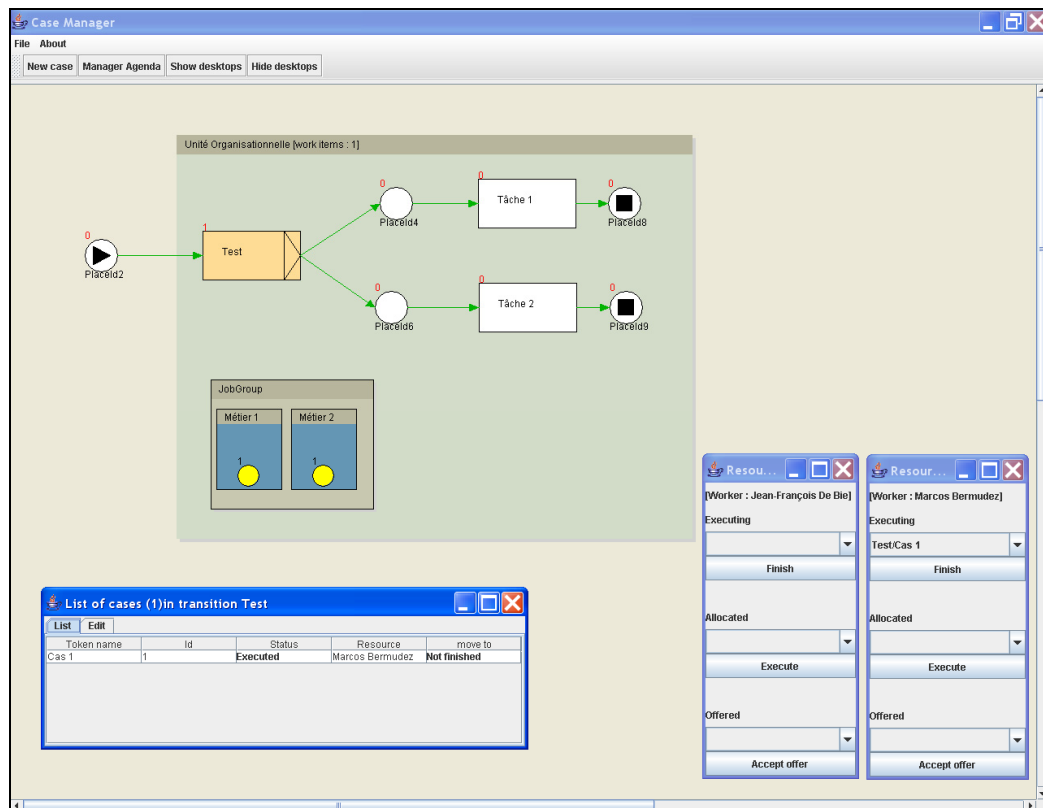
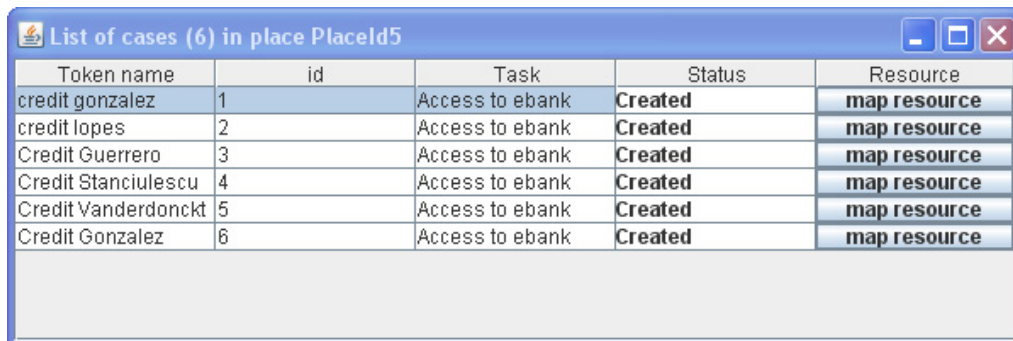


Figure 5-14. Managing workflow.

5.3.1 List of cases

The user starts by charging the specification of a workflow realized previously thanks to the editor. This stage creates one or more new case via the menu (new case). These cases are found at the beginning in the place of entry. Their advance will depend on the cartography established with the editor. When a case is in a place, it means that the next task could be associated a resource. In this moment the access to the option of assignment of resources is available.

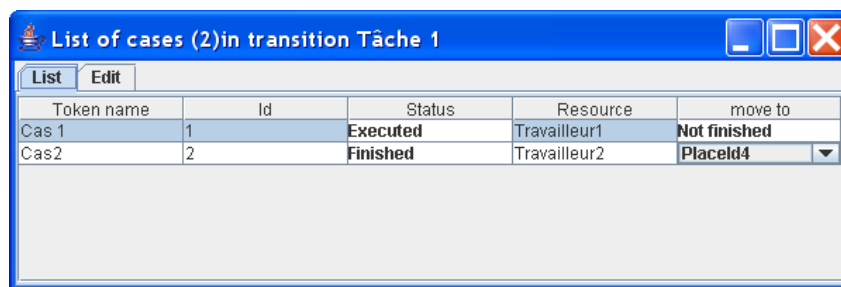
Chapter 5. Software for supporting the method



Token name	id	Task	Status	Resource
credit gonzalez	1	Access to ebank	Created	map resource
credit lopes	2	Access to ebank	Created	map resource
Credit Guerrero	3	Access to ebank	Created	map resource
Credit Stanciulescu	4	Access to ebank	Created	map resource
Credit Vanderdonckt	5	Access to ebank	Created	map resource
Credit Gonzalez	6	Access to ebank	Created	map resource

Figure 5-15. List of cases.

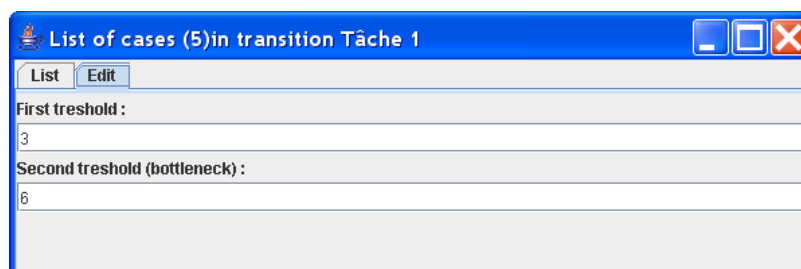
Once that a resource was in charge of the realization of the task and that the execution of the latter took place, the token is moved in the corresponding transition. While clicking on this transition we obtain the following window.



Token name	Id	Status	Resource	move to
Cas 1	1	Executed	Travailleur1	Not finished
Cas2	2	Finished	Travailleur2	Placeld4

Figure 5-16. Update of a list of cases pertaining to a transition.

To help the persons in charge while allowing them to make decisions on the basis of drawing of the workflow, one associates any transition to two skeletal values. These two thresholds are numbers of case being in the transition. If this number reaches the first threshold, the transition will be colored out of orange and red in the case of the second. It thus acts of a visualization of the bottlenecks. The parameterization of these two thresholds is done using the following window.



Token name	Id	Status	Resource	move to
Cas 1	1	Executed	Travailleur1	Not finished
Cas2	2	Finished	Travailleur2	Placeld4

First treshold :
3

Second treshold (bottleneck) :
6

Figure 5-17. Specification of threshold.

Chapter 5. Software for supporting the method

The following figure represents a workflow having the three possible levels. A bottleneck is in red for transition 5.

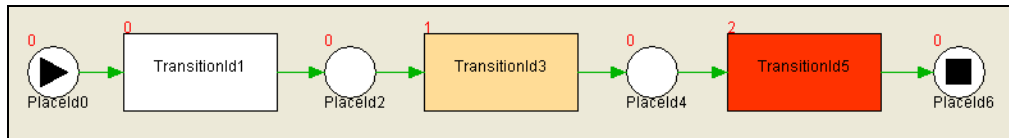


Figure 5-18. Three possible states of a transition.

5.3.2 Agenda and workflow list

The manager's agenda is a summary of the whole of the tokens, indicating in which organizational unit they are and in which place or transition. That offers a general sight, in the shape of the window below.

Token name	id	Organizational unit	Place/Transiton
Cas1	1	Chirurgie	Placeld5
Cas2	2	Psychologie	Placeld6
Cas3	3	General	TransitionId0
Cas4	4	Dermatologie	Placeld7
Cas5	5	General	Placeld4

Figure 5-19. Manager Agenda window.

In the resource mapping the assignment of work to the resources is made. Each time that a task must be carried out for a given case we will have recourse to this resource administrator. This stage the token is still in the place connected to the task to come. It will be only later, when the task is in the course of execution and consequently becomes an activity, that the token will be moved in the transition containing the task. When the user clicks on the button "resource mapper" via the agenda corresponding to the transition, then s/he obtains the window of following assignment of the resources.

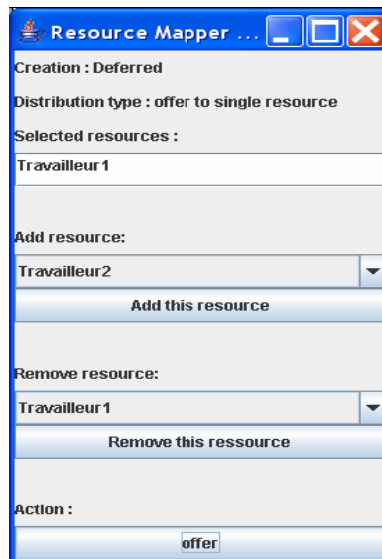


Figure 5-20. Assignment of resources.

The resource desktop window allows the simulation of the communication of assignment of work the resources. In real situation, these windows would be distributed among the computers of the workers and information would be propagated by a network. In our case the goal was to clarify the way in which a worker reacts to the offer or the allocation of work, and whose state of an object of work (the association of a token and a particular task) evolves/moves in the course of time. We can thus see all the advance of work since its creation while passing by the distribution of work to the resources, its execution and its termination. The window making possible to the worker to summarize her/his report with the various cases and the associated tasks is as follows:

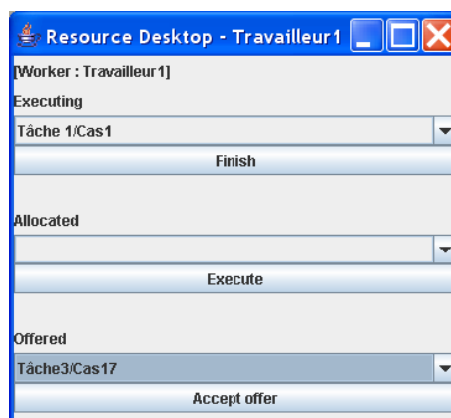


Figure 5-21. Resource desktop window.

5.3.3 User interface flow

With this work we expect to solve the problem of synchronizing the communication between UIs (agendas and task UIs) and the workflow view. The solution provided should be the classical client-server architecture, see Figure 5-22.

The solution should provide communication channels from the workflow manager application (server) to every userStereotype agenda (clients). When a task is finished, for instance, a book is purchased in the Web Store, then the agendas in the accounting and shipping departments must be updated with the new task item. In this transition, the workflow manager application must be updated accordingly to the new state of the item.

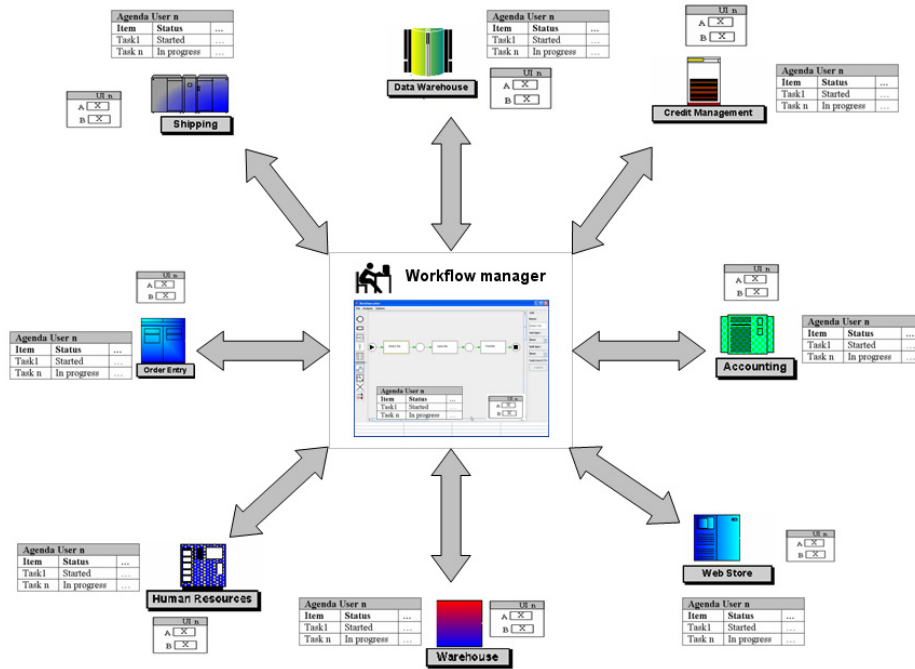


Figure 5-22. General view of the synchronization model to support the communication between UIs.

As, all the information about allocation of task is stored in the workflow manager application, it looks normal to use it also as an intermediate to send messages once tasks are finished to the agendas. For the users the serve will be transparent. They will just fell an application that performs as shown in the Figure 5-23. A user buys a book in the web store, this send a message to the agendas of the shipping and accounting departments; once processed the request, the user will received conformation letters mails to its agenda or he can check the status of his request any time. This aspect is not supported in the workflow manager just simulated.

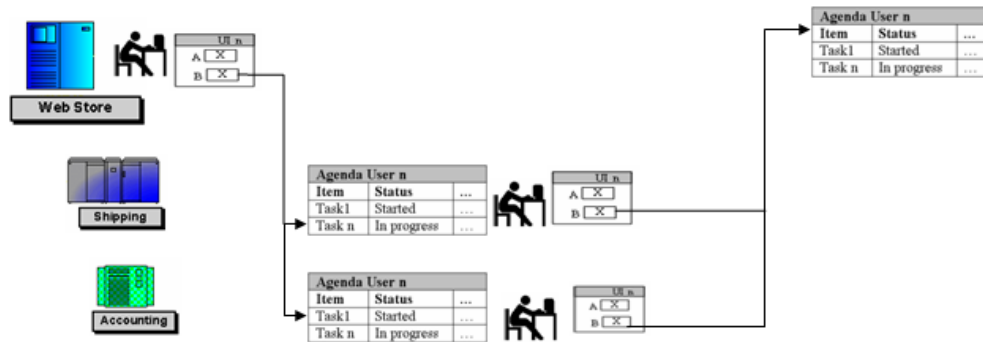


Figure 5-23. Communication between UIs.

5.4 UsiXML software family

A family of software tools (Figure 5-24) supports the development of UIs from task models based on UsiXML. An overview of this software family was introduced in [Vand05b]. The most significant tools belonging to this suite are:

- TransformiXML [Limb04c] is a Java application for defining, storing, manipulating, and executing model-to-model transformations expressed as graph transformations contained in graph grammars.
- IdealXML [Mont06] is a Java graphical editor for the task model, the domain model, and the abstract model. It can also establish any mapping between these models either manually (by direct manipulation) or semi-automatically (by calling TransformiXML). This plug-in is included in our design environment.
- GrafiXML [Mich08] is a UsiXML high-fidelity editor with editing of the CUI, the context model and the relationships between. It is able to automatically generate UI code in HTML, XHTML, XUL and Java thanks to a series of plug-ins.
- SketchiXML [Coye05] consists of a Java low-fidelity tool for sketching a UI for multiple users, multiple platforms (e.g., a Web browser, a PDA), and multiple contexts of use.
- Several renderers [USIX07] are currently being implemented: FlashiXML opens a CUI UsiXML file and renders it in Flash, QtXML in the Tcl/Tk environment, and InterpiXML for Java.

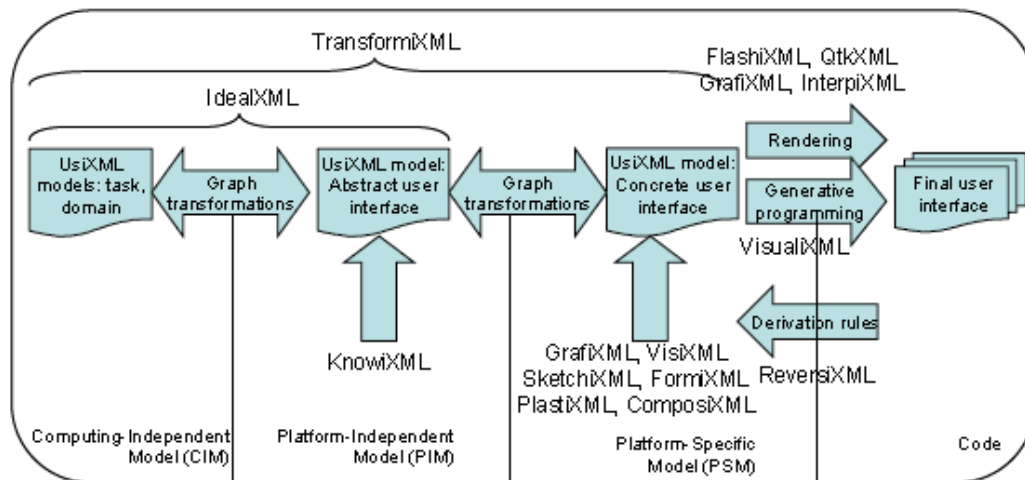


Figure 5-24. The suite of UsiXML tools structured according to the MDA classification, [Vand05b].

5.5 Chapter summary

This chapter presented the software solution supporting the method proposed by the current thesis. This software covers the elicitation from textual scenarios, WfIS specification, UI development based on UsiXML, workflow manager.

Beyond the facilities for automated classification the elicitation software allows editing facilities within a same model and across models. Its main advantage relies in its capability of supporting designers in identifying text fragments that should be considered for model elicitation and in helping them to informally check some desirable model properties.

The Workflow tools support the graphical editing of the concepts in an integrated way. It then enables designers to pick any of the predefined workflow patterns that are later attached to a corresponding UI pattern in UsiXML.

The workflow manager is sketched and some of the design knowledge to build such a system is depicted.

Finally, we described the set of tools used for the UI development based on UsiXML approach.

Chapter 5. Software for supporting the method

Chapter 6 Validation

6.1 Context

The third design cycle proposed by Hevner [Hevn07] is the *design cycle*, it iterates between the constructions of a design artifact, its evaluation, and subsequent feedback to refine the design further (Figure 6-1).

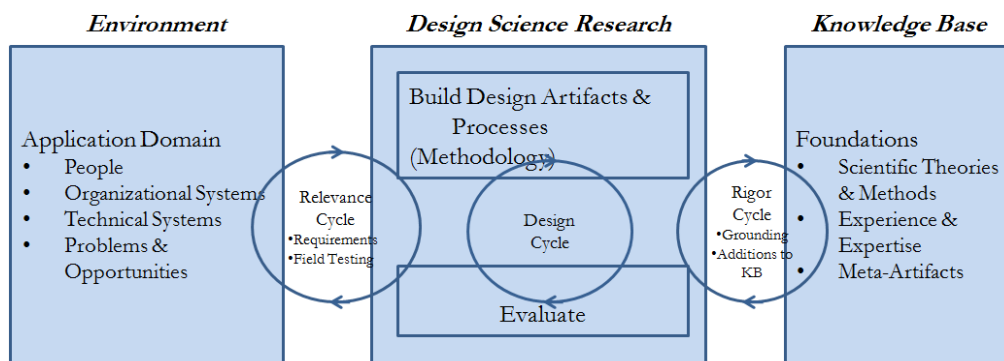


Figure 6-1. Requirement Validation Schema for design science research cycles (adapted from [Hevn07]).

Through evaluation component is possible to demonstrate the utility, quality, and efficacy of a design artifact. Normally the evaluation is based on methodologies available in the knowledge base. This chapter presents the evaluation through the use of some empirical validation methods. For this purpose we rely on:

- *External validation*: the goal is to serve as a proof of the concept of the different principles introduced in the method, and to prove the feasibility of method through a set of real life case studies having different levels of complexity and coverage. This validation is assumed to be globally accepted as it is result of research that has been published in per-reviewed conferences.
- *Internal validation*: aims at assessing the method against the requirements. This is a self validation of the results achieved compared to the expectations we had at the beginning of this work. Also, we discuss the evaluation of the graphical notation using the cognitive dimension method.

6.2 External validation

Zelkowitz proposes a taxonomy used to classify the 11 empirical validation methods [Zelk08]: project monitoring, case study, field study, literature search, legacy data, lessons learned, static analysis, replicated experiment, synthetic, dynamic analysis, and simulation. In our case, we cover the next validation methods:

- *Literature search.* We evaluated published studies that analyze the behavior of similar methods for represent concepts related to workflow, task models, user interfaces description languages, and tools.
- *Lessons learned.* After reviewing the literature, we looked for a meta-model to represent workflow aspects that have some impact on the user interface design. For this purpose, the workflow is decomposed into processes which are in turn decomposed into task. Each task gives rise to a task model whose structure, ordering, and connection with the *domain model* allows the generation of corresponding UIs.
- *Field study.* A project (in the context of SINF2382 course at UCL) was generated where the students went to the different organizations involved in the planning a sport event, made observations and asked questions among the general public involved in the running of the event.
- *Synthetic.* Evaluate the integrity of the tool software with respect features such as level of method support, easiness to use the tool, user-friendly related aspects, etc.
- *Case study.* A collection of different detailed projects was generated to determine if the method and the software are easier to reproduce and cover requirements of section 2.7.2.

The decision for selecting them is merely opportunistic at during the construction of this work we were confronted to such validation.

6.2.1 Case studies

This section present the result obtained after the application of the method and software in a variety domains by students of management and information systems at UCL, and by us in different published papers.

As the method does not force any sequence of steps the two developed cases studies follows different paths. This shows the flexibility of the proposed method.

To gather the case studies we asked the students to propose a real life case study with at least twenty complex tasks developed in four different organizational units, involving sufficiently diversity of resources. The students, under our supervision, developed the case studies by completing this report:

Chapter 6. Validation

5. Job and user modeling

Specify jobs and user stereotypes, that is

- For each job,
 - o Provide a definition of the job
 - o Locate the job in the organizational units where it is applicable.
 - o List the tasks included in the job responsibilities
 - o If needed, list some individuals playing this job.
- For each user,
 - o Provide an individual user model
 - o Provide the hierarchy of all user models after having identified the relationships between them.

6. Process modeling

Organize tasks into processes according to definitions of jobs and user stereotypes by relying on identification criteria for processes. For this purpose,

- Provide the process models resulting from the organization of tasks according to jobs and users
- Report on the identification criteria used to identify them
- Check the completeness and consistency rules for process modeling.

7. Workflow modeling

Organize previously identified processes into a workflow model and model the whole thing thanks to high-level Petri nets. For this purpose, reuse workflow patterns as much as possible and rely on identification criteria for workflows. In this section,

- Provide the complete workflow model resulting from the organization of processes according to identification criteria, assembling rules, and workflow patterns.
- Clearly identify where each workflow pattern has been used and justify why this was appropriate.
- Check the completeness and consistency rules for workflow modeling.

8. UI definition

In this section, report on the user interfaces corresponding to all parts and workers involved in the workflow.

- From the workflow model, derive the specifications of the UI. For each pattern, choose a UI corresponding to the pattern. Give parameters used for this derivation.
- For the others, derive the UI from the corresponding task model.
- Assemble the individual UIs derived into a UI flow.
- Reproduce the UIs in the complete drawing.

9. Workflow simulation and implementation

In this section, report on the results obtained after using the workflow models and its corresponding user interfaces as they have been simulated, implemented in the software used (e.g., Lotus Component Design, Project zero). Describe in detail how the workflow model has been incorporated in the software and discuss the advantages and shortcomings resulting from this implementation.

10. Workflow analysis

In this section, analyze the current workflow as it is modeled today, identify portions of the workflow where some optimization could be introduced, and discuss how and why you may want to optimize this part. This section also consists of a general discussion section where all the pro's and contra's of the current workflow vs. envisioned workflow are discussed.

Table 6-1 shows principal features of the case studies developed to validate the method and software proposed in this thesis. After, we describe two complete case studies.

Chapter 6. Validation

Title	Subject	Area	Principal Tasks	Organizational units involved	Jobs	Process relationships	Patterns	Date
Requesting a credit to buy car	This case study details the process to get a credit to buy a car.	Information systems	16	5	5	3	4	September 2007
Credit car request	This case study details the process to get a credit card	Information systems	36	10	13	6	10	September 2007
Order personalized compression stockings	This case study deals with an Internet order system, allowing the ordering of personalized support stockings.	Information systems	22	4	7	3	5	January 2008
Organization of a Triathlon	This case study explains all the steps needed to organize a sport event	Information systems	72	14	28	6	12	January 2008
Organization of '24h vélos de Louvain-La-Neuve'	This case study deals with the organization of a cycle race	Information systems	55	7	18	6	9	January 2008
Tech Support for Copy Machines	This case study is about the way that a copy machine gets maintenance	Management	17	3	7	3	3	June 2008
Airport passenger workflow	The case study illustrates the steps that a passenger need to follow in order to do the check-in at airport	Management	20	4	5	6	6	June 2008
How to get my driving license	This case study shows the steps to follow in order to	Management	25	3	8	6	4	June 2008

Chapter 6. Validation

	get a driving license type "B" in Belgium							
Buying process in a company	This case study is about the manner in which an organization deals with a buying order	Information systems	16	5	8	3	9	June 2008
Managing a private hospital	This case study presents how a private clinic is managed	Information systems	20	5	9	4	3	June 2008
Organizing conferences	This case study analyzes how people organize the program of small conferences by using a review tool	Management	22	3	3	3	7	July 2008
Installing hardware at Allianz group	This case study presents how an organization deals with the acquisition of hardware	Information systems	36	5	8	4	8	December 2008
Container transport by ships, trains and trucks	This case study shows how an organization gives the service of transporting container from an A point to a B point	Information systems	41	3	5	4	13	January 2009
Manufacture of adhesives in the company MAC-tac	This case study is about the manner in which an organization deals with a buying order	Information systems	38	8	14	5	8	January 2009
Creating a collaborative blog	This case study deals with the problem of creating a blog for a research group	Information systems	11	5	4	2	3	March 2009

Chapter 6. Validation

	working at same university but in different departments							
Workflow of MDC Group	This case study focuses in the Consulting part, this is done by offering several options to the client, giving courses and consulting about several fields: Logistics and events, Design and development of product, Industrial Processes, Sub hiring of staff, Payroll Management, Recruitment and Selection, Evaluation of Human Talent.	Management	23	4	5	6	8	June 2009
Replenishing shelves at Carrefour, Belgium	This case study manages the way to replenish shelves in a supermarket	Management	25	6	10	3	10	September 2009

Table 6-1 Case studies features

6.2.1.a Case study 1. *Order personalised compression stockings over Internet.*

Introduction

The case study is situated in the phlebology domain. It deals with an Internet order system, allowing the ordering of personalized support stockings. The main idea of this system is to calculate a 3D model of the customer's legs from a series of digital pictures taken from his/her legs. This model will be sent, coupled with a specific order, via Internet to the manufacturing department. This system is currently used in Germany by the Bauerfeind AG under the name of "Image 3D".

The calculation of the 3D model is done with a technology developed by the corpus.e AG, a society in which several distant family members of the student are involved, which explains the personal interest in this case study. The information on the system was quite vague so that several assumptions have been made.

Scenario

This case study is based on support stocking order system currently used in Germany. The goal of the system is to better serve customers in need of elastic support stockings. The support stockings are personalized, based on the 3D model generated. They will be better adapted to the customer's morphology than stockings based on manually taken measurements could ever be. The system uses a proprietary 3D digitalization technology developed by the corpus.e AG, situated in Stuttgart. The information transfer between the three involved sites is done via Internet. The three sites are the following: the shop located anywhere in Germany, the servers of the corpus.e AG situated in Stuttgart (Germany) and finally the Bauerfeind production site located in Zeulenroda (Germany).

This service is commercialized by the Bauerfeind AG under the name "Image 3D". The main users of this system will be the employees of the shops participating in the Image 3D program.

Description

1st step: The client visits a specialized shop with his medical prescription. The shop, located anywhere in Germany (assumed to be located in X), must be equipped with the Image 3D material and have access to the Bauerfeind Internet Ordering system.

2nd step: In order to indent for a pair of personalized stockings the 3D model of the patient's legs must be built. The patient must dress in special stockings (having a photogrammetric marking). The patient then climbs onto a rotary disc. The employee releases the latch and takes a series of 10 pictures of the patient's legs, each picture taken from a different view (by turning the rotary disc). Finally the disc is arrested again and the patient can take of the special stockings.

Chapter 6. Validation

3rd step: The digital camera is placed on a docking station connected to the terminal. On the terminal the MagicalSkin Scanner program is started and by the use of several buttons the pictures are transferred to the terminal. The terminal checks the quality of the pictures locally. If the quality is found satisfactory the pictures are upon request of the employee (after login) compressed and sent to the servers of the corpus.e in Stuttgart. In Stuttgart the 3D model of the customer's legs is calculated 100% automatically and then sent back to the shop in X. In the meanwhile employee and customer choose the model and color of stockings.

4th step: The 3D model can be visualized on the terminal in the shop in X using specially designed software. The employee can now initialize a new order. He enters different information (Name, First Name, Address, Birthday, Sex, Size, Weight, Health Insurance Company, Social Security Number) on the customer and on the desired stockings (Compression class, Design, Product, Band, Version, Color, Toehold, Quantity) and sends this to the Bauerfeind servers in Zeulenroda.

5th step: The 3D model is compared with the standard stockings produced by Bauerfeind. The results are sent back to the shop in X. The results are displayed on the terminal. Employee and customer then decide whether to order a standard or a personalized product. The employee enters the customer's choice into the terminal (Standard or Personalized, if Standard the employee enters also Size and Design).

6th step: The employee finally commits the order. It is then sent to the Bauerfeind servers at Zeulenroda. Here the order is automatically inserted in the order database and in the products database. It is then passed on to the planning system. This system analyses the different database entries (products database) and decides which pairs to produce when and on which machine.

7th step: When the moment of production has arrived, an employee of the production department fetches the necessary material from the stock (using a fork-lift) and introduces it into the knitting machine.

8th step: The knitting machine, connected to the system, extracts the list of stockings to produce, their sizes as well as their identification number from the database. The machine starts the production of the different pairs. At the end of the knitting process each pair is marked with a barcode.

9th step: The stockings are then transported by an employee of the production department to the sewing where they are finalized by an employee specialized in sewing.

10th step: Next an employee of the production department will transport the stockings to the dyeing machine where he introduces each pair into the dyeing machine. The machine then dyes the stockings.

11th step: The finished pairs are transferred to the logistics department by an employee of the production department. There they are packaged by hand (logistics

Chapter 6. Validation

department employee). Prior to the packaging the barcode of the pair is read and the barcode as well as the clients name and the shop's address are printed on the wrapping. The employee of the logistics department will use the barcode to retrieve the delivery type chosen for the stockings.

12th step: In the case of an ordinary delivery the pair of stockings will be sent with the next charge in the region of the shop. Meanwhile the stockings are stored at the service together with other pairs for the same shop. The decision on which transport the stockings will be taken along depends on an employee of the logistics department.

13th step case A: The deliverer, employee of the Bauerfeind logistics department as well, fetches the charges and personalized stockings in planned intervals. The logistics department employee tells the deliverer which pairs to take along. At a given moment he will take the personalized pairs along.

14th step case A: The deliverer takes a predefined itinerary. He uses his van for the transport. He delivers the ordered pairs at each shop where an employee will accept them.

13th step case B: In the case of an express delivery the pair is sent by post. In this case an employee of the logistics service must prepare the parcel to be sent. The address of the shop having originated the order is indicated on the parcel.

14th step case B: As a great number of parcels are sent each day an employee of DHL (since the Deutsche Post uses exclusively DHL for its parcel transport) comes to fetch the parcels once a day. He loads the parcels into his van and takes them to the closest DHL depot.

15th step case B: At the depot the van is unloaded. The parcels are introduced into the sorting system. This system reads the destination addresses on each parcel and gives each parcel an identification number. An entry is created for each parcel in the national parcels database. A barcode, representing its identification number is printed on each parcel.

16th step case B: Next the system checks whether a zip code is given in the destination address. If yes, the system decides where to send the parcel (to other depot or to the recipient). If not this decision has to be taken by an employee. The system sorts the parcels by destination.

17th step case B: The parcel is taken to the recipient shop or to another depot where the previous task is repeated.

18th step case B: Finally, the parcel is attributed to a deliverer. He delivers the parcel personally at the shop where an employee accepts it.

19th step: The employee contacts the customer. The customer comes to the shop to try the stockings. If the stockings fit, the customer pays. Else, another order will be done.

Chapter 6. Validation

Identification of tasks					
ID	Task name	Pred.	Definition	Rationale	Nature
1	Picture taking	/	The employee takes a series of ten pictures of the customer's legs covered with the special stockings.	Chgress-materiel - terminal +spec.stocking + camera + rotary disc	User
2	Uploading pictures	1	The employee uploads the series of pictures taken to the terminal connected to the system.	Chgress-mat. -spec.Stocking - rotary disc + terminal + docking station	Interactive
3	Quality verification	2	A local application verifies the quality of the uploaded pictures.	Chgress-person - employee Chgress-materiel - docking station	System
4	Sending pictures	3	If the quality of the pictures is OK, the employee triggers the sending of the pictures to Stuttgart.	Chgress-person + employee Chgress-materiel - camera Decision-Waiting-Point "quality"	Interactive
5	Choice of model	/	The customer and the employee choose together the model of stockings to be ordered.		User
6	Calculation and sending of 3D model	4	The 3D model is calculated and sent back to the shop.	Chgress-materiel - terminal Chgress-person - employee Chgress-info + 3D model Chg-place X --- Stuttgart	System
7	Customer identification	5,6	The employee identifies the client using a terminal connected to the system.		Interactive
8	Entering data and sending	6,7	The employee enters the different choices of the customer into the terminal. He sends them to the servers of the Bauerfeind AG together with the 3D model.	Chgress-materiel - terminal Chgress-person - employee Chgress-info + 3D model Chg-place X --- Stuttgart	Interactive

Chapter 6. Validation

p	Comparison with catalogue and sending	8	The model and the wishes of the customer are compared to the available stockings. The list of suitable stockings is sent back to the shop.	Chgress-materiel - terminal Chgress-person - employee Chg-place X --- Zeulenroda	System
10	Finalization of order and sending	9	The employee encodes the choice of the customer and sends the finalized order back to the Bauerfeind AIG.	Chgress-materiel +terminal Chgress-person +employee Chgress-info +suitable standard models Chg-place Zeulenroda --- X	Interactive
11	Encoding in databases	10	On the Bauerfeind production site the order is stored in different databases.	Chgress-materiel -terminal + database Chgress-person -employee Chg-place X --- Zeulenroda	System
12	Planning	11	The planning system chooses the best moment for the production of the ordered stockings.	Chgress-materiel +planning system	System
13	Preparation of machine	12	An employee fetches the necessary material from stock and prepares the knitting machine.	Chgress-materiel +fork-lift Chgress-person +employee Chg-temporal Interruption	User
14	Knitting of stockings	13	The machine knits, among other, the ordered pair of stocking.	Chgress-materiel -fork-lift +knitting machine Chgress-person -employee	System
15	Sewing of toehold	14	An employee sews close the toehold of each stocking.	Chgress-materiel -knitting machine +sewing machine Chgress-person +employee	User
16	Dyeing of stockings	15	An employee sorts the stockings by color, prepares the coloring machine and colors the stockings.	Chgress-mat. - terminal - sewing machine +dyeing machine +barcode reader Chgress-person - employee + employee	User

Chapter 6. Validation

17	Packaging	16	An employee queries the destination of each pair of stockings. He packages them and prepares the parcels. The parcels are then sorted by destination and stored until delivery time.	Chgress-mat. + terminal + database + barcode reader - dyeing machine - barcode reader Chg-place Production Dep. --- Logistics Dep.	Interactive
18A	Delivery A	17	A driver delivers the ordered pairs at shop.	Chgress-mat. - database - barcode reader + van Chgress-person - employee + driver Chg-temporal Interruption	User
18B	Transport post	17	A postman comes to fetch the parcels to be sent by post and takes them to the closest DHL depot.	Chgress-materiel - database + van DHL Chgress-person - employee + employee DHL Chg-temporal Interruption (< 12h)	User
19B	Sorting by machine	18B	The sorting machine sorts the parcels after their ZIP code.	Chgress-materiel - van + sorting machine Chgress-person - driver Chg-place Logistics Dep. --- DHL Decision Waiting Point "Next Destination"	System

Chapter 6. Validation

19C	Sorting by hand	18B	A post employee sorts the parcels by hand after their ZIP code.	Chgress-materiel - van Chgress-person - driver + employee Chg-place Logistics Dep.--- DHL Decision Waiting Point "Next Destination"	User
20B	Transport	19/C	The parcels are brought to a DHL depot closer to their destination.	Chgress-materiel + van Chgress-person - driver Chg-place Post A --- Post B	User
20C	Delivery B	19B/C	The parcels are delivered to the shop.	Chgress-materiel + van Chgress-person - driver Chg-place Post B --- X	User
21	Notification customer	18A,20C	The customer is notified	Chgress-materiel - van Chgress-person - driver + employee	User
22	Fitting of stockings	21	The customer comes to fit on the stockings. If they are convenient he pays and takes them home	Chgress-person + customer Chg-temporal Interruption	User

Task modeling

In the following pages the task trees for the first 10 tasks are provided. We chose this part of the workflow because it is the most interesting one with respect to the data exchanged and because we had the most precise information on this part.

Picture taking

The very first task in the workflow is to take the pictures of the customer's legs in order to calculate the corresponding 3D model. The pictures must correspond to a given profile. The legs must be covered in special stockings and a series of pictures needs to be provided to allow the calculation of the model. The employee must first determine the customer's stocking size, fetch the stockings and hand them to the customer. The customer then steps onto the rotary disc. The employee now undoes the latch and switches on the camera; there is no given order between these two actions. He then takes a picture and turns the disc by one position. After having taken 10 pictures the disc is arrested. The actions involving the rotary disc and the camera are mechanical.

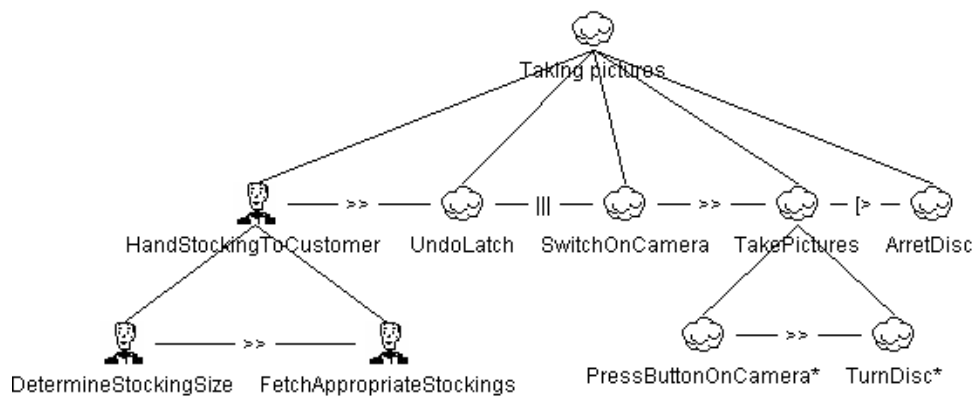


Figure 6-2. Taking picture - task model.

Uploading pictures

The next task is the uploading of the pictures just taken. To do this the employee must start the MagicalSkinScanner computer program. Next the camera is placed onto the docking station. The docking station is switched on and the transfer is started by pushing the "OK" button. The Starting of the program and the connection to the Docking Station can be done in any order. However one action will be finished before the other is started in order to avoid confusion. All the other actions must be executed sequentially. The actions involving the terminal are interactive, while those involving the docking station can be seen as mechanical.

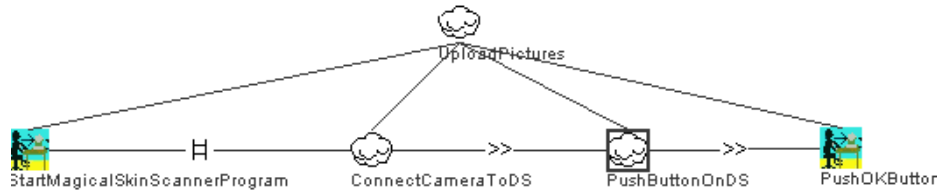


Figure 6-3. Upload pictures - task model.

Quality verification

The consequent task is fully automatic. An image processing program verifies whether the quality of the pictures uploaded is sufficient to calculate the 3D model. This action involves simply the check in itself and the return/output of the result to the terminal.



Figure 6-4. Quality verification – task model.

Sending pictures

The SendingPictures task consists in the sending of the uploaded (and verified) pictures to Stuttgart, where the 3D Model will be calculated. This must be triggered by the employee. The employee first needs to login which consists in entering Username and Password in any order. Both actions can be executed completely concurrently. The employee should push the "Jetzt" button after having entered his Login Information to achieve the goal of this action. After the button has been pushed the pictures are sent to Stuttgart, in fact pushing the button triggers the sending. Thus those two actions are sequential. Only the sending of the data is automatic, the other actions are interactive.

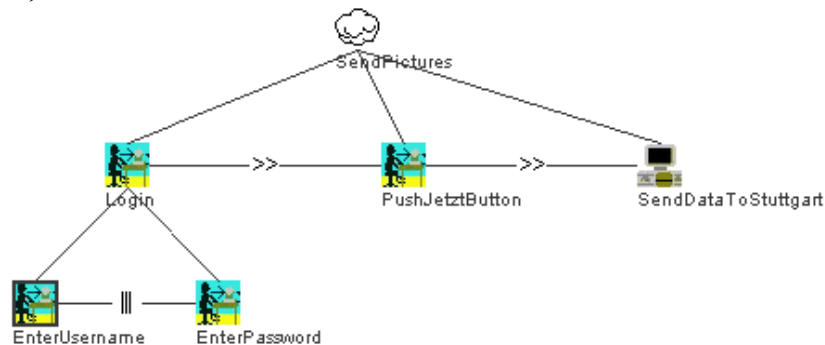


Figure 6-5. Send pictures – task model.

Choice of model

The choice of the stockings model to order is a task realized in cooperation by the employee and the customer. Together they consult the catalogue and take a decision. The viewing of the catalogue is optional. The customer might already know what he wants. Also, both actions (consulting and taking decision) could very well be executed concurrently.

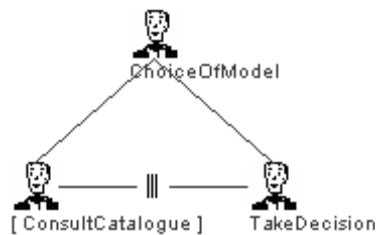


Figure 6-6. Choice of model -task model.

Calculation and sending of 3D model

The next task is the calculation of the 3D model of the customer's legs. This is done fully automatically at the servers of the corpus.e AG in Stuttgart. After calculating the model the data is sent back to the shop.



Figure 6-7. Calculate 3D model - task model.

Entering data and sending

In the next task the employee needs to enter a lot of different information. Since the task tree is too big to figure in this document, the different sub-trees will be provided separately.

Four first actions:

The employee must push two buttons sequentially and then choose his name from a list. He then pushes the "Auftrag Hinzufügen" Button to add a new order.

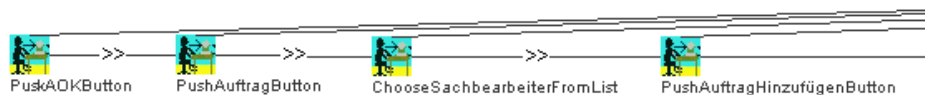


Figure 6-8. Enter data (1) - task model.

Chapter 6. Validation

The next step is the identification of the customer. The employee can either enter all information as new, or, in the case of a returning customer, search for him in the list of customers.

Enter data

There are several fields to be filled in. Some fields are optional however.



Figure 6-9. Enter data (2) - task model.

After the employee has entered name, first name and gender of the customer, the program automatically searches for already registered customers with the same particulars. It then returns the list of those customers. The first three actions can be executed totally concurrently. The search begins only after those three actions have been terminated and the results are only returned after the search has been completed. The employee can now either select a customer from a list or continue with entering information (see next picture).

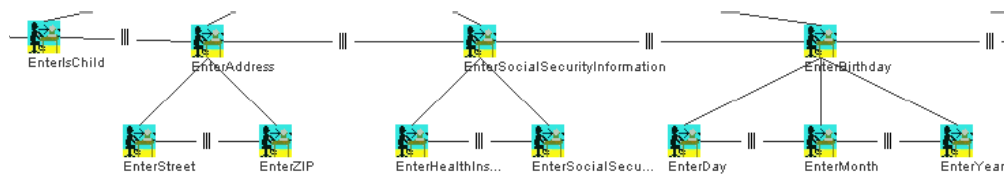


Figure 6-10. Enter data (3) - task model.

All the above information can be entered concurrently. The employee can optionally enter a comment.

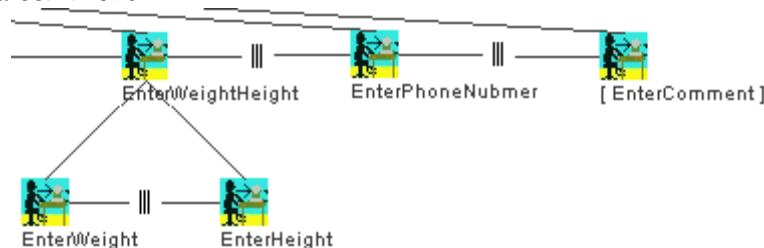


Figure 6-11. Enter data (4) - task model.

Search customer

The search customer action comprises 3 further actions that need to be executed sequentially. First the employee must push the "List" Button. The terminal automatically computes the list of customers. Next the employee will select a customer from the proposed list. He then pushes the "OK" button.

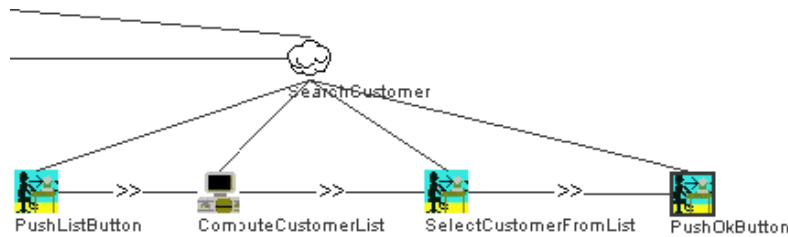


Figure 6-12. Search customer (1) - task model.

The IdentifyCustomer action is sequentially followed by the PushWeiterButton action, whose name is self-explanatory. The next action is the ProductConfiguration.

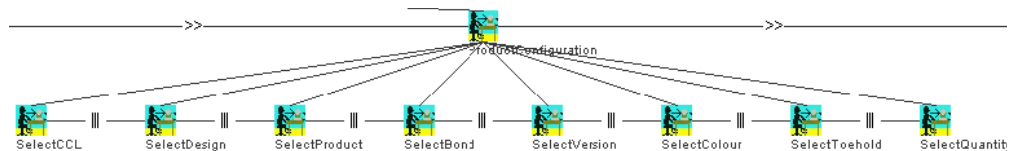


Figure 6-13. Search customer (2) - task model.

Here the employee enters different information about the stockings to be ordered in the provided form. ProductConfiguration is followed by three more actions which are also executed sequentially. The employee pushes the "OK" Button and then the "Berechnen" Button. The program now automatically sends the information entered in the previous steps along with the 3D model to the Bauerfeind servers.

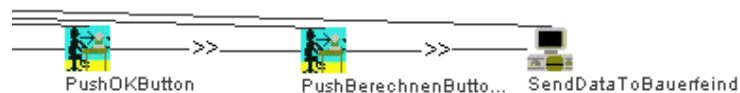


Figure 6-14. Search customer (3) - task model.

Comparison with catalogue and sending

On the Bauerfeind servers the sent data is compared to the catalogue of the available products. The result of this comparison is sent back to the sender, i.e. the shop. All this is done automatically and sequentially.

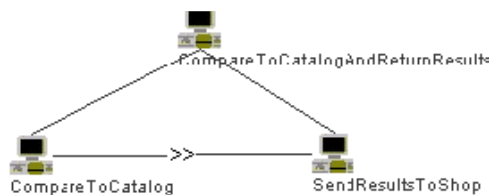


Figure 6-15. Compare to catalogue - task model.

Finalization of order and sending

This is the last task that takes place at the shop. According to the results returned by the previous task the decision is taken to order personalized or standard stockings. The next action is either the ordering of standard or of personalized stockings. To order standard stockings the employee first selects the size and confirms with the "Serienanfertigung" button. For the personalized stockings he simply pushes the "Sonderanfertigung" button. Finally the "Abschicken" button confirms the order.

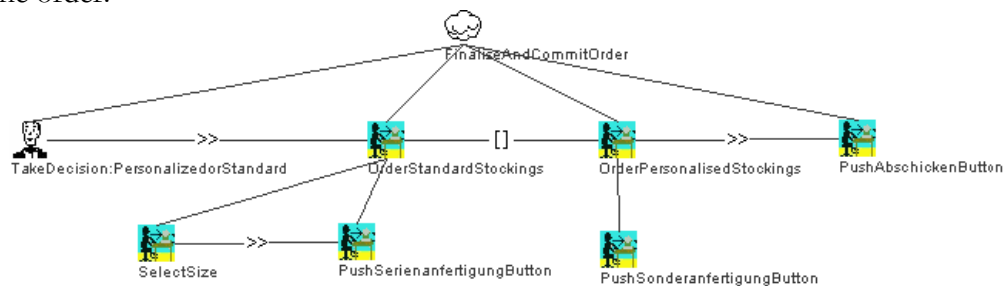


Figure 6-16. Finalization of order - task model.

Packaging

The packaging task only takes place later in the workflow. However in order to demonstrate how the task tree of such a less interactive task looks like: The employee (of the Bauerfeind logistics department) will read the bar code printed on the stockings. This will return the database entry of the pair of stockings. He then prepares the wrapping which implies the printing of the shop name and customer name. The package is then either prepared for postal delivery or for standard delivery.

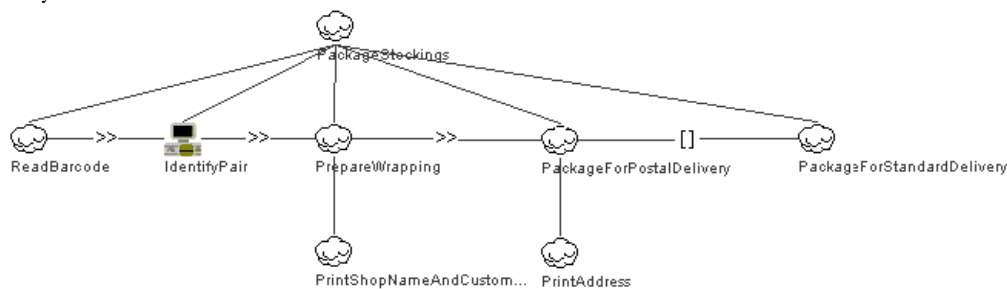


Figure 6-17. Packaging - task model.

Organization modeling

We can discern 4 separate organizations in the workflow: the shop, the Corpus.e AG, the Bauerfeind AG, and the Deutsche Post.

The Bauerfeind AG itself is subdivided in several departments.

Chapter 6. Validation

The **Shop** is the organization performing the first tasks of the workflow. It is any specialized shop working in cooperation with Bauerfeind and being connected to the Image 3D System.

Bauerfeind is a producer of medical aids, and in our case, the producer of the compression stockings. The Image 3D System belongs to Bauerfeind, who is thus, the main organization of our organizational chart. Bauerfeind has three main departments, **Order Management Dep.**, **Production Dep.** and **Logistics Dep.** As Bauerfeind produces not only stockings but also other medical aids belonging to other domains the production department is subdivided into four further departments: Prosthetics, Phlebology, Orthopedics and Podiatry.

The **Prosthetics Department** produces artificial limbs while the **Phlebology Department** is responsible for compression stockings. The **Orthopedics Department** produces braces and the **Podiatry Department** shoe lifts.

Of these four departments only the Phlebology Department is relevant for our workflow.

Corpus .e is an independent company having developed the technology used to calculate the 3D models. Since this is a proprietary technology the calculation itself is still done at the servers of Corpus.e, it can be seen as a service provided to the Image 3D System.

Finally the **Deutsche Post** is the last organization. The Deutsche Post is the national German Post Service. The Deutsche Post also offers parcel deliveries which are completely handled by **DHL**, which can thus be seen as a department of the Deutsche Post.

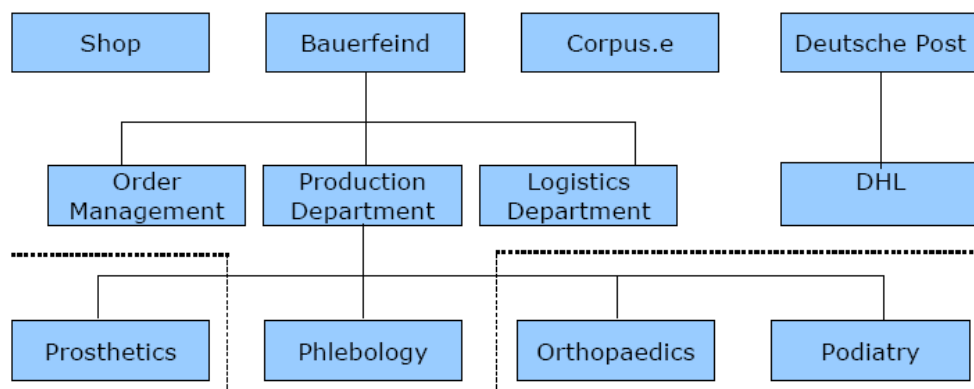


Figure 6-18. Organization chart.

Chapter 6. Validation

Task resource specification

Resource	Attributes	Job	Associated tasks
Customer		Customer	Choice of model Fitting on of stockings
Shop employee A Any employee of the shop having the equipment as well as the web application and having a user account	Task experience 2-5 System experience 2-5 Motivation 1-4	Customer Handling A	Customer identification Picture taking Uploading pictures Choice of model Entering data and sending Finalization of order and sending
Terminal program A software with image processing capabilities running on a terminal	Task experience 5	Verificator	Verify the quality
3D calculating program The software responsible for the calculation of 3D models	Task experience 5	3D calculating	Calculating and sending of 3D models
Order management software The software handling the incoming orders or comparison queries	Task experience 5	Order management	Comparison with catalogue and sending Encoding in database
Planning software The software scheduling the production	Task experience 5	Project software	Planning
Bauerfeind production employee A An employee knowing how to handle and prepare the different machines	Task experience 2-5 System experience 2 Motivation 2-4	Producer A	Preparation of machine Dyeing of stockings
Knitting machine Special machine for the knitting of stockings	Task experience 5	Knitter	Knitting of stockings
Bauerfeind production employee B An employee having the knowledge and ability to sew the toehold	Task experience 2-5 System experience 1 Motivation 1-4	Producer B	Sewing of toehold
Bauerfeind logistics	Task experience 1-5	Logistic A	Packaging

Chapter 6. Validation

employee An employee trained in the use of the code bar reader and terminal	System experience 1-5 Motivation 2-4		
Bauerfeind driver A driver for the Bauerfeind company	Task experience 3-5 System experience 1 Motivation 2-4	Logistic B	Delivery A
Deutsche post employee A A driver and employee of the Deutsche post or DHL	Task experience 3-5 System experience 2 Motivation 2-5	Postman A	Transport post Transport Delivery B
Parcel sorting machine Machine used to sort parcels by ZIP code	Task experience 5	Delivery machine	Sorting by machine Sorting by hand
Shop employee B An employee of the shop, no special knowledge however high hierarchical level	Task experience 4-5 System experience 2-4 Motivation 4-5	Customer Handling B	Notification customer Fitting of stockings

Process Models

We can identify 6 different processes in the workflow; they are reproduced in the following table:

Name	Tasks	Level
Selling	CustomerIdentification PictureTaking UploadingPictures SendingPictures ChoiceOfModel EnteringDataAndSending FinalisationOfOrderAndSending ChoiceOfModel FittingOnOfStockings	Production
Planning	Planning	Managerial
Production	PreparationOfMachine DyeingOfStockings KnittingOfStockings SewingOfToehold	Production

Chapter 6. Validation

Packaging	Packaging	Production
Bauerfeind Delivery	Delivery A	Support
Express Delivery	TransportPost Transport DeliveryBS SortingByMachine SortingByHand	Support

Workflow resource patterns

PictureTaking

The allocation of this resource is done based on the role. Only an employee trained in the use of the equipment and the computer program can handle this task. However there are multiple employees having this knowledge, the task is offered to all of them the moment the customer expresses his wishes.

Uploading Pictures - FinalisationOfOrderAndSending

All the tasks in this interval, except those not executed at the shop, are allocated based on case-handling. The same employee who took the pictures will handle these tasks. The tasks are allocated to a single resource on starting.

CalculationAndSendingOf3DModel

This task can only be allocated to the Corpus.e software running on their servers in Stuttgart. Thus the allocation is direct, and to single resource. As the program reacts to service demands the task is allocated only on starting.

ComparisonWithCatalogueAndSending

This task falls under the same principle as the Calculation of the 3D model. It is directly allocated on starting of the task.

Planning

This task is also directly allocated since there is but one instance of the software handling it. However, given the magnitude of the task it is scheduled early.

PreparationOfMachine

The knitting machine must be prepared by a trained employee. Thus this allocation is role based. The allocation is done early and offered to multiple resources. This allows the employees to have a clear schedule at the beginning of their day.

Chapter 6. Validation

KnittingOfStockings

This task is allocated to one of the multiple knitting machines. The allocation is history-based and to a single resource. This allows distributing the work equally among the machines. The task is allocated on starting when the machine fetches its next job from the database.

SewingOfToehold

The sewing of the stockings is a task directly allocated. A deferred allocation would make more sense at this place.

ColouringOfStockings

The coloring of the stockings is again a role based allocation. It is, for the same reasons as for the preparation of the machine offered to multiple resources and allocated early.

Packaging

The same principle as previously explained holds for the packaging task.

Delivery A

The delivery task is allocated based on history. It is offered to a single resource and early allocated. This task is allocated history-based to ensure that each driver has the same route each time. It is furthermore allocated early to allow the driver to have a schedule before the beginning of each task.

TransportPost

The task TransportPost is allocated based on the role. The employee must be a trained driver. The task is offered to multiple resources and allocated early for the same reasons as mentioned in the previous task.

SortingByMachine

The sorting by machine task is directly allocated as there is only one machine. This task is allocated on starting.

SortingByHand

The sorting by hand is also role based. It is allocated merely on starting since this is a task that is not always necessary.

Transport and Delivery B

Chapter 6. Validation

For these two tasks the same patterns apply as for the Delivery A. The reasons to choose this pattern are the same as for this previous task, as the three tasks are very similar.

NotificationCustomer

The notification of the customer should be done by the responsible of the shop. Thus this allocation is hierarchy based. It is offered to multiple resources as there might be different suitable employees. It is allocated on starting

FittingOfStockings

Finally the FittingOftheStockings is allocated using the case-handling pattern. The same employee who handled the customer's order earlier on will assist in the fitting of the now delivered stockings. This task is allocated on starting, the moment the customer enters the shop.

Graphical Representation of Workflow:

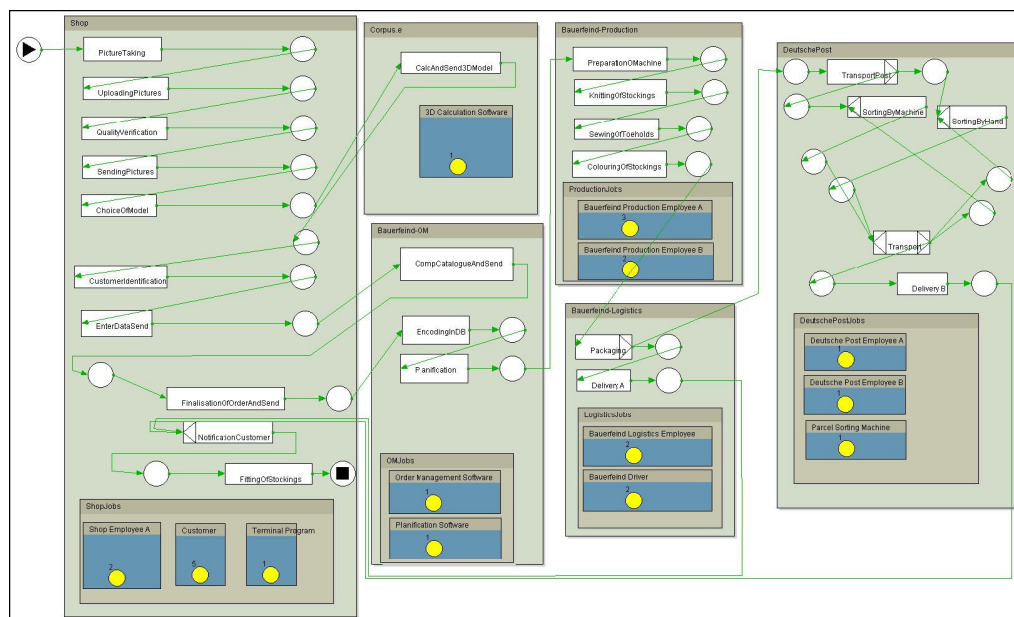


Figure 6-19. Workflow.

UI Definition

Since the User Interfaces for the first tasks of the workflow are already clearly defined in the Image 3D User manual, the user interfaces used in the Production and Logistics Department will be specified in this document. The models of these

Chapter 6. Validation

tasks are all quite abstract as no specific information is known about those interfaces, thus the following correspond to hypothetical interfaces.

Preparation of machine:

In this task the employee will first trigger the information gathering. Upon trigger the next charge to be produced will be fetched from the database. The information on the necessary material will be displayed. After having introduced the material, the employee will trigger the knitting.

Information to be displayed: ID of each pair of stockings, type of fabric.



Next Lot to be knitted:

IDs of stockings

Fabric Type

Figure 6-20. Preparation of machine UI.

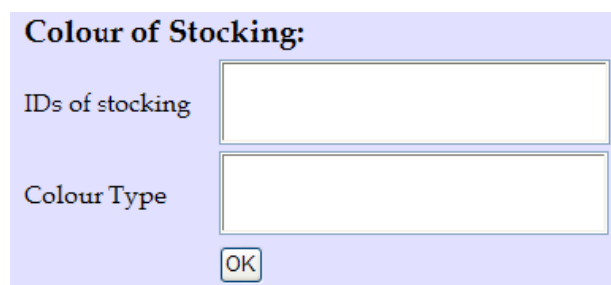
Sewing of toehold:

As this consists simply in sewing together every pair brought no interaction with the system is necessary.

Dyeing of stockings:

This consists in first sorting the different stockings after the color they are going to be dyed. This involves fetching this information from the database using a barcode reader. The employee then reads the necessary information from a screen (color id) and prepares the machine.

Information to be displayed: ID of pair, color



Colour of Stocking:

IDs of stocking

Colour Type

Figure 6-21. Dyeing of stockings UI.

Packaging:

This task involves again identification using the barcode reader. The address and dispatching information of the pair of stockings is displayed on the screen. The employee can print the address and will after packaging set the status of this pair to "waiting for dispatching". Information to be displayed: ID of pair, address, dispatching type.

Packaging

Stockings ID

Address

Dispatching Type

Figure 6-22. Packaging UI.

User Interface Flow

The following diagram shows the user interface flow for the above described interfaces. The first user selects the resource (himself, thus he accepts the task) and prepares the machine. The following user again accepts a task and then dyes the stockings. The third user again accepts the task and wraps the stockings in a package. The last user finally decides who will be the driver for the charge. This allocation is history-based and the allocation of this task is binding.

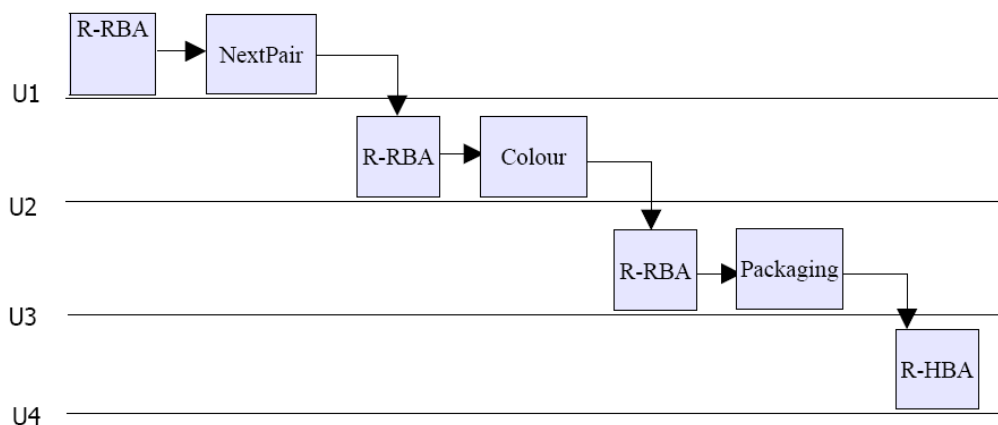


Figure 6-23. User interface flow.

6.2.1.b Case study 2. *Requesting a credit to buy a car.*

Introduction

Traditionally bank services were just relegated to bank branches. In that scenario, there were just bank's employees those able to perform bank's tasks. For instance, open an account, make money transfers, etc. However, the increasing market of e-commerce has made the banks to move towards a new market e-banking. In this new scenario banks need to change or adapt their process in order to satisfy clients' demands. Specifically, there is a set of products which are nowadays available using bank internet portals. Once you are a bank client it is possible to purchase or acquire: insurances, credits, saving accounts, among other services.

In this work we will focus just in analyzing the workflow of how to request a credit to buy a car. We consider this service interesting as it involves at least three different organizations (the bank, car agencies, car buyer-seller company), related in a strategic joint venture alliance. Car agencies and the car seller-buyer agency are benefit from the bank credits. So, in any case, the allied, the bank, appear as an entity of the organization and for the clients is transparent this second institution. Similarly, bank clients can chose, almost, any car model directly from the agencies or find an option for a second hand car.

The information was gathered by analyzing different bank web sites, where this kind of credit is offered, so as, car agencies and car dealers' web sites. In addition, based on our experience working in a bank, information about the bank internal process is well known.

Scenario

To give or to get a credit to buy a car is nowadays very common; usually you can get a credit with a bank or car agency.

Bank scenario

In this section we will focus on the bank credit from the bank perspective. Typically bank clients can go to a bank branch in order to get information about the car credits. However, with the popularity of Internet this information can be obtained for bank portals. In addition, users can also take advantages to do more than just checking the services provided by the bank. They can really purchase cars using e-commerce applications, next section describes this front-end that the bank offers to their clients to buy cars.

External bank user scenario

External users (bank clients in this example) have different options when they access the e-commerce application of a bank. Specifically, referring to credits to buy cars the set of options, which is not limited, could be:

- Information. This section is divided in:
 - Product information. Product information refers to all the relevant data related to a credit to buy a car. This means, description of: benefits, requirements, selection criteria, frequently asked questions, etc.
 - Used / Second hand cars. Similarly as previous section, it includes description of: benefits, rules, requirements, selection criteria, frequently asked questions, etc. Related to buying used or second hand cars or even to sell a car.
- Links of interest. This section provides extra information to the users.
 - Buying tips. Buying tips is a section intended to help users to decide which car to buy. By explaining the advantages and disadvantages of buying new or used cars the users can take better decisions.
 - Compare your car. Compare your car with other cars is an interesting feature if you have doubts between more than one options. Two categories can be defined for the comparison, price and model.
 - Nearly new cars. This link is directly a connection to another web site which is related to the bank but still is independent. This second web site is dedicated to sell and buy used cars and allows users to use bank credits for the payment.
 - Credit simulator. The credit simulator is a tool that helps the user to simulate a credit to buy a car. Some parameters are needed such as: make, model, year, submodel, then the system calculate the price and can show, if the user indicates, extra information like: series, comfort, security, exterior, options.
- Choose your car. The “choose your car” tool is intended to show cars descriptions without making a simulation of the credit. This kind of search can be done by using the same parameters as in the previous section, such as: model, year, submodel.

Chapter 6. Validation

Internal bank user scenario

Inside the bank there is a complete different scenario than from external users. In fact the acquisition of a credit launches a set of procedures inside the bank organizational structure. Notice that the previously described scenario applies also to a second category of users, bank workers in a branch. The extra characteristic that a bank worker can do is to gather all the documents that prove the information the client is providing to request a credit.

So, once a request is submitted to the systems, the credit department needs to evaluate if the client has a good behavior as debtor of any credit company, this information is normally gathered in the bureau of credit. Once the data were checked, the next step is to contact the client, if this was not previously done, and make an appointment to collect the documents that prove the information s/he submitted on her/his demand. Once the documents are in possession of the bank, the client signs the credit request.

The next step is on the investigation department that needs to verify references of the client. In parallel the credit department evaluates the capacity of payment of the client. Once the client's information is checked and her/his capacity of payment generated, the credit department decides whether the client can be trust, based on the information s/he provided, maybe using a data based and some data mining techniques to check if her/his profile is trustable for the bank.

The resolution, credit refused or approved, is then notified to the client. In the acceptance scenario, the bank announces, also, to the car agency, in the case a new car was bought, and request a car. The client then must make a down payment, if the client has account in the bank they used that account so that they make their payments monthly; in opposite case the client must open an account with the bank.

Once the car is ready to be delivered; the bank pays to the agency and successfully obtain the car's documents. The bank contacts the client to make an appointment in order to deliver the car and to successfully obtain the clients' signatures for the car agency and for the bank. Once all the documents were signed the representative of the bank obtains the invoice and sends it to the credit department.

The client deposits her/his monthly payment in the bank account until the credit is completed, then the client is going to the bank to recover its invoice of the car and the bank finishes and closes the credit.

Cars Agency scenario

The bank is specialized in credits but operations of selling and buying cars require a specialized entity, independent of the bank, with the knowledge and infrastructure to support this kind of transactions. The bank can have a direct interaction and agreement with cars' distributors to sell new cars. However, negotiations with sellers and buyers, demands a car dealer. Joint venture agreements are typically ways of solving this lack of specialization for a particular company. Both sides of the society are benefit from the new services they can provide, the bank extends its range of services and the car dealer is benefited with the bank credits.

The lists of services of a car dealer, not limited, are:

- Used cars. A list of features to handle operations of used cars. Users can announce your car, a guide of prices, tips to buy, the services they offer and the ways to get a Bank credit.
- Available cars. "*Available cars*" is a list showing a navigation of the data base of cars. This is a way of quickly access to available cars. Users select the main category which can be the maker; the type of vehicle (for instance armored) then the system provided the elements found in the previously specified category. The set of elements are direct links to specific details of the cars selected. If one item is selected, for instance, Cadillac car, then a table with the description of all the available Cadillac cars is presented.
- Find used cars. These feature, allows user to use more precise ways of searching cars. By defining characteristics, such as: maker, range of price, characteristics included (credit, images, from agency, from a particular, etc.), location, range of year, range of mileage.
- Preference display. These are means of distributing the information of a search in a table, for instance, recently added, recommended or cheapest.

Elicitation from textual scenario

With the model elicitation tool (section 5.2.1), we can classify the workflow elements involved in this case study (Figure 6-24).

Chapter 6. Validation

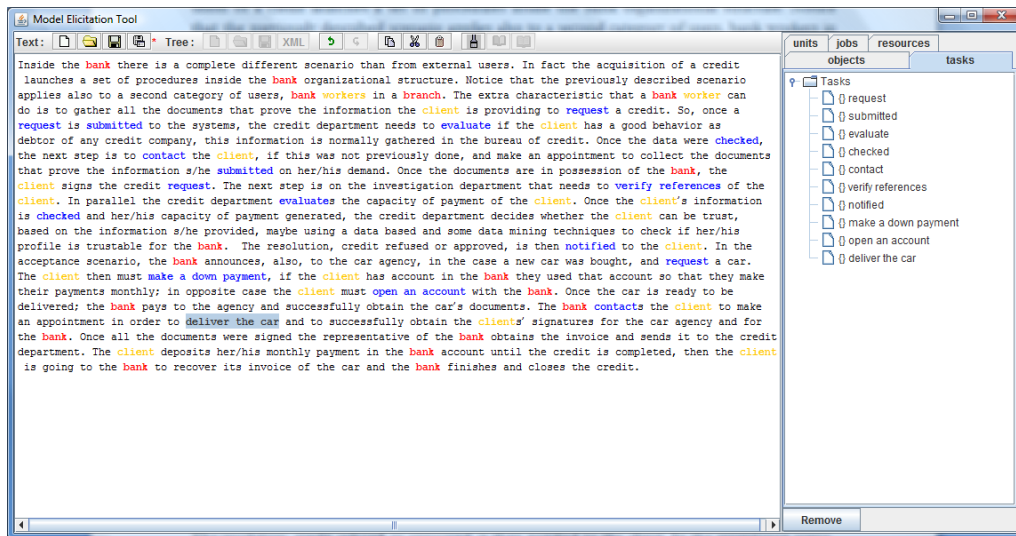


Figure 6-24. Elicitation.

Identification of tasks

Next step is the identification of task in the tool. Figure 6-25 shows the table with the different attributes to identify tasks; in order to present this identification of tasks in a legible way, we use the “Export excel” option of the tool to showing the complete Table 6-2.

Task Table					
File		About			
ID	Task name	Predicate	Definition	Nature	Rationale
1	Navigate Links of in-terest	/	This section provides extra information to the users such as buying tips, intended to help users to decide which car to buy.	The task is Interactive when the user request information but automatic when the system communicates the results of the	Links of interest is part of the services offered by the bank. Takes place in the eBanking service, just one user is involved, the pe-riod of time to perform this task is variable depends on the user.
2.1	Credit simulator	/	The credit simulator is a task that helps the user to simulate a credit to buy a car. Some parameters are needed such as	The task is Interactive when the user set the parameters but automatic when the system calculates and communicates	The task is iterative, as the user can navigate through the information
2.2	Financial simulator	2.1	The financial simulator is a task in which the client calculates their payment capacity	The task is Interactive when the user set the parameters but automatic when the system calculates and communicates	The credit simulator task is part of the services offered by the bank. A very useful tool for clients to identify the options offered by the bank and test the different payment plans.
2.3	Financial plans	2.2	The bank processes the data provided by the user and determines the financial plans	The task is Interactive when the user select the plan but automatic	The financial simulator task is part of the services offered by the bank. It is relevant to fulfill the request credit task.
2.4	Request the credit	2.3	Once defined the financial plan the user can formally request the credit. The client fulfills a form with their personal data name	The task is Interactive when the user set the parameters.	Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user.
3	Choose your car	/	The "choose your car" task is intended to show cars descriptions. This kind of search can be done by using some	The task is Interactive when the user set the parameters but automatic when the system communicates the results of the	The financial plans task is relevant to fulfill the request credit task. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user.

Figure 6-25. Task identification.

Chapter 6. Validation

ID	Task name	Predicate	Definition	Nature	Rationale
1	Navigate Links of interest.	/	This section provides extra information to the users such as: buying tips, intended to help users to decide which car to buy, <u>Nearly new cars</u> , a connection to another web site which is directly related to the bank but still is independent. This second web site is dedicated to sell and buy used cars and allows users to use bank credits for the payment. It is possible to request this info by email.	The task is Interactive when the user request information but automatic when the system communicates the results of the queries.	Links of interest is part of the services offered by the bank. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user can navigate through the information as much as they want even none, consequently is an optional task.
2.1	Credit simulator	/	The credit simulator is a task that helps the user to simulate a credit to buy a car. Some parameters are needed such as: maker, model, year, submodel, and then the system calculate the price and can show, if the user indicates, extra information like: series, comfort, security, exterior, options.	The task is Interactive when the user set the parameters but automatic when the system calculates and communicates the results of the queries.	The credit simulator task is part of the services offered by the bank. A very useful tool for clients to identify the options offered by the bank and test the different payment plans. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user can simulate as many configurations as they want even none, consequently is an optional task.
2.2	Financial simulator	2.1 Credit simulator	The financial simulator is a task in which the client calculates their payment capacity depending on the car selected. The client established the down payment percentage or the quantity they can afford and the system determine the remaining quantity that will be the credit. The user specifies their monthly incomes, direct salary, and from other sources,	The task is Interactive when the user set the parameters but automatic when the system calculates and communicates the results of the queries.	The financial simulator task is part of the services offered by the bank. It is relevant to fulfill the request credit task. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user change the values of the attributes

Chapter 6. Validation

			and their monthly payments from other debts. The system calculates the monthly payment capacity. The user can select any other car at any moment. Then the user send the information to check the financial plan that the bank can offer to them		and create as many configurations as they want.
2.3	Financial plans	2.2 Financial simulator	The bank processes the data provided by the user and determines the financial plans that the bank can offer to the clients, for 12 to 48 months of credit. At any moment the client can return to the previous step or send their selection.	The task is Interactive when the user select the plan but automatic.	The financial plans task is relevant to fulfill the request credit task. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user change the financial plan desired as many times as they want.
2.4	Request the credit	2.3 Financial plans	Once defined the financial plan the user can formally request the credit. The client fulfills a form with their personal data, name (first name, last name), email (two times), address (street, number int and ext, colony, delegation/commune, city, state, postal code), phone number (number, preferable time to contact the client, type of phone: house, mobile, work) and comments. The client can clear the form at any time. The task end when the user sends the form.	The task is Interactive when the user set the parameters.	The request the credit task is part of the services offered by the bank. It is the last step to formalize the request of the credit. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user change the values of the attributes and create as many configurations as they want.
3	Choose	/	The “choose your car” task is intended to	The task is Interactive	This task is part of the services offered

Chapter 6. Validation

	your car		show cars descriptions. This kind of search can be done by using some parameters such as: maker, model, year and submodel.	when the user set the parameters but automatic when the system communicates the results of the queries.	by the bank. It is a list of prices and car descriptions. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user can check as many cars as they want even none, consequently is an optional task.
4	Compare your car.	/	Compare your car with other cars is an interesting feature is your intention is to sell your car. Two categories can be defined for the comparison, price and model.	The task is Interactive when the user set the parameters but automatic when the system communicates the results of the queries.	This task is part of the services offered by the bank. Takes place in the eBanking service, just one user is involved, the period of time to perform this task is variable depends on the user. The task is iterative, as the user can compare as many cars as they want even none, consequently is an optional task.
5	Receive request in line	2.4 Re-request the credit	So, once a credit request is submitted to the systems, the credit department starts formally the procedure. Once received the request it proceeds to distribute work to the different organizational units for investigation. It needs to send an email to the client to arrange a meeting to collect the documents that prove the information provided in any of the bank branches. So the system should provide an option to the client, normally the closest to its house.	The task is Interactive when the user accepts the task but automatic when the system communicates to the others	This is a coordination task that is important to disseminate work within the bank. A worker of the Credit department received the request information, evaluates its content, the system generates an id to the request and generates the following tasks: verify the status at the credit bureau; send an email to the client to request him a meeting.
6	Communicate with the client	5 Receive request	The bank system sends an email to contact the client, if this was not previously done, and make an appointment to collect the documents that prove the information he submitted on his	The task is Interactive when the user set the parameters but automatic when the system commu-	The bank needs to contact the client and to propose him a meeting to collect all the probative documents regarding the credit requested. The task can be

Chapter 6. Validation

		in line	demand. In this task the client, after receiving an invitation email, confirms or programs a meeting by selecting the branch of the bank that accommodates him the most. Then the user sends all the data to the bank.	nicates the results of the queries.	iterative while the user search for the most appropriated branch. Or even optional because the user could previously provide all the documents so this task is optional at this level. It takes place at the eBanking web site.
7	Obtain customer's documents	6 Communicate with the client	This is the official collection of the documents which are necessary to start the investigations to award a credit. The bank worker checks the information the client provided, using the eBanking system, or fulfills the data. If everything is correct then the client signs the formal request and the bank worker sends the file to investigation.	The task is Interactive when the bank worker captures or checks in the system all the client's data.	There is a need to physically check the documents so as to get the signature from the client. This task takes place on the Bank branch is mandatory, and it is the bank worker who is in charge of performing the task.
8	Verify state in bureau of credit	7 Obtain customer's documents	The task verifies state in bureau of credit refers to investigate in the bureau of credit data base whether the client has a good behavior as debtor of any credit company or not.	The task is Interactive.	There is a need to evaluate if the client has a good behavior as debtor of any credit company; this information is normally gathered in the bureau of credit can be accessible in the Credit department It is just one user who is responsible of this task which can be iterative as the parameters of search can be changed.
9	Verify customer's references	7 Obtain customer's documents	This task consists on the investigation to confirm that the client works, lives where they said. Also the phone numbers are checked, etc.	The task is manual but interactive to confirm the data to the system.	This task is part of the investigation that is mandatory before grant a credit. It takes place in the investigation department. The task can be split in different subtasks, due to the fact that there is no specific time in which data can be checked, so as there is no need to assign this to one single resource.

Chapter 6. Validation

10	Evaluate request	8 and 9 Verify state in bureau of credit Verify customer's references	The credit department with the confirmation of the data provided by the client, with the information of the historical credit behavior, can make a decision whether grant the credit or not. The results of the evaluation are sent to the client and depending on the results to actions are taken. If the credit is rejected, the reason is explained to the client by email, the result is also notified to the branch worker that collected the information. If the credit is accepted then the client and the branch worker are notified, so as a notification is sent to the credit department to start the procedure of acquiring the car from an agency.	The task is interactive.	This step is crucial and the decision must be reflected in the system. It takes place on the Credit department. It is just one person in charge of the task that using a system that basically will support his decision.
11	Buy a car from a car agency	10 Evaluate request	This task refers to contact the car agency and start the procedure of buying a car. An employee of the credit department receives the information regarding the credit and the car desired. They contact the car agency sending an email with the information of the car that the bank wants to buy.	The task is interactive.	This task is relevant as it takes place in a different time than the previous one. The employee of this task can be anybody at the credit department not necessarily the one who accepted the grant of the credit.
12	Car agency receive request to buy a car	11 Buy a car from a car agency	The car agency receives the request from the bank. They look into their stock data base to determine when it is possible to deliver the car. It informs the result of the search and sends to the bank the details of the cost, the account number for the deposit and the delivery details.	The task is interactive.	A worker in the car agency checks the request from the bank and determines whether they can provide the service or not.
13	Bank notifies the client	12 Car agency	The bank notifies to the client that he can make the transfer of the corresponding amount of the down payment.	The task is interactive	The bank cannot proceed unless they receive the down payment of the car. This task can be iterative in specific

Chapter 6. Validation

		receive request to buy a car			periods of time
14	Customer makes a down payment	13 Bank notifies the client	The client makes the down payment for the car, either using a bank branch or the eBanking system. As soon as the bank detects this payment the credit department receives the payment order.	The task is interactive	The client must make the down payment after continuing with the procedure. Either using the eBanking system or at any branch.
15	Bank pays to the agency	14 Customer makes a down payment	The bank is ready to transfer the money to pay the cost of the car and arrange a meeting with the client to go to pick the car together to the car agency.	The task is interactive	This task takes place at the credit department, in which a worker makes the money transfer to pay the car and send mails to the bank branch, to the worker in charge of the credit and to the client.
16	Deliver the car to customer	15 Bank pays to the agency	The bank representative and the car agency representative deliver the car to the client, the client signs the documents, the bank	This task is interactive when registering to the system	This task takes place in the car agency. The client gets its car. The bank ends with this task the process to grant a credit.

Table 6-2. Results of the task identification for the Bank.

Organizational units

From our scenario we identify the organizational units that are relevant for the system. We show them in organizational chart.

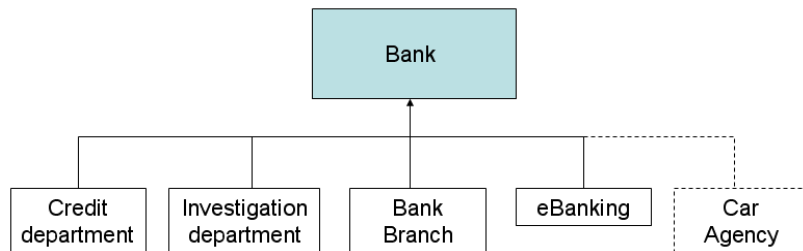


Figure 6-26. Bank structure.

Once the elicitation part has been done, we can use the “organizational tree” sub-menu of the tool and obtain the next view:

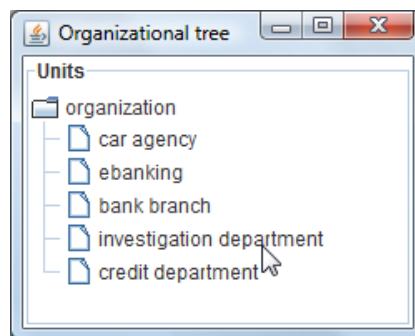


Figure 6-27. Organizational tree.

Jobs specification

We can use the next window to add the jobs in the tool.

The screenshot shows a window titled "Job Editor". It has a "Choice" tab selected. On the left, there is a "Job selection:" label and a dropdown menu with "client" selected. On the right, under the "Edit existing job" section, there are several input fields: "Name:" (with "client" entered), "Specifications:" (empty), "Family:" (empty), "Grade:" (empty), and "Privileges:" (empty).

Figure 6-28. Job editor view.

Chapter 6. Validation

In our scenario we identify the following jobs:

Job	Organizational Unit	Associated task
Customer. It is any client of the bank which is registered in the database of the bank and has access to the EBanking services.	EBanking	Navigate Links of interest. Credit simulator. Financial simulator. Financial plans. Request the credit. Choose your car. Compare your car, Customer makes a down payment
	Bank Branch	Verify customer's references, Bank notifies the client, Customer makes a down payment
Credit representative. Is on charge of attending all the procedures of the credit department.	Credit department	Receive request in line, Communicate with the client, Verify state in bureau of credit, Evaluate request, Buy a car from a car agency, Bank pays to the agency
Sales representative. It is a person in charge of the sales in a bank branch.	Bank branch	Obtain customer's documents, Verify customer's references, and Credit simulator. Financial simulator. Financial plans. Request the credit.
Investigator is a job in charge of investigates the data provided by clients and determine whether is true or not.	Investigation department	Verify customer's references
Sales representative of the car agency representative is a job responsible on the sales of car in the car agency.	Car agency	Car agency receive request to buy a car, Deliver the car to customer

Table 6-3. Jobs identification.

Process model

We provide the process corresponding model resulting from the organization of processes according to identification criteria, assembling rules, and workflow patterns. As knows, the structure of a Petri Net has only one final state; nevertheless, we want to avoid some large and confusing lines (arcs) in Figure 6-29 adding 2 final states.

Chapter 6. Validation

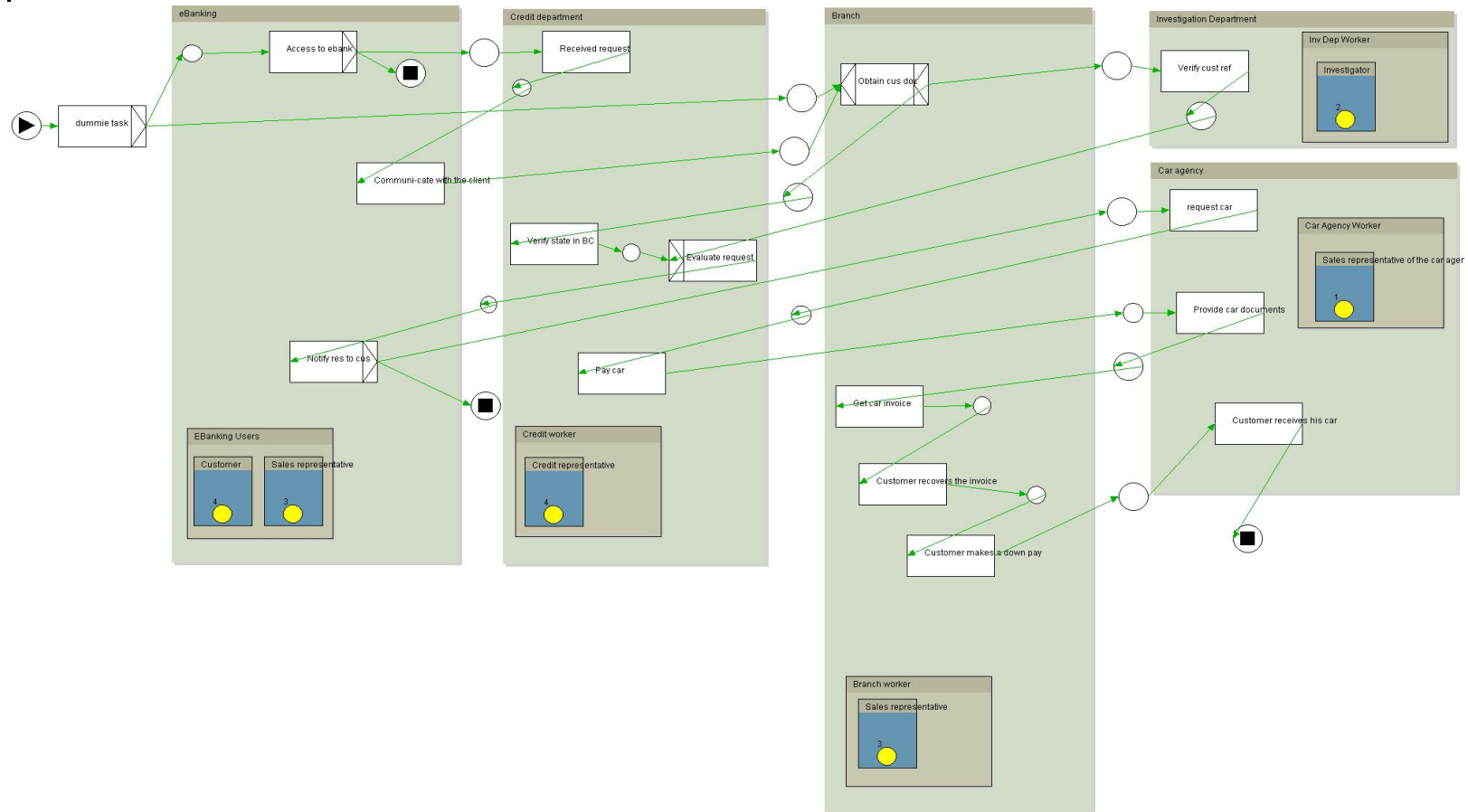


Figure 6-29. Process model view.

Workflow resource patterns

Access to EBanking.

This includes the entire EBanking task, the allocation is *deferred*, there is no way to assign the task to a specific client of the bank, they login the system randomly so allocation will be made in Run-Time and it can be any. There is no distribution type, as the task cannot be delegated; the person will know that they are going to perform the task so the distribution is early.

Receive request in line.

This task has for allocation *capability-based* pattern. Is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Communicate with the client.

This task has for allocation *retain familiar* pattern if the task was performed by a bank worker at a branch so the worker who starts the procedure normally continues with it. If the client starts the process on the web then the task is *capability-based*. The task is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Obtain customer's documents.

This task has for allocation *retain familiar* pattern as a bank worker already was selected to perform this task in the previous step. The task is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Verify state in bureau of credit.

This task has for allocation *capability-based* pattern. Is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Verify customer's references.

This task has for allocation *capability-based* pattern. Is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Evaluate request.

This task is allocated to a different credit representative as the first that took care of the request, this is a bank policy, so the allocation is a *separation of duties and capability-based*. Is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Buy a car from a car agency.

This task is allocated to a different credit representative as the first that took care of the request and to the one who did the evaluation, this is a bank policy, so the allocation is a *separation of duties and capability-based*. Is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Car agency receive request to buy a car.

This task has for allocation *capability-based* pattern. Is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Bank notifies the client.

This task has for allocation *retain familiar* pattern as a bank worker already was selected to perform this task in the previous step. The task is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Customer makes a down payment.

This task has for allocation *retain familiar* pattern as a client was already defined. The task is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Bank pays to the agency.

This task has for allocation *retain familiar* pattern as a bank worker already was selected to perform this task in the previous step. The task is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Deliver the car to customer.

This task has for allocation *retain familiar* pattern as a bank worker already was selected to perform this task in the previous step. The task is allocated to a single resource and the distribution is on starting. There are no other jobs that can be authorized to perform this task.

Task modeling

In the following pages the task trees for some tasks are provided. We chose this part of the workflow because it is the most interesting one with respect to the data exchanged.

E-Banking

All tasks identified to the EBanking system are depicted in the task model. *Show buying tips* is a task that provides extra information to the users in order to find

Chapter 6. Validation

what they are looking for. Is just informative and the system display what is in a data base.

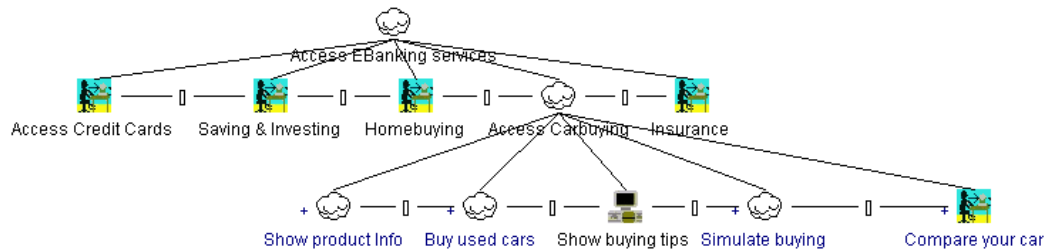


Figure 6-30. E-Banking task model.

Show product information

This section provides extra information to the users such as: buying tips, intended to help users to decide which car to buy, you can ask this info by email but you must be registered to the system. So, after requesting the data you must sign in or sign up. At this level you can also check the terms and conditions of the credits or directly if you one to try simulate a credit.

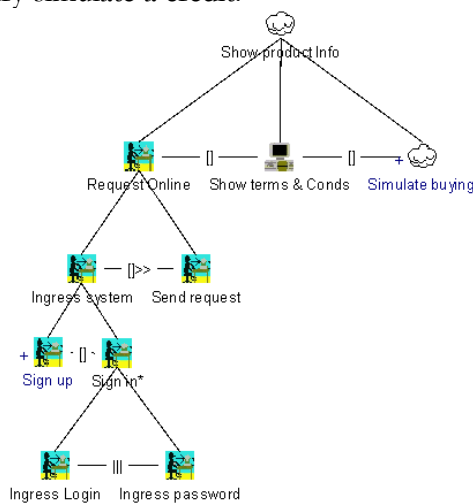


Figure 6-31. Show product information task model.

The *credit simulator* is a task that helps the user to simulate a credit to buy a car. Some parameters are needed such as: maker, model, year, submodel, and then the system calculate the price and can show, if the user indicates, extra information like: series, comfort, security, exterior, options. We called this task request more info (Figure 6-32). Once this information is collected then the client calculates their payment capacity depending on the car selected. The client established the

Chapter 6. Validation

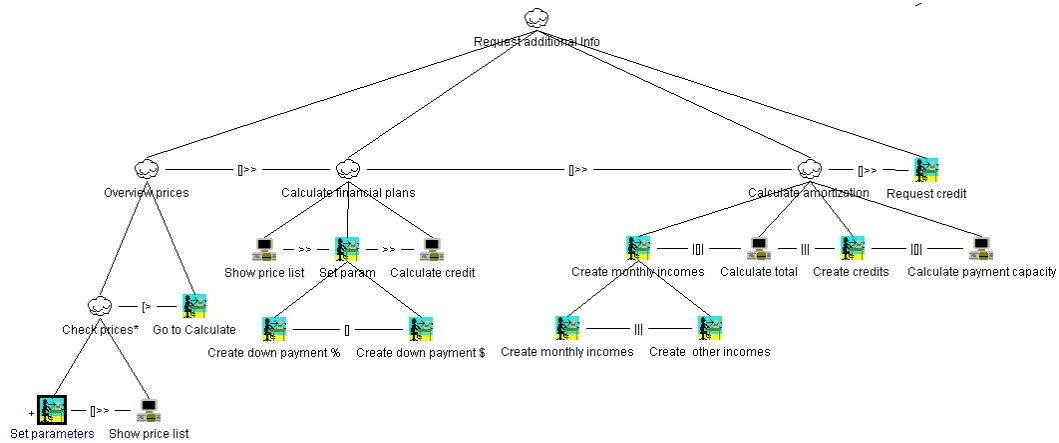


Figure 6-33. Request additional information task model.

Buying used cars

A nearly new car is a task that makes links to connect to another web site which is directly related to the bank but still is independent. This second web site is dedicated to sell and buy used cars and allows users to use bank credits for the payment.

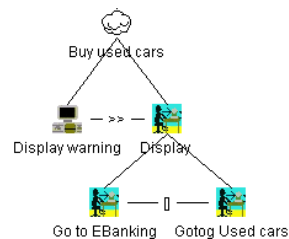


Figure 6-34. Buying used cars task model.

Compare your car

This task is intended to show cars descriptions. This kind of search can be done by using some parameters such as: maker, model, year and submodel. Compare your car with other cars is an interesting feature is your intention is to sell your car. Two categories can be defined for the comparison, price and model.

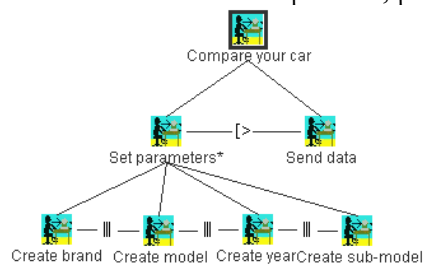


Figure 6-35. Compare your car task model.

Chapter 6. Validation

User interface definition

In section 4.2.3.a the method for developing UIs was presented, following that method from tasks models defined above we can obtain the UIs corresponding to this case study.

Figure 6-36 represents the principal page that the user has when s/he access to credit to buy a car option.

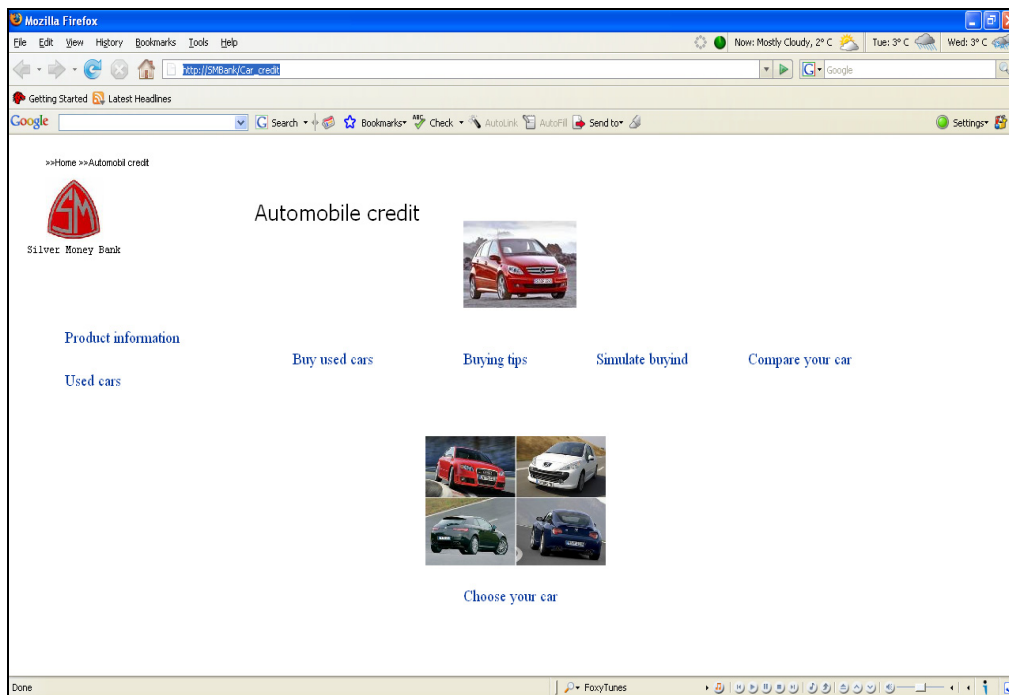


Figure 6-36. Credit to buy a car page.

When the user goes to the option “Simulate buying” s/he has access to this page.

Chapter 6. Validation

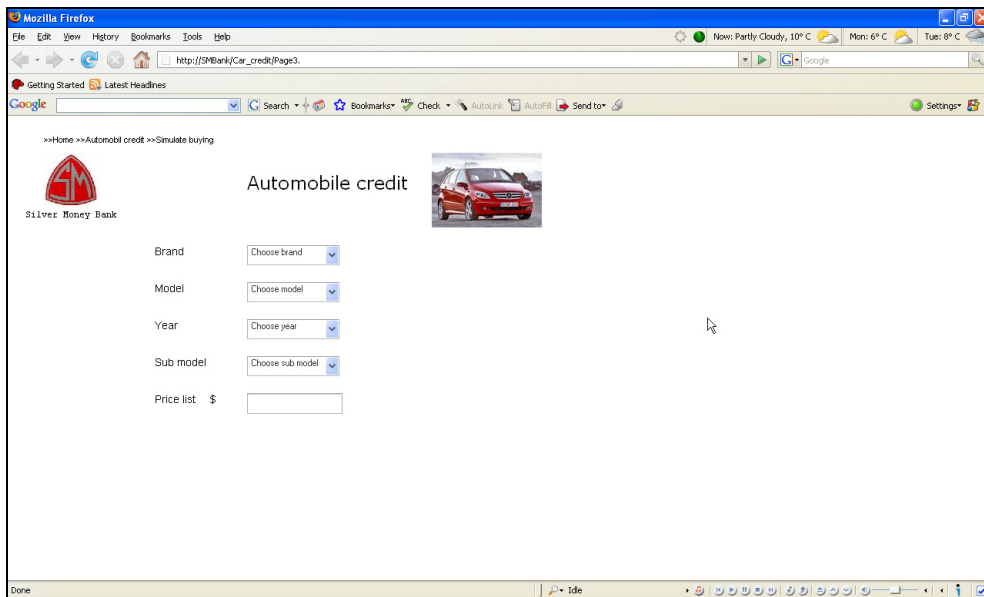


Figure 6-37. Simulate buying page.

When the user wants to apply for acquiring a credit for a car, s/he needs to submit her/his personal information; in this UI it is possible to do it.

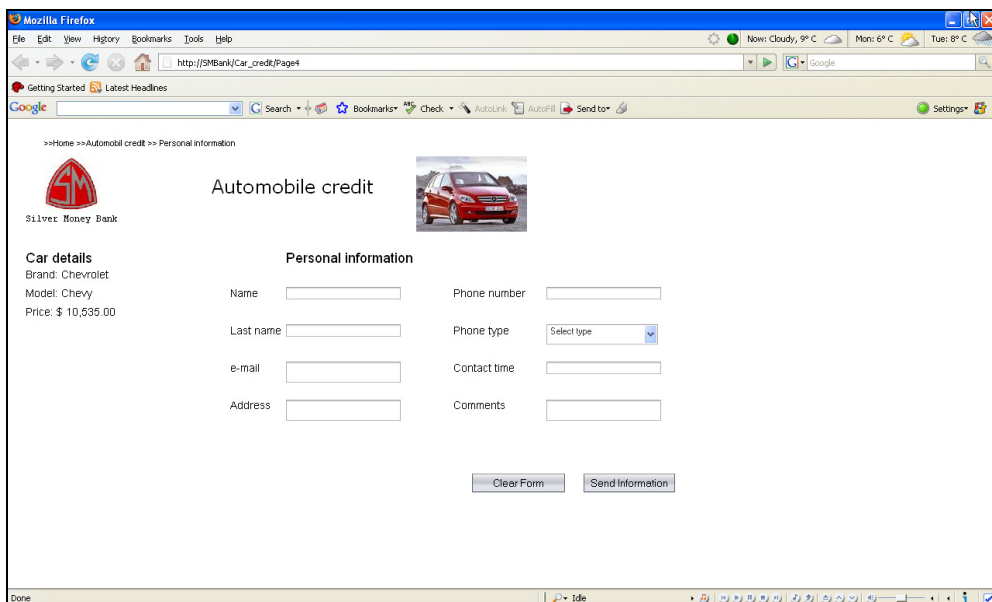


Figure 6-38. Personal information page.

The rest of UIs can be generated following the same principles illustrated in this thesis.

6.3 Internal validation

6.3.1 Requirements validation

The internal validation of the method proposed in this dissertation consists in assessing its characteristics against a set of selected requirements described in section 2.7.2. We provide a score of the perception we have about the percentage of accomplishment of the requirements. We considered that we fully satisfy the requirements. When there is nothing to add we put the score of 100%; 90% is assigned when small details are missing; 80% when missing parts are for more consideration.

Modeling requirements

1. *The ontology must be extensible.* The ontology structure should allow the extension to workflow and task concepts.

Comment: This can be done, as the set of transformational rules is not fixed or closed, there is always a way to change the rules, the path, etc. also it is possible to extend this methodology. Score: 90%.

2. *Expressivity.* The context models should, at least, provide enough details to allow an implementation of the system it describes.

Comment: Concepts at workflow model provide enough details to cover the aspects involved in organizations. This has been namely illustrated in Chapter 3 and in the case studies of section 6.2.1. The expressivity of our task model outweighs the one of CTT and IdealXML, the reference formalisms that were chosen to represent user's tasks. Score: 90%.

3. *Organizational modeling.* Strict separation between the description of the organization and workflow process specification.

Comment: One of the characterizing aspects of this method is the separation of concerns between modeling the organization and modeling the workflow processes within a given organization. From the organizational perspective, the following aspects have to be considered: the structure of the organization, the identification of a task resource using the workflow management system, the authorization for a given userStereotype to perform a given task, from the workflow perspective. These perspectives are illustrated on the basis of the concepts defined in the meta-model in Figure 3-6, which shows details of the organizational model. Score: 90%.

4. *Management of agendas.* Allow user to optimize distributed tasks providing them the capability to store and manage a flexible personal TODO list of tasks,

and provides the manager the capability to manage his own task list.

Comment: This requirement is achieved thanks to the *agenda*, *agenda item*, *work list*, and *work item* elements described in the meta-model, sections 3.2.2 and 3.2.4. Score: 90%.

5. *Support carrying out group tasks.* From the individual level continuously throughout the global level: individual, within groups, for the group as a whole, among groups, within organization, and among organizations.

Comment: This requirement is covered with the task model for representing the individual level, and the workflow model considering group requirements. Score: 80%.

6. *Support multiple ways to support a group task.* In principle, there should not be a unique way to carry out a single group task, but several mechanisms should be offered for this purpose. If a mechanism is no longer available, another one should be selectable.

Comment: By exploiting task model descriptions different scenarios could be conducted. Each scenario represents a particular sequence of actions that can successfully be performed to reach a goal (section 4.2.2.b). Score: 80%.

7. *Support the group evolution over time.* When the group evolves over time, the workflow definition should be easily maintained and reflected in the system.

Comment: *Job*, *userStereotype*, and *organizationalUnit* components (section 3.2.4) characterize group members; they are specified during organization design, then this part of the workflow model is capable for supporting any type of change and be propagated over the complete method. Score: 80%.

Language requirements

8. *Machine readable.* States that the proposed ontology should be legible by a machine.

Comment: This requirement is completely met by the definition of an XML syntax enabling the expression of the concepts of the ontology and in compliance with the abstract syntax defined for this ontology. The collection of tools that manipulate UsiXML format is an evidence of the machine readability of this syntax. Score: 90%.

9. *Extendibility.* It refers to the ease of adapting a conceptual structure to the occurrence of newly elicited concepts. HCI is a vast area covering the definition of multiple types of interfaces, interaction techniques, and interaction con-

texts. A specification language should be equipped with extension mechanisms to allow its evolution in parallel with the artifact it seeks to model.

Comment: Each model of UsiXML framework is defined independently of the others. Extending this framework to other models is possible by simply incorporating the concepts into the framework. This clear structuring facilitates the introduction of new concepts. Score: 100%.

Method requirements

10. *Method explicitness.* States that the component steps of our method should define in a comprehensive way their logic and application. This requirement is motivated by the lack of explicitness of the existing approaches in describing the proposed transformational process.

Comment: This requirement is ensured by the decomposition of the transformational approach into development steps (section 4.2). Score: 100%.

11. *Identification criteria* introduced a set of precise criteria that can be used in order to identify a task in a textual scenario.

Comment: This requirement is achieved defining a group of standards to identify tasks and process (sections 4.2.2.a and 4.2.2.b). Score: 100%.

12. *Consider a complete task life cycle.* Considering all the possible states in order to create a complete task life cycle from the beginning to the end.

Comment: An expanded task life cycle was proposed in section 4.2.2.e trying to cover all the possible states that task goes through. Score: 90%.

13. *Flexibility in creating UIs.* It is important to generate UIs for workflow in a manner which the platform is independent, customizable, and extensible.

Comment: Relaying on both UsiXML and Cameleon Reference Framework that gather platform independent concepts. Score: 90%.

Software requirements

14. *Support software for method for specifying workflow.* It refers to the possibility to develop software to support the method for specifying workflow information systems.

Comment: This requirement is achieved thanks to the software developed specially to be compatible with this method. The software is divided in *workflow editor* and *workflow manager* trying to cover the dynamic and static aspects of the method. Score: 80%.

15. *Support software for method for developing user interfaces.* It refers to the possibility to develop software to support the method for developing workflow user interfaces.

Comment: This requirement is achieved relaying on the assembly of UsiXML transformation tools that can be used to support the transformation from one model to another. Score: 90%.

6.3.2 Evaluation of the graphical notation

The evaluation of graphical notations is not an exact process which often results in variegated and/or subjective conclusions [DeBo06]. However, it gives us hints to improve the notation and the quality of the designs it produces. Since the early conception of the tool and during its, still under construction, evolution there have been significant changes always based on users feedback after intense used of the tool.

We rely on the cognitive dimension framework (CD) [Gree89] for the evaluation of the graphical notation. Accordingly to the authors [Gree98], cognitive dimensions are a tool to aid: non-HCI specialists in evaluating usability of information-based artifacts (summative evaluation); and designers to prompt possible improvements (formative evaluation). The name artifact refers to a combination of three elements: the notation, environment and medium. For our evaluation the notation is consistent with the environment where it is represented, and the medium is always a computer screen.

In our case we strive for the formative evaluation as our ultimate goal of applying CD analysis is to evaluate FlowiXML design with respect to the impact that they will have on its users. In this section, concepts related to CD framework are in *italics* just to denote that when we are compliant to their vocabulary.

The CD framework comes with 14 dimensions which focus on different aspects of the notation, each of which is cognitively relevant, giving a 'profile'. The profile determines the suitability for various tasks [Gree98]. Each dimension can either be positive or negative, depending on the application in which the notation is applied. This context is referred as the *type of activity* from which we will get the preferred profile. Green [Gree98] distinguishes four possible applications they are listed in Table 6-4. FlowiXML is relevant to the four *types of applications*.

As stated before the CD framework defines 14 cognitive dimensions. We explore them and determined which of them were of interest for our artifact. The results are summarized in Table 6.5.

Many of the selected cognitive dimensions are related to each other. This means that changes in one can affect other. Many of the cognitive dimensions are pair wise independent: any two can be varied independently, as long as some third dimension is allowed to change freely.

All details of this evaluation for each dimension are in Appendix C. We just discuss the viscosity dimension in the next paragraphs.

Viscosity is a property of the system as a whole. It refers to resistance of the system to change. This means that it becomes hard after modifying our model to get a desire state. Changes may be related to different operations, such as: adding, removing, modifying, consequently viscosity may be very different depending on the operation and the operator. For instance, viscous is the operation of removing an organizational unit while fluid could be adding a task to a process. It is known that viscosity might be problematic in an exploratory design if not tackled correctly [Gree98] for at least one reason: redesigning in a graphical editor usually requires much tedious work, and frequently many similar alterations need to be made to different parts of the picture.

Adding, removing and/or modifying text in the model elicitation tool is considered less viscous for all the advantages that it offers, such as: automatic identification of action verbs and the *transcription* of textual scenarios into a task spreadsheet.

Adding, removing and/or modifying a task in the task spreadsheet editor is considered less viscous there is no affection on the task list when an operations is done on the task identification, on the contrary the *transcription* of the changes to the workflow editor is automatically done.

Adding, removing, and/or modifying a job from the job editor is considered viscous because of its impact in the model might require manual adaptation of the process model to determine the role in charge to perform the task previously assigned to other roles. Similarly, adding, removing, and/or modifying a user stereotype is considered viscous because of its impact in the model. Likewise, when operations are on organizational units where a whole restructuration of the workflow might be needed if an organizational unit disappear. In any case organizational changes have a viscous nature per se and it is good to have a viscous system supporting operations related.

The process modeling is an activity that we limit to the structuring of the workflow in Petri Nets. When operations over the Petri net take place we could imagine different scenarios with viscous or less viscous impact. Adding, removing or modifying a process of the Petri Net might require lines connecting it to other

Chapter 6. Validation

boxes will have to be moved (viscosity problem known as *knock-on*). Then most each line will have to be redrawn individually (problem known as *repetition*).

Allocating a task has been explored in deep. Task allocation patterns have constraints while combined. This has been explored and the tool checks the validity of any attempt to add, change, and remove an allocation pattern. The problem becomes viscous when such changes need to be propagated, *knock-on*, and new jobs satisfies the new allocation model.

The task model is viscous due to its hierarchical structure. Deleting leaf tasks might not produce a lot of work in reorganizing the tree structure but when the deletion is done on a parent node then all the lines connecting it to other nodes will have to be moved, *knock-on*. Then most each line will have to be redrawn individually, *repetition*.

As a whole we consider the *viscosity* of FlowiXML as acceptable as most of the operations it supports are less viscous and those which are viscous are due to the intrinsic nature of their representation (Petri Nets and Task Models) and the operations (organizational changes).

Even that FlowiXML is composed of several notations, mainly, task, process and workflow models. We analyze every notation in each cognitive dimension, if it was something to say about it, each model has just one notation associated.

Also, being related to the software tool the reader might confuse the usability of the software with the usage of the methodology. The methodology can be used independently of the software support.

Type of Activity	Definition	Example	Relatedness to FlowiXML
Incrementation	Adding new items	Adding a new organizational unit to a workflow model; adding a user stereotype to a job.	This type of application clearly fits into the <i>application</i> we have as there the design of the workflow is incremental.
Transcription	Translating items from one domain of application to another.	Copying BPMN workflow models into FlowiXML workflow editor.	FlowiXML offers the possibility to translate some of its representations to other representations. For instance the requirements elicitation table can be translated into a Microsoft Excel table.
Modification	Modifying an item.	Rearranging and changing a process model in	This characteristic is intrinsic to the problem we are dealing with.

Chapter 6. Validation

		the workflow.	The organization is a dynamic entity thus provoking constant changes in the workflow design.
Exploratory Design	Combining incrementation and modification, with the characteristic that the desired end state is not known in advance.	Designing workflow information systems.	The final result of the workflow model is never known in advance, one of the goals of workflow information systems is to enhance and improve business process with the introduction of automation of their process. Then constant refinement is desirable before reaching the, at that moment, the desired final state.

Table 6-4. Type of activity and FlowiXML relatedness.

Chapter 6. Validation

Criteria	Definition	Negative example	Positive Example	Relatedness to FlowiXML	Acceptance for exploratory design
Abstraction, abstraction barrier, abstraction hunger	Is a group of elements to be treated as one entity, either to lower the viscosity or to make the notation more like the user's conceptual structure.	Dialing a telephone number from a list number by number.	Dialing a telephone number from a list avoids the need to dial number by number.	Personalization of the notation is desired but has not been addressed. More meaningful icons should be used for the jobs (doctor, secretary), organizational units (factory, hospital, university), or even for the task model anybody is free to choose the representation of their stylistics. Some predefined or any custom loaded from a file. Also, it might be of interest to provide means to the users to store patterns of their design [Mont08]. For instance the task model of a log in task. Thus saving them time. However, FlowiXML does not allow the creation of custom notations thus this feature is not related to FlowiXML.	Harmful for non experts just when modifications are to be implemented.
	The abstraction barrier is the minimum number of abstractions that must be known before the notation can be used.	All concepts in FlowiXML must be understood before using them.			
	Abstraction- <i>hungry</i> systems can only be used by deploying user-defined abstractions.		In the programming language Smalltalk to start writing a program, you first modify the class hierarchy.		
	Abstraction- <i>tolerant</i> systems permit but do not require user-defined abstractions.	Not to allowed personalized styles in word.	Personalized styles in word processors.		
	Abstraction- <i>hating</i> systems do not allow users to define new abstractions (and typically contain few built-in abstractions).	A spreadsheet.			
Closeness of mapping	Closeness of representation to domain	In FlowiXML, there is no way to know the size of the system you are designing (number of processes, jobs).	Workflows using the Petri Net notation have a common understanding in the workflow community. So as the task modeling notation used in HCI communities.	Relying on a notation that does not change a common understanding of the domain of the problem is important and this is why we always tried to rely on concepts and notations that were known.	Harmful if not addressed
Consistency	Similar semantics are expressed in similar syntactic forms	Operation to manipulate a node in the Petri net is different compared to the manipulation of node in the task model.	Contextualized menus, The while system always have contextualized menus.	Consistency is considered one of the most important aspects of usability [Niel89]. Then this aspect is of relevance for FlowiXML.	Harmful if not addressed

Chapter 6. Validation

Criteria	Definition	Negative example	Positive Example	Relatedness to FlowiXML	Acceptance for exploratory design
Diffuseness	Expresses the verboseness of a notation.	Using task patterns allocation demands to keep in mind their descriptions thus limiting users to implement them easily.	Petri Nets is composed of simple, limited and focused notation.	The notation for workflow modeling must be not to diffuse.	Harmful if not addressed.
Error-proneness	Notation invites to make errors.	The task attributes and their relatedness has shown to be source of errors.	Petri Nets come along with three basic elements, connector, transition, and place. Connections are just possible between a state and a place.	It is relevant to avoid any invitation to commit a mistake due to the notation.	Harmful if not addressed.
Hard mental operations	High demand on cognitive resources.	Resource allocation patterns (40) are hard to identify.	Process attributes are simple and easy to understand. Filling this information does not require high cognitive load.	High cognitive demand will make hard or impossible use of FlowiXML.	Acceptable for small tasks. Harmful if not addressed.
Hidden dependencies	A hidden dependency is a relationship between two components such that one of them is dependent on the other, but that the dependency is not fully visible.	The task model that details the work to be performed in a task of the process model is hidden in the process model view; this information is not fully available, unless the task model editor is launched.	The Petri Net explicitly shows dependencies.	Hidden dependencies slow up information finding, this criterion is of relevance to be evaluated.	Acceptable for small tasks. Harmful for modification particularly at later stages of the modeling.
Premature Commitment	Constraints on the order of doing things force the user to make a decision before the proper information is available.	Design an allocation pattern at design time that constraints the allocation pattern needed at run time.	Selecting tasks, process and workflow from the textual scenario will contribute to build the workflow model. The premature selection must be based on the guidance provided by the identification criteria.	The potential disruption to the process of building the workflow due to the incapacity of the designer to look ahead its design makes this criterion relevant to be considered.	Harmful if not addressed

Chapter 6. Validation

Criteria	Definition	Negative example	Positive Example	Relatedness to FlowiXML	Acceptance for exploratory design
Progressive Evaluation	Means that the work in progress can be easily checked for as far as it is finished.	Task descriptions in the elicitation model cannot be evaluated.	Reachability and completeness of the Petri Net can be checked at any time.	In an exploratory design it is important to have the opportunity to validate the work in progress constantly.	Important to be considered
Provisionality	Degree of commitment to actions or marks.	Task allocation patterns must be selected carefully. Because once selected they block others.	Sketch prototypes of the workflow.	This feature reduces the premature commitment because it is not hardly attached to your early decisions.	Convenient
Role-Expressiveness	The purpose of a component is readily inferred	Task attributes are not easily reached and might be even not all available for manipulation.	Details about the organizational units can be clearly identified and analyzed from a workflow model	Distinguish the different components or a logical block in a notation is a key feature for modeling.	
Secondary notation	Extra information carried by other means than the official syntax.	Imagine a mapping between the workflow model and the workflow model used by the BPMN notation. Any change in the BPMN notation makes incomplete the mapping.	The graphical notation is exported to a XML format that can be processed by a machine for other purposes, such as is the case of the task model for generating User Interfaces.	FlowiXML does consider alternative notation for their different tools, always at least by transforming it into a XML-based format.	Harmful if not addressed
Viscosity	The resistance to change.	Modifying a node that is not a leaf in a task tree might require a lot of rearrangements in the drawing and redoing some connections.	Automatic propagation of changes when the task list is modified.	As the viscosity is a property to be evaluated in systems that encourages reflection and planning, this criterion is of relevance to be evaluated.	Problematic if not addressed. Benefit in critical systems.
Visibility	The ability to view components easily	In a system is a problem when interaction is just possible with one window at the time while the others are disabled.	In FlowiXML it is possible to open different windows of different task models at the same time thus winning visibility.	Visibility of information is key for operations such as: comparison, modification, exploration. This feature is relevant to be evaluated in FlowiXML.	Important to be considered

Table 6-5. CD Criteria and FlowiXML relatedness.

6.4 Chapter summary

This chapter presented the external and internal validation of the method.

In order to conclude this chapter we provide hereafter a set of conclusions issued from the internal validation and from the external validation.

6.4.1 Conclusion from internal validation

As a conclusion to the discussions offered in the internal validation section, Figure 6-39 summarizes the subjective estimation of the extent to which we have addressed the requirements identified in the context of this thesis. It can be noticed that these requirements were covered in a great proportion.

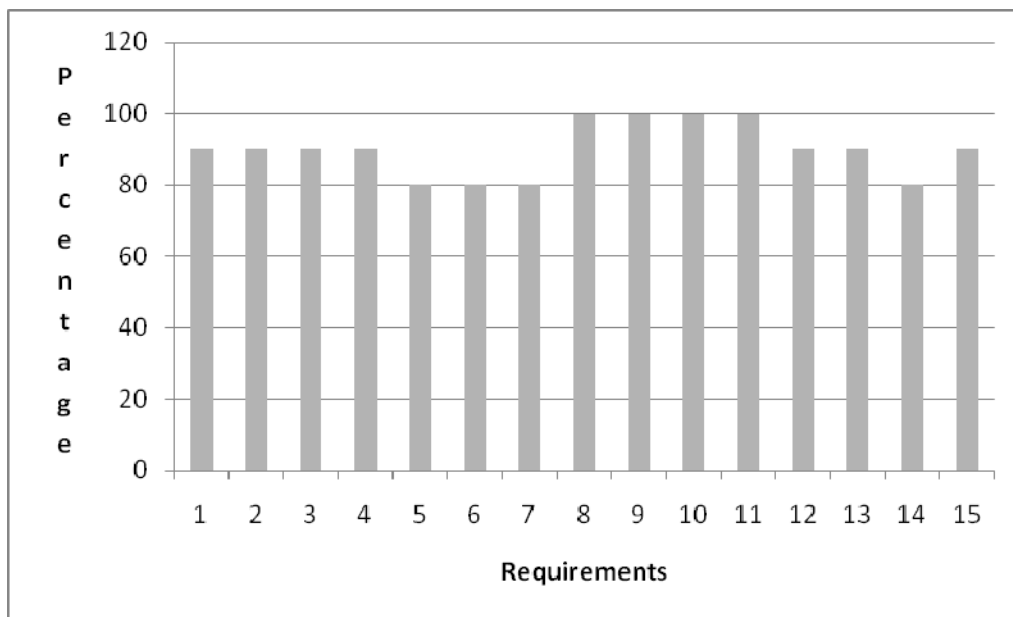


Figure 6-39. Personal subjective requirements coverage rate.

The use of cognitive dimension evaluation of the graphical notation showed to be very valuable to identify problems in the guidance and the software support. Thus, they represent the starting point for future improvements of the whole artifact.

6.4.2 Conclusion from external validation

To validate the method and the software of this dissertation, we used some empirical validation methods. A group of case studies was elaborated applying the method in different disciplines. The diversity of case studies highlights the strengths

Chapter 6. Validation

of our method, as the feasibility to model UIs for workflow information systems, the possibility of manipulating UIs related artifacts according to different development scenarios and pave the way to consider multiple other alternatives, the flexibility in creating UIs according to each user, the benefit that all the design knowledge required to progressively move from a workflow specification to its corresponding UIs is expressed in the model and mapping rules. The method preserves continuity and traceability of its enactment.

Also, we have some weakness of our method, for instance, identification criteria was hard to understand for some users, combination of workflow resource patterns was confused. Referring to the software was difficult to make connections between states and transitions without touch another arc, lack of a view for alternative patterns.

Chapter 6. Validation

Chapter 7 Conclusion

7.1 Context

This chapter summarizes the contribution brought by the current dissertation to the development process of workflow UIs for information systems with respect to the following aspects: (1) theoretical and conceptual contributions related to the definition, usage and validation of some new original concepts pertaining to the problem, (2) methodological contributions concerned with the methodological guidance provided to UI designers in order to manipulate the newly introduced concepts, and (3) software to support the methodological guidance.

By observing the current state of the art in the field of workflow UIs we noticed that most of the development issues tackled one aspect, workflow or UIs development but few tried to address both. The technologies that support them are often dedicated to one purpose, or very complex and resource consuming (e.g., time, processing power).

With respect to these observations, the method proposed in the current thesis is desired not for workflows per se, but for workflow aspects (or concepts) that have some impact on the user interface of a WfIS. The goal here is not to provide yet another meta-model of workflows (there are several others for this purpose), but to provide a meta-model of concepts that are related to UI issues. Then, a method uses the models in coherent way to develop WfIS. A set of software tools are described as a mean to develop the WfIS with respect to the UI. The design knowledge on the basic features a workflow system must have beyond the workflow management but considering UI flow.

The methodology is evaluated with a set of empirical validation and some internal reflections were made to estimate the results obtained with regard to the requirements. Finally, a validation of the notation was conducted using the cognitive dimension method. The results of this analysis help us to improve our methodology and to identify future work.

7.2 Summary of contributions

The contributions of this thesis can be summarized depending on the aspects composing the methodology for developing workflow UIs.

7.2.1 Theoretical and conceptual contributions

- *Expanded task model.* We identified that the existing task model considered for the development of user interfaces could be expanded to better respond to the requirements imposed by workflow model, we expanded the model by adding several classes with attributes along with their values and adding attributes to existing classes.
- *Expanded mapping model.* After reviewing the literature, we noticed that the allocation of tasks is important, which motivated us to expand the model by adding concepts in order to map tasks with task resources.
- *Task life cycle.* The introduction of a task life cycle in this stage was useful to identify task attributes that could represent the different task states, such as *terminationValue*, *preCondition*, and *postCondition*.
- *Mappings between AUI events and their concretization in terms of CUI events used in ECA rules.* We identify that for a same abstract event in a dialog model for AUI, several different mappings can be ensured with concrete events in a dialog model for CUI depending on the context of use, particularly the various interaction modalities, in this case: graphical and vocal. Thus, contributing to UsiXML.
- *Colored concepts.* Colors have been considered in order to make a distinction between the concepts for defining workflow (static classes) and the concepts for executing it (dynamic classes). Thus proved to be useful for the development of the software tool.
- *Stylistics for workflow concepts.* The need of facilitating the understanding and the manipulation of objects employed in software tools required the introduction of a graphical representation associated to some of the introduced workflow concepts.
- *Agenda.* Normally a workflow management system manage the concept of *worklist*, however, we introduce the concept of *agenda* that provides the user the capability to store a personal todo list of tasks and allow the communication with other users.
- *Workflow user interface patterns.* We introduce the concept or Workflow User Interface Patterns (WUIPs) to represent a library of user interface design patterns that are particularly applicable to UIs of workflow information systems. Designers are able now to specify resource allocation patterns.

7.2.2 Methodological contributions

- *Identification criteria.* We have introduced a set of precise criteria that can be used in order to identify a task in a textual scenario and to distinguish a task from other concepts like process and workflow which are located at another level in the hierarchy, but at same level of abstraction.
- *Task life cycle.* In this stage, the task life cycle is useful to illustrate and know the time that a task is created and follow it until to the final completion or failure.
- *User interface flow.* We introduced a formal definition of the flow of UIs as an octuple, and we specified some rules that can be applied to facilitate the modeling of the UIs flow.
- *Methodological steps.* We propose development steps for specifying workflow information systems that ensure a well-formed definition of the workflow concepts.

7.2.3 Software developed

A workflow editor was developed to support the method proposed in this thesis. This editor is divided in the *workflow editor* and the *workflow manager*. The editor is useful to design the process, the task models, to allocate task to user in design time, to specify jobs and user, and all the aspects of the static part of the workflow. The manager is useful to control and manage the workflow in run time, i.e. the dynamic part.

7.3 Future work

Future work could address the following concerns:

- *Sharing UIs.* In order to support group interactions, we want to give the possibility to share agendas. Nowadays, this option is very useful to organize people activities. Also, we want to evaluate the migration of UIs when the tasks are delegated. We consider that maybe the delegation can be done just by sending the part of the UI that is not yet completed; splitting or migrating UIs for this purposes demands a deep study to determine the scenarios in which it can applied.
- *Multi-user interfaces.* Been more ambitious, we can applied the notion of “What You See Is What I See” (WYSIWIS)[Stef86] in order to create multi-user interfaces to better support the cooperative work.
- *Implementing the complete set of workflow resource patterns.* As we pointed early,

workflow resource patterns can be applied at design or at run time; even that we consider all of them, our software just reflects the pattern that can be applied at design time.

- *Workflow analysis.* We would like to include workflow analysis to our approach in order to get statistics of the users of the workflow, and of the cases. We can imagine that we can provide some guides to managers to make decisions referring to the organizational process. For instance, include more employees where there are bottlenecks. Also, collected data for workflow analysis could be used in optimization algorithms to make even better decision about resource allocation including not just human resources.
- *Workflow summary.* To provide a summary of all the components involved in the workflow development (i.e. number of tasks, process, job, user stereotypes, workflow patterns, and so on).
- *Overview of workflow resource patterns.* In order to help in the allocation of task using the appropriate pattern, it will be useful to implement an overview of each pattern in the software and to present alternate patterns to be used or combined.
- *Managing exceptions.* In a workflow system is important to handle abnormal events that may happen during the execution of a task and need to be managed properly.
- *Meta-model refinement and update.* Future work will include further implementation experiences, to allow the refinement and refreshment of our approach.

7.4 Remarks

In this thesis, we have defined the thesis statement based on a set of concerns of workflow and user interfaces for developing UIs for a workflow information systems (Chapter 1). We made a review and analysis of existing literature concerning workflow, tasks model [Guer08b], and user interface description language [Guer09a] (Chapter 2). We concluded with a list of shortcomings of current works. Based on these shortcomings a set of requirements were identified.

We presented a framework for workflow model [Guer08a] and showed how UsiXML concepts have been expanded by directly introducing workflow concepts [Guer06] (Chapter 3). We have introduced a methodology for defining workflow user interface patterns [Guer08f]. To achieve this methodology we have defined the task life cycle and the users involved.

Chapter 7. Conclusion

Further, we have defined a method (Chapter 5) for specifying workflow [Guer08d][Guer08e], besides identification criteria were established to recognize tasks during task modeling [Guer08g], and then the method for developing user interfaces [Guer08c] [Guer08h] was presented.

The software to support our method was introduced (Chapter 6). This software is composed of several modules, among them: the model elicitation tool [Lema08], the task spreadsheet [Guer08g], the resource editor, the workflow editor and manager [Guer08e] [Guer09b].

We have addressed the external and internal validation of the method (Chapter 7), some case studies are considering the framework to future work [Guer09b]. Finally, we presented a summary of contributions and future work in prospect (Chapter 8).

A major benefit of the method is that all the design knowledge required to progressively move from a workflow specification to its corresponding UIs is expressed in the metamodel and the mapping rules. The method preserves continuity (all subsequent models are derived from previous ones) and traceability of its enactment (it is possible to trace how a particular workflow is decomposed into processes and tasks, with their corresponding user interfaces). In this way, it is possible to change any level (workflow, process, task, and UI) and to propagate the changes throughout the other levels by navigating through the mappings established at design time. The strengths of this work are: separation of concerns principle is respected; it bridges the gap between WIS and UI design, the steps of the approach are defined in a comprehensive and logic way for their application. Even more important, this solution is independent of the language and the software tool used.

We want to highlight that part of the theoretical and methodological framework has been considered to be part of the *Deliverable 1.1 SoTA of User Interface Description Languages* of ITEA 2 -UsiXML project; as well as the result of cooperation with two standardization process: the W3C Model Based User Interface [W3C10] and the FP7 European project NexOF-RA [NEXO09].

Finally, last remark regarding the notations used in this work. Some could argue that the used of different notations for each level of details might conduct to some problems in the usage of the methodology due to amount of knowledge needed for doing that. We wanted to keep the common usage of notations in the different fields. Process models are normally modeled with Petri nets or similar building blocks, while task models have the tendency to be model using hierarchical tree structures. In the MDA paradigm this is an accepted way to express a

Chapter 7. Conclusion

task model for UI development. Using exclusively task models for modeling big and complex systems has been reported in [Vand09]. Their lesson learned is that task models were not enough expressive for high level of discussion. The one desired at the organizational level with process and workflows, resource management, etc. The other way around, getting into details from process to task models at first glance looked appropriate. Using the process notation until atomic task are reached and hiding details using the sub-process notion. However, simple operation such as disabling or message passing would not be naturally and easily to identify in the diagram.

References

A

[Abra99]

Abrams, M. Phanouriou, C., Batongbacal, A.L., Williams, S., & Shuster, J. *UIML: An Appliance-Independent XML User Interface Language*. In A. Mendelzon, editor. Proceedings of 8th International World-Wide Web Conference WWW'8 (Toronto, May 11-14, 1999), Amsterdam, 1999. Elsevier Science Publishers. Accessible at <http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html>.

[Andr03]

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., & Weerawarana, S., *Business Process Execution Language for Web Services Specification*, version 1.1 dated May 5, 2003.
<http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf>

[Anne67]

Annett, J., & Duncan, K. *Task Analysis and Training Design*. Occupational Psychology, 41, 211-227. (1967)

[Anne04]

Annett, J. *Hierarchical Task Analysis*. The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates, Mahwah, 2004. pp. 67-82.

[Arga03]

Argamon, S., Koppel, M., Fine, J., & Shimony, A. R.. *Gender, genre, and writing style in formal written texts*. Text, 23(3), 2003.

[Arga05]

Argamon, S., Dhawle, S., Koppel, M., & Pennebaker, J. *Lexical Predictors of Personality Type*. In: 2005 Joint Annual Meeting of the Interface and the Classification Society of North America, 8-12 Jun 2005, St. Louis, MO.

[Azev00]

Azevedo, P., Merrick, P., & Roberts, D. *OVID to AUIML user oriented interface modeling*. In Proceedings of 1st International Workshop Towards a UML Profile for Interactive Systems Development TUPIS00. York, October 2000.

B

[Bast90]

Bastide, R. & Palanque, P. *Petri net objects for the design, validation and prototyping of user-driven interfaces*. Dan Diaper, David J. Gilmore, Gilbert Cockton, Brian Shackel (Eds.): Human-Computer Interaction, INTERACT '90, Proceedings of the IFIP TC13 Third International Conference on Human-Computer Interaction, Cambridge, UK, 27-31 August, 1990. North-Holland 1990, 625-631

References

- [Bast96] Bastide, R. & Palanque, P. (1996). *Implementation Techniques for Petri Net Based Specifications of Human-Computer Dialogues*. In J. Vanderdonckt (Ed.), Proc. of 2nd Workshop on Computer Aided Design of User Interfaces CADUI'96 (Namur, 5-7 June 1996). Namur: Presses Universitaires de Namur.
- [Bast04] Bastide, R., Navarre, D., Palanque, P., Schyn, A., & Dragicevic, P. (2004). *A Model-Based Approach for Real-Time Embedded Multimodal Systems in Military Aircrafts*. In Proc. of 6th ACM International Conference on Multimodal Interfaces ICMI'04 (Pennsylvania State University, USA, October 14-15, 2004). New York: ACM Press.
- [Baum00] Baumeister, L.K., John, B.E., & Byrne, M.D. *A Comparison of Tools for Building GOMS models*. Proceedings of the ACM Conference on Human Factors in Computer Systems (CHI-00) The Hague, The Netherlands, ACM Press. New York, 502:509.
- [Blac00] Blackwell, A.F. & Green, T.R.G. *A Cognitive Dimensions questionnaire optimised for users*. In A.F. Blackwell & E. Bilotta (Eds.) Proceedings of the Twelfth Annual Meeting of the Psychology of Programming Interest Group (PPIG), Cozenza, Italy, April 2000 pp. 137-152.
- [Blac01] Blackwell, A.F., Britton, C., Cox, A. Green, T.R.G., Gurr, C.A., Kadoda, G.F., Kutar, M., Loomes, M., Nehaniv, C.L., Petre, M., Roast, C., Roes, C., Wong, A. & Young, R.M. *Cognitive Dimensions of Notations: Design tools for cognitive technology*. In M. Beynon, C.L. Nehaniv, and K. Dautenhahn (Eds.) Cognitive Technology 2001 (LNAI 2117), Springer-Verlag, pp. 325-341.
- [Bick09] Bickmore, T., Schulman, D., & Shaw, G., *DTask and LiteBody: Open Source, Standards-Based Tools for Building Web-Deployed Embodied Conversational Agents*. In: IVA , Vol. 5773Springer (2009) , p. 425-431.
- [Boda89] Bodart, F. & Pigneur, Y., *Conception assistée des systèmes d'information : méthode, modèles, outils*. Masson, 1989.
- [Boeh88] Boehm, B. *A Spiral Model of Software Development and Enhancement*. IEEE Communications 21, 5 (May 1988): 61-72
- [Bosw02] Boswell, D., King, B., Oeschger, I., Collins, P., & Murphy, E. *Introduction to XUL*. In “Creating Applications with Mozilla”, O'Reilly, Sebastopol, September 2002.

References

- [Brac84]
Bracchi, G., Pernici, B. *The Design Requirements of Office Systems*, ACM Trans. on Office Information Systems, Vol. 2, No. 2, April 1984.
- [Bram07]
Brambilla, M., Comai, S., Fraternali, P., & Matera, M., *Designing web applications with WebML and WebRatio*. In: Rossi, G. et al. (eds.) Web Engineering: Modelling and Implementing Web Applications, Human-Computer Interaction Series. Springer, October (2007)
- [Burre79]
Burrell, G. & Morgan, G. *Sociological Paradigms and Organizational Analysis*, Heinemann: London, 1979.
- C**
- [Calv03]
Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonck, J. *A Unifying Reference Framework for Multi-Target User Interfaces*. Interacting with Computers, vol. 15, no. 3, 289–308. 2003.
- [Card83]
Card, S.K., Moran, T.P. & Newell, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, 1983.
- [Carl97]
Carlsen, S. *Conceptual modeling and composition of flexible workflow models*. PhD-Thesis, Norwegian university of Science and Technology, Norway. 1997.
- [Casa96]
Casati, F. et al. *WIDE Workflow model and architecture*. Technical Report 96-19, Centre for Telematics and Information Technology (CTIT), University of Twente, 1996.
- [CEA07]
CEA-2018 (ANSI). *Task Model Description (CE TASK 1.0)*. CEA. [CEA 2007] http://www.ce.org/Standards/browseByCommittee_4467.asp
- [Cole97]
Coleman, D. *Groupware: Collaborative Strategies for Corporate LANs and Intranets*, Amazon, 1997.
- [Coll08]
Collignon, B., Vanderdonck, J., & Calvary, G., *Model-Driven Engineering of Multi-target Plastic User Interfaces*, Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08), 2008, pp.7-14.
- [Corr97]
Corradini, Ehrig H. , Heckel R. , Korff M. , Löwe M. , Ribeiro L. , & Wagner A., *Algebraic approaches to graph transformation - part I: Single pushout approach and comparison with double pu-*

References

shout approach, in Rozenberg G. (Ed.), Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations, World Scientific, 1997, pp. 247-312.

[Coye05]

Coyette, A. & Vanderdonckt, J., *A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces*. In: Proc. of 10th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'2005 (Rome, 12-16 September 2005). Lecture Notes in Computer Science. Springer-Verlag, Berlin (2005)

[CPNT03]

CPN Tools. *What is CPN Tools?* November 2003. Available on: http://wiki.daimi.au.dk/cpntools/what_is_cpn_tools.wiki?cmd=get&anchor=What+is+CPN+Tools?

[CPNT06]

CPN Tools. *Getting started with CPN Tools*. 2006. Available on: http://wiki.daimi.au.dk/cpntools-help/getting_started_with_cpn_.wiki

[Crow03]

Crowle, S. & Hole, L. *ISML: An interface specification meta-language*, 10th International Workshop on Design, Specification and Verification of Interactive Systems, Madeira. (2003)

D

[DeBo06]

De Boeck, J., Raymaekers, C. Coninx, K., *Comparing NiMMiT and Data-Driven Notations for Describing Multimodal Interaction*, Tamodia 2006 (Diepenbeek, Belgium), October 2006.

[Defi]

Defimedia. www.defimedia.be

[Deml03]

Demler, G., Wasmund, M., Grassel, G., Priestestersbach, A. & Ziegert, T. *Flexible pagination and layouting for device independent authoring*, WWW2003 Emerging Applications for Wireless and Mobile access Workshop. (2003)

[Dena]

Denali. www.denali.be

[DSou99]

D'Souza, D.F. & Wills, A.C. *Objects, Components and Frameworks with UML: The Catalysis Approach*. Addison-Wesley, Reading, 1999.

[Duma01]

Dumas, M. and ter Hofstede, A. *UML Activity Diagrams as a Workflow Specification Language*. In M. Gogolla and C. Kobryn, editors, Fourth International Conference on the Unified Modeling Language (UML 2001), pp. 76-90, Toronto, Canada, 2001

References

E

- [Ehri99] Ehrig, H., Engels, G., Kreowski, H.-J., & Rozenberg, G. (eds.), *Handbook of Graph Grammars and Computing by Graph Transformation, Application, Languages and Tools*, Vol. 2, World Scientific, Singapore, 1999.
- [Eich04] Eichholz, C., Dittmar, A., & Forbrig, P. (2004) *Using Task Modeling Concepts for Achieving Adaptive Workflows*. Proceedings of DSV-IS-EHCT'94, Springer-Verlag, Berlin, LNCS,3425.
- [Eins98] Eisenstein, J. & Puerta, A.R. *Towards a general computational framework for model-based interface development systems*. In Proceedings of the 4th International Conference on Intelligent User Interfaces, pp. 171-178. 1998.
- [Eise00] Eisenstein J., Vanderdonckt J., & Puerta A., *Adapting to Mobile Contexts with User-Interface Modeling*, Proceedings of 3rd IEEE Workshop on Mobile Computing Systems and Applications WMCSA'2000 (Monterey, 7-8 December 2000), IEEE Press, Los Alamitos, 2000, pp. 83-92.
- [Eise01] Eisenstein J., Vanderdonckt J., & Puerta A., *Model-Based User-Interface Development Techniques for Mobile Computing*, Proceedings of 5th ACM Int. Conf. on Intelligent User Interfaces IUI'2001 (Santa Fe, 14-17 January 2001), Lester, J. (Ed.), ACM Press, New York, 2001, pp. 69-76.
- [Erik03] Erik Andriessen, J.H. *Working with Groupware. Understanding and Evaluating Collaboration Technology*. Springer, 2003.
- [Espo05] Esposito, D. *Getting Started with Microsoft Windows Workflow Foundation: A Developer Walk-through*. September 2005. <http://msdn.microsoft.com/winfx/reference/workflow>

F

- [Fitz98] Fitzgerald, B., & Howcroft, D. *Competing dichotomies in IS research and possible strategies for revolution*. In Internal Conference on Information Systems, Helsinki. 1998.
- [Fokk92] Fokkinga M.M., *A gentle introduction to category theory: the calculational approach*, in Lecture Notes of the STOP 1992 Summerschool on Constructive Algorithmics, University of Utrecht, September 1992, pp. 1-72.
- [Fran93] Frank, M. & Foley, J. *Model-based user interface design by example and by answering questions*. In proceedings of INTERCHI, ACM Conference on Human Factors in Computing Systems, p.p. 161-162; April 1993.

References

G

[Gali97]

Galitz, W.O., *The Essential Guide to User Interface Design. An introduction to GUI design principles and techniques*. Wiley Computer Publishing, 199. ISBN 0-471-15755-4

[Garl01]

Garland, A., Ryall, K., & Rich, Ch., *Learning hierarchical task models by defining and refining examples*. In: Proc. of the 1st Int. Conf. on Knowledge Capture (Victoria, 2001). ACM Press, New York (2001) 44–51.

[Garr03]

Garrido Bullejos, J.L. *AMENIT IES: Una metodología para el desarrollo de sistemas cooperativos basada en modelos de comportamiento y tareas*. Ph.D thesis. Granada, España, 2003.

[Ghez91]

Ghezzi, C., Jazayeri, M., & Mandrioli, D. (1991). *Fundamentals of Software Engineering*. Englewood Cliffs (United States): Prentice Hall.

[Gies08]

Giese, M., Mistrzyk, T., Pfau, A., Szwillus, G. and von Detten, M. *AMBOSS: A Task Modelling Approach for Safety-Critical Systems*. Proc. of 7th Int. Workshop on Task Models and Diagrams TAMODIA'2008 (Pisa, 25-26 September 2008), Lecture Notes in Computer Science, Springer-Verlag, Berlin, to appear. Available at: http://wwwcs.uni-paderborn.de/cs/ag-szwillus/lehre/ws05_06/PG/PGAMBOSS/index.php

[Glan96]

Glance, N. S.; Pagani, D. S.; Pareschi, R. *Generalized Process Structure Grammars (GPSG) for flexible representations of work*. In: M. Ackerman (Ed.): CSCW'96: Proceedings of the Conference on Computer Supported Cooperative Work. Boston, MA, 1996, pp. 180-189.

[Gome04]

Gomes de Sousa, L., & Leite, J.C. *XICL- An Extensible Mark-up Language for Developing User Interface and Components*. Proceedings of the Fifth International Conference on Computer-Aided Design of User Interface CADUI'2004.

[Gonz09]

Gonzalez, J.M., Guerrero, J., Muñoz, J., Vanderdonckt, J., & Martinez, J.F., *A method for generating multiplatform user interfaces for e-learning environments*. Chapter 6, in Tiong T. Goh (ed.), "Multiplatform E-Learning Systems and Technologies", Information Science Reference, IGI Global Inc., Hershey, 2009, pp. 90-111

[Gree89]

Green, T.: *Cognitive dimensions of notations*. In: People and Computers, Cambridge University Press, Cambridge, UK (1989) 443–460.

[Gree98]

Green, T., Blackwell, A.: *Cognitive dimensions of information artefacts: a tutorial*. Available on: <http://www.ndirect.co.uk/thomas.green/workstuff/papers/> (2005)

References

- [Greg02]
Gregor, S. *A theory of theories in information systems*. In S. Gregor and D. Hart (Eds.), *Information Systems Foundations: Building the Theoretical Base*. Australian National University, Canberra, 1-20. 2002.
- [Grud88]
Grudin, J. *Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces*. 1988. Proc. CSCW 88, 85-93.
- [Grud94]
Grudin, J. CSCW: History and focus. *IEEE Computer*, 27, 5, 1994, 19-26
- [Guer06]
Guerrero Garcia, J. *Conceptual Modeling of User Interfaces to Workflow Information Systems*, DEA thesis, Université Catholique de Louvain, Belgique. September 2006
- [Guer08a]
Guerrero, J., Vanderdonckt, J., Gonzalez Calleros, J.M., *FlowiXML: a Step towards Designing Workflow Management Systems*, *Journal of Web Engineering*, Vol. 4, No. 2, 2008, pp. 163-182.
- [Guer08b]
Guerrero, J., Vanderdonckt, J., Gonzalez Calleros, J.M., *Towards a Multi-User Interaction Meta-Model*. Working paper 08/28 ULC, Louvain School of Management. December 2008.
- [Guer08c]
Guerrero, J., Vanderdonckt, J., Gonzalez, J.M., Winckler, M. *Modeling User Interfaces to Workflow Information Systems*, Proc. of 4th International Conference on Autonomic and Autonomous Systems ICAS'2008 (Gosier, 16-21 March 2008), IEEE Computer Society Press, Los Alamitos, 2008.
- [Guer08d]
Guerrero, J., Vanderdonckt, J., Lemaigre, Ch., González Calleros, J.M., *How to Describe Workflow Information Systems to Support Business Process*, Proc. of 10th IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services CEC'2008 (Washington D.C., July 21, 2008), IEEE Computer Society Press, Los Alamitos, 2008, pp. 404-411
- [Guer08e]
Guerrero García, J., Lemaigre, Ch., Vanderdonckt, J., González Calleros, J.M., *Model-Driven Engineering of Workflow User Interfaces*, Proc. of 7th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2008 (Albacete, 11-13 June 2008), Springer, Berlin, 2008
- [Guer08f]
Guerrero, J., Vanderdonckt, J., González Calleros, J.M., Winckler, M., *Towards a Library of Workflow User Interface Patterns*, Proc. of 15th Int. Workshop on Design, Specification, and

References

- Verification of Interactive Systems DSV-IS'2008 (Kingston, July 16-18, 2008), Lecture Notes in Computer Sciences, Vol. 5136, Springer, Berlin, 2008, pp. 96-101
- [Guer08g] Guerrero, J., Vanderdonckt, J., Lemaigre, Ch., *Identification Criteria in Task Modeling*, Proc. of 1st IFIP TC 13 Human-Computer Interaction Symposium HCIS'2008 (Milan, 8-9 September 2008), P. Forbrig, F. Paterno, A.M. Pejtersen (eds.), International Federation of Information Processing, Vol. 272, Springer, Boston, 2008, pp. 7-20.
- [Guer08h] Guerrero, J., Lemaigre, Ch., Gonzalez Calleros, J.M., Vanderdonckt, J., *Towards a Model-Based User Interface Development for Workflow Information Systems*, International Journal of Universal Computer Science, Vol. 14, No. 19, 2008, pp. 3236-3249
- [Guer09a] Guerrero-García, J., González-Calleros, J.M., Vanderdonckt, J., Muñoz-Arteaga, J., *A Theoretical Survey of User Interface Description Languages: Preliminary Results*, Extended Proc. of 4th Latin American Conference on Human-Computer Interaction CLIHC'2009 (Merida, 9-11 November 2009), A. L. Morán, E. Chávez, E. S. Furtado, M. E. Tentori, M. D. Rodríguez (Eds.), Universidad Autónoma de Baja California, Ensenada, 2009, pp. 8-15.
- [Guer09b] Guerrero Garcia, J., Vanderdonckt, J., González Calleros, J.M., *Developing user interfaces for community-oriented workflow information systems*, Chapter 16, in D. Akoumianakis (ed.), "Virtual Communities of Practice and Social Interactive Technologies: Lifecycle and Workflow Analysis", IGI Global Inc., Hershey, 2009, pp. 307-329
- ## H
- [Hart90] Hartson, H. R., Siochi, A. C et al.: *The UAN - a User-Oriented Representation for Direct Manipulation Interface Designs*. Acm Transactions on Information Systems 8(3) (1990) 181-203.
- [Hatc06] Hatch, M.J., *Organization Theory: Modern, symbolic, and postmodern perspectives*. 2nd Ed. Oxford University Press (2006) ISBN 0-19-926021-4.
- [Heck95] Heckel R. & Wagner A., *Ensuring consistency of conditional graph grammars - a constructive approach*, in Lecture Notes in Theoretical Computer Science, Springer Verlag, 1995.
- [Helm08] Helms, J. Schaefer, R, Luyten, K., Vanderdonckt, J., Vermeulen, J. & Abrams, M., UIML Version 4.0: Committee Draft (2008), Available at <http://www.oasis-open.org/committees/download.php/28457/uiml-4.0-cd01.pdf>.

References

[Hevn04]

Hevner, A.R., March, S.T., Park, J., & Ram, S. *Design Science in Information Systems Research*, MIS Quarterly, 28(1), 2004, pp. 75-105

[Hevn07]

Hevner, A.R., A Three Cycle View of Design Science Research, Scandinavian Journal of Information Systems, 19(2), 2007, pp. 87-92

[Hirs89]

Hirschheim, R. & Klein, H. *Four Paradigms of Information Systems Development*. Communication of the ACM, vol. 32, No. 10. October, 1989.

I

[IBM02]

IBM WSXL specification, April 2002.

<http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-wsxl/ws-wsxl2.pdf>

[IBM06]

IBM WebSphere® MQ Workflow, 2006.

<http://www-306.ibm.com/software/integration/wmqwf/>

[ITEA10]

Deliverable 1.1 SoI of User Interface Description Languages of ITEA 2 -UsiXML project (<http://itea.defimedia.be>).

J

[Jens97]

Jensen, K. *Colored Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Volume 1, Basic Concepts. Monographs in Theoretical Computer Science, Springer-Verlag, 2nd corrected printing 1997.

[John89]

Johnson, P. and Johnson, H., *Knowledge analysis of task: Task analysis and specification for human-computer systems*. In A. Downton, (Ed.). Engineering the human-computer interface (pp. 119-144), London: McGraw-Hill (1989).

[John91]

Johnson, P., & Johnson, H.: *Task Knowledge Structures: Psychological basis and integration into system design*. Acta Psychologica 78, pp 3-26, 1991

[John01]

Johnson M., & Dampney, *On category theory as a (meta) ontology for information systems*, Proceedings of the international conference on Formal Ontology in Information Systems. FOIS 01 (Ogunquit, Maine, USA), 2001, pp. 59-69.

References

[Jure06]

Jureta, I., Faulkner, S., Kolp, M., Formalizing Agent-Oriented Enterprise Models. In: M. Kopl et al. (Eds): AOIS 2005, LNAI 3529, pp. 184-199, 2006. Springer-Verlag Berlin Heidelberg 2006.

K

[Karl94]

Karlgren, J. *Stylistic Experiments for Information Retrieval*. PhD thesis, SICS, 2000.

[Kats03]

Katsurada, K., Nakamura, Y., Yamada, H. & Nitta, T. *XISL: A Language for Describing Multimodal Interaction Scenarios*. In Proc. of 5th Int. Conf. on Multimodal Interfaces ICMI'2003 (Vancouver, 5-7 November 2003), ACM Press, New York, pages 281–284, 2003.

[Kelle92]

Keller, G., Nuttgens, M., & Scheer, A.W., *Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK)*. Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

[Khos95]

Khoshafian, S., & Buckiewicz, M. *Introduction to Groupware, Workflow, and Workgroup computing*. John Wiley & Sons, Inc. ISBN 0 471 02946 7, 1995.

[Kier94]

Kieras, D. (1994) *A guide to GOMS task analysis*.

[Kier95]

Kieras, D., Wood, S.D., Abotel, K., & Hornof, A. *GLEAN: A Computer-Based Tool for Rapid GOMS Model Usability Evaluation of User Interface*. Proceedings of the ACM Symposium on User Interface Software and Technology, 1995. pp.91- 100.

[Kier99]

Kieras, D. *A Guide to GOMS Model Usability Evaluation using GOMS_L and GLEAN₃*. Jan-12, 1999. University of Michigan.

<http://www.eecs.umich.edu/~kieras/goms.html>

[Klein85]

Klein, H. & Hirschheim, R. *Fundamental issues of decision support systems: A consequentialist perspective*. Decis. Supp. Syst. 2,1. 5-23. 1985.

[Kost04]

Kost, S., *Dynamically generated multi-modal application interfaces*. PhD. Thesis at Technical University of Dresden and Leipzig University of Applied Sciences. Germany, 2004.

[Kris07]

Kristiansen, R. & Trøttemberg, H. *Model-Based User Interface Design in the Context of Workflow Models*. In Proc. of Tamodia'2007. Springer, Berlin (2007) 227–239

L

[Laud06]

Laudon, K., & Laudon, J. (2006). *Management information systems: managing the digital firm*. Upper Saddle River (United States): Pearson Prentice Hall.

References

[Lema07]

Lemaigre, C. *Développement d'un éditeur graphique de workflow générant automatiquement ses spécifications fonctionnelle*. Bachelor-Thesis, Université Catholique de Louvain, Belgique. 2007.

[Lema08]

Lemaigre, Ch., Guerrero, J., Vanderdonckt, J., *Interface Model Elicitation from Textual Scenarios*, Proc. of 1st IFIP TC 13 Human-Computer Interaction Symposium HCIS'2008 (Milan, 8-9 September 2008), P. Forbrig, F. Paterno, A.M. Pejtersen (eds.), International Federation of Information Processing, Vol. 272, Springer, Boston, 2008, pp. 53-66.

[Leym94]

Leymann, F.; Altenhuber, W. *Managing Business Processes as an Information Resource*. IBM Systems Journal, 33 (1994) 2, pp. 326-348.

[Limb04a]

Limbourg, Q. *Multi-Path Development of User Interfaces*. PhD-Thesis, Université Catholique de Louvain, Belgique. 2004.

[Limb04b]

Limbourg, Q. & Vanderdonckt, J., *Comparing Task Models for User Interface Design*, The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates, Mahwah, 2004. pp. 135-154.

[Limb04c]

Limbourg, Q. & Vanderdonckt, J., *Addressing the Mapping Problem in User Interface Design with UsiXML*, Proc. of 3rd Int. Workshop on Task Models and Diagrams for user interface design TAMODIA'2004, 2004, pp. 155-163.

[Lu02]

Lu, S., Paris, C., & Vander Linden, K. *Tamot : Towards a Flexible Task Modeling Tool*. Proceedings of Human Factors 2002. Melbourne, Australia. November, 2002.

[Luyt04]

Luyten, K., Abrams, M., Vanderdonckt, J. & Limbourg, Q. *Developing User Interfaces with XML: Advances on User Interface Description Languages*, Satellite workshop of Advanced Visual Interfaces 2004, Gallipoli, Italy (2004).

M

[Mahf97]

Mahfoudhi, A., TOOD: Une methodologie de description orientee objet des taches utilisateur pour la specification et la conception des interfaces hommes-machines. Ph.D. thesis, Univ. of Valenciennes, 1997.

[Maho04]

Mahoney, M. J. *What is constructivism and why is growing?*. Contemporary Psychology, 49, 360-363. 2004. Available on line: http://www.constructivism123.com/What_Is/What_is_constructivism.htm.

References

- [Mand94]
Mandviwalla, M., & Olfman, L. *What do groups need? A proposed set of generic groupware requirements*, ACM Transactions on Computer-Human Interaction, Vol. 1, No. 3, pp. 245 – 268, 1994.
- [Mano02]
Dragos A. Manolescu., *An Extensible Workflow Architecture with Objects and Patterns*. TOOLSEE 2001, March 2002, Sofia, Bulgaria.
- [Marc95]
March, S.T. & Smith, G.F., *Design and Natural Science Research on Information Technology*, Decision Support Systems, 15, 1995, pp. 251-266
- [Mars94]
Marshak, R.T., *Workflow White Paper – An overview of Workflow Software*, Workflow'94, San Jose, 1994.
- [Mars97]
Marshak, R.T. *Workflow: Applying Automation to Group Process*. In Coleman, D. (ed.): *Groupware-Collaborative Strategies for Corporate LANs and Intranets*. Prentice Hall PTR, 1997, p.p. 143-181.
- [Marq97]
Marquis J.-P., *Stanford encyclopedia of philosophy: Category theory*, 1997. Available online: <http://plato.stanford.edu/entries/category-theory/>.
- [Medi92]
Medina-Mora, R.; Winograd, T.; Flores, R.; Flores, F. *The Action Workflow Approach to Workflow Management Technology*. In: CSCW'92: Proceedings of the Conference on Computer Supported Cooperative Work, ACM Publishers, New York, pp. 281-288.
- [Mens99]
Mens T., *A Formal Foundation for Object-Oriented Software Evolution*, PhD thesis, Vrije Universiteit Brussel, 1999.
- [Mey90]
Meyer, B. *Introduction to the theory of programming languages*. Prentice Hall, New York, 1990.
- [Mich08]
Michotte, B. & Vanderdonckt, J., *GrafiXML, A Multi-Target User Interface Builder based on UsiXML*, Proc. of 4th International Conference on Autonomic and Autonomous Systems ICAS'2008 (Gosier, 16-21 March 2008), IEEE Computer Society Press, Los Alamitos, 2008.
- [Mint82]
Mintzberg, H. *Structure & Dynamique des Organisations*. Les Editions D'Organization, Paris, 1982.

References

- [Moli05]
Molina Masso, J.P., Vanderdonckt, J., & González López, P. *Towards Virtualization of User Interfaces based on UsiXML*, 10th International Conference on 3D Web Technology, University of Wales, Bangor, UK, March 29 – April 1, 2005.
- [Moli06]
Molina, A.I. *Una Propuesta Metodológica para el Desarrollo de la Interfaz de Usuario en Sistemas Groupware*. Tesis Doctoral. Universidad de Castilla La Mancha, 2006.
- [Mont70]
Montanari, U.G., *Separable Graphs, planar Graphs and Web Grammars*, in Inf. Contr., 16, 1970, pp. 243-267.
- [Mont05]
Montero, F., López-Jaquero, V., Vanderdonckt, J., Gonzalez, P., & Lozano, M.D.. *Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML*. In Proc. of 12th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSVIS'2005 (Newcastle upon Tyne, July 13–15, 2005), Springer-Verlag, Berlin, 2005, to appear.
- [Mont06]
Montero, F., López-Jaquero, V., *IdealXML: An Interaction Design Tool-A Task-Based Approach to User Interfaces Design*, Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006 (Bucharest, 6-8 June 2006), Chapter 20, Springer-Verlag, Berlin, 2006, pp. 245-252.
- [Mont08]
Montero Simarro, F., Vanderdonckt, J., *Generative pattern-based design of user interfaces*. LSM Working papers 08/13, April 2008. Available online at: http://www.uclouvain.be/cps/ucl/doc/iag/documents/WP_08-13.pdf
- [Mori02]
Mori, G., Paternò, F. & Santoro, C. *CTTE: Support for Developing and Analyzing Task Models for Interactive Systems Design*. IEEE Transactions on Software Engineering, September 2002.
- [Mura89]
Murata, T. *Petri Nets: Properties, Analysis and Applications*. Proceeding of the IEEE, vol. 77, No. 4, April 1989.
- [Myer92]
Myers, B., & Rosson, M. "Survey on User Interface Programming". Proceedings CHI'92, New York, ACM, 1992, pág. 195-202.
- [Myer00]
Myers, B., Hudson, S., & Pausch, R. (2000). *Past, present, future of user interface tools*. ACM Transactions on Computer-Human Interaction, 7(1), 3-28.

References

N

[Nava09]

Navarre, D., Palanque, P., & Winckler, M., Task Models and System Models as a Bridge between HCI and Software Engineering. Human-Centered Software Engineering: Software Engineering Models, Patterns and Architectures for HCI. Springer London, 2009, pp. 357-385. ISBN 978-1-84800-906-6.

[Nexo09]

NEXOF-RA Investigation Team on Declarative UI Authoring and Context Models. Set of Reference Models for Interactive Applications, 2009. Available on-line at: http://forge.morfeo-project.org/wiki_en/index.php/Task_Model

[Niel89]

Nielsen, J., *Coordinating User Interfaces for Consistency*. (Ed.) Academic Press Professional, Inc., San Diego, CA, 1989.

[Niga95]

Nigay, L. & Coutaz, J. *A generic platform for addressing the multimodal challenge*. In proceedings of CHI'95, AC: Press, New York 1995, p.p. 98-105.

O

[OMG06]

Object Management Group: *Business process modeling notation specification, final adopted specification* dtc/06-02-02 (2006).

[OMG07]

OMG. *Software Process Engineering Metamodel Specification, v2.0*. February 2007.

P

[Padr02]

Padrón, C. *Interpretaciones históricas acerca del conocimiento científico*. In Idea Sapiens. Barcelona. 2002.
<http://www.ideasapiens.com/filosofia.sxx/fciencia/interpretacionesconoc.cientifico.htm>

[Pala99]

Palanque, P., Farenc, Ch., & Bastide, R. *Embedding Ergonomic Rules as Generic Requirements in a Formal Development Process of Interactive Software*. In Proceedings of IFIP TC 13 Conference on Human-Computer Interaction INTERACT'99 (Edinburg, Scotland, 1-4 September 1999)

[Pari04]

Paris, C., Lu, S. & Vander Linden, K. *Environments for the Construction and Use of Task Models*. The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates, Mahwah, 2004. pp. 467-482.

References

- [Pate97] Paternò, F., Mancini, C., & Meniconi, S. (1997) *ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models*. (Proceedings Interact'97), Chapman&Hall, pp. 362-369.
- [Pate99] Paternò, F. *Model-based design and evaluation of interactive applications*. Applied Computing, Springer. 1999.
- [Pate00] Paternò, F. (2000) *Model-based design of interactive applications*. Springer Verlag, Berlin.
- [Pate02] Paternò, F. & Santoro, C. One Model, Many Interfaces. Proceedings Fourth International Conference en Computer-Aided Design of User Interfaces, pp. 143-154, Kluwer Academics Publishers, Valenciennes, May 2002.
- [Pate03] Paternò, F. & Santoro, C., *A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms*, Interacting with Computers, Elsevier, 15, pp. 349-366. (2003)
- [Peni03] Penichet, V.M.R. *Sistemas Groupware: Metodos y Herramientas*. Estudio de Casos, UCLM, Albacete, 2003.
- [Peni07] Penichet, V.M.R. *Modelo de Proceso para el Desarrollo de Interfaces en Entornos CSCW Centrado en los Usuarios y Dirigido por Tareas*. Ph.D. Thesis. Universidad de Castilla-la Mancha, España, 2007.
- [Peti94] Petitjean T. (1994) *Contribution a la specification de situations de cooperation ad hoc et a leur prise en compte dans les systèmes de Workflow*. Ph.D. thesis, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium.
- [Peti02] Petit, M., *Some Methodological Clues for Defining a Unified Enterprise Modelling Language*, Proceedings of the IFIP TC5/WG5.12 International Conference on Enterprise Integration and Modeling Technique: Enterprise Inter- and Intra-Organizational Integration: Building International Consensus, p.359-369, April 24-26, 2002
- [Pica03] Picard, E., Fierstone, J., Pinna-Dery, A-M. & M. Riveill. *Atelier de composition d'ihm et évaluation du modèle de composants*. Livrable I3, RNTL ASPECT, Laboratoire I3S, mai. (2003).
- [Puer96] Puerta, A. R. (1996). *The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development*. Proceedings of the 2nd International Workshop on Computer-Aided Design of User Interfaces CADUI'96 (5-7 June, Namur).
- [Puer97] Puerta, A. R.. *A model-based interface development environment*. IEEE Software, 14(4), 40-47. 1997.
- [Puer99] Puerta, A. & Eisensteint, J., Towards a General Computational Framework for Model-

References

Based Interface Development Systems Model-Based Interfaces. In proceedings of 3rd International ACM Conference on Intelligent User Interfaces IUI'99 (Redondo Beach, 5-8 January 1999), ACM Press, New York, 1999, pp. 171-178.

[Puer02]

Puerta, A., & Eisenstein, J. *XIML: A common representation for interaction data*. In Proceedings of the 7th International Conference on Intelligent User Interfaces, pp. 69-76. ACM Press, January 2002.

R

[Raut00]

Rauterberg, G.W.M., *How to characterize a research line for user-system interaction*. IPO Annual Progress Report 35. 2000.

[Rich09]

Rich, C. (2009). *Building task-based user interfaces with ANSI/CEA-2018*. Computer, 42(8):20–27.

[Rieh96]

Riehle, D. & Züllighoven, H. *Understanding and Using Patterns in Software Development*. Theory and Practice of Object Systems, 2(1):3-13, 1996.

[Rock00]

Rockwell Software. “*Arena Basic Edition user’s Guide*”. Rockwell Software Inc. 2000.

[Rose06]

Rosemann, M., Recker, J., Indulska, M., & Green, P., *A study of the evolution of the representational capabilities of process modeling grammars*. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 447–461. Springer, Heidelberg (2006)

[Russ04]

N. Russell, A.H.M. ter Hofstede, D. Edmond, & W.M.P. van der Aalst. *Workflow Data Patterns*. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004.

[Russ05]

Russell N., van der Aalst, W.M.P., ter Hofstede, A.H.M. & Edmond, D. (2005) *Workflow Resource Patterns*. In the 17th Conference on Advanced Information Systems Engineering (CAISE'05). Porto, Portugal. 13-17 June.

S

[Scha06]

Schaefer, R., Steffen, B., & Wolfgang, M., *Dialog modeling for Multiple Devices and Multiple Interaction Modalities*. Task Models and Diagrams for User Interface Design, Proceedings of 5th International Workshop, TAMODIA'2006 (Hasselt, Belgium, October 2006), Lecture Notes in Computer Science, Vol. 4385, Springer Verlag Berlin, 2006, pp. 39-53.

References

- [SCXML05]
State Chart XML (SCXML): *State Machine Notation for Control Abstraction 1.0*. W3C Working Draft 5 July 2005. <http://www.w3.org/TR/2005/WD-scxml-20050705/>
- [Sioc90]
Siochi, A. C., Hartson, H. R., & Hrx, D. *Notational techniques for accommodating user intention shifts*. TR 90-18, Dept. of Computer Science, Virginia Polytechnic Institute and State Univ., 1990.
- [Slon95]
Slonneger, K., Kurtz, B. *Formal Syntax and Semantics of Programming Languages: A Laboratory Based Approach*. Addison-Wesley Pub Co, 1995.
- [Souc03]
Souchon, N. & Vanderdonckt, J. *A review of XML-compliant user interface description languages*. DSV-IS2003, 2003.
- [Sowa92]
Sowa J. F., *Conceptual Graphs Summary*, in Eklund P., Nagle T., Nagle J., and Gerholz L. (Eds.), *Conceptual Structures: Current Research and Practice*, Ellis Horwood, 1992, pp. 3-52.
- [Stam00]
Stamatatos, E., Fakotakis, N., & Kokkinakis, G.K. *Automatic text categorization in terms of genre, author*. Computational Linguistics, 26(4):471–495, 2000.
- [Stan08]
Stanciulescu, A. *A Methodology for Developing Multimodal User Interfaces of Information Systems*, Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, 25 June 2008.
- [Stav04]
Stavness, N. & Schneider, K. *Supporting Flexible Business Processes with a Progression Model*. Workshop: Making Model-based UI Design Practical: Usable and Open Methods and Tool, 2004.
- [Stav05]
Stavness, N. *Supporting Flexible Workflow Process with a Progression Model*. Thesis. University of Saskatchewan, Saskatoon. 2005.
- [Stef86]
Stefik, M., Bobrow, D.G., Lanning, S., Tratar, D. *WYSIWIS Revised: Early Experiences with Multi-user Interfaces*. In proceedings of ACM Conference on Computer-Supported Cooperative Work CSCW'86. (December 3-5, 1986, Austin, Texas, USA) ACM Press, New-York, pp. 276-290.
- [Subr09]
Subramanian, S., & Sindre, G., *An Optimization Rule for ActiveXML Workflows*. In M. Gaedke, M. Grossniklaus, and O. Diaz (Eds.): ICWE 2009, LNCS 5648, pp. 410–418. Springer-Verlag Berlin Heidelberg 2009

References

[Szek96]

Szekely, P. (1996). *Retrospective and Challenges for Model-Based Interface Development*. Proceedings of the 2nd International Workshop on Computer-Aided Design of User Interfaces CADUI '96 (5-7 June, Namur).

T

[Taba00]

Tabary, D. & Abed, M. *A Software Tool Task Object-Oriented Design (TOOD) Method*. Proceedings ACIDCA'2000, March 22– 24, 2000, Monastir, Tunisia.

[Taba02]

Tabary D. & Abed M., *A software environment task object-oriented design (ETOOD)* Journal of Systems and Software, 60 (2), pp. 129-140. 2002.

[Tarb96]

Tarby, J-C. & Barthet, M-F. *The DLANE+ method*. In Computer-Aided Design of User Interfaces, Proceedings of the Second International Workshop on Computer-Aided Design of User Interfaces (CADUI'96). Namur, Belgium. J. Vanderdonckt (Ed.), Presses Universitaires de Namur, Namur. 1996. pp. 95-119.

[Teor86]

Teory, T.J., Yang, O. & Fry, J.P. *A logical Design Methodology for Relation Databases Using the Extended Entity-Relationship Model*. ACM Computing Surveys 18(2), June 1986, p.p. 197-222.

[Thie03]

Thiétart, R.A. et al. *Méthodes de Recherche en Management*. Dunod, Paris. 2003.

[Trae99]

Traetteberg, H. *Modeling work: Workflow and Task modeling*. In: Vanderdonckt, J., Puerta, A.R.(eds.): Proc. of 3rd Int. Conf. on Computer-Aided Design of User Interfaces CADUI'99 (Louvain-la-Neuve, 21-23 October 1999). Kluwer Academics. Dordrecht (1999) 275-280.

U

[Usch98]

Uschold, M., King, M., Moralee, S., & Zorgios, Y. *The Enterprise Ontology*. The Knowledge Engineering Review, Vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate), 1998.

[USIX05]

UsiXML Consortium, *UsiXML, a General Purpose XML Compliant User Interface Description Language*, UsiXML V1.6.3, 16 June 2005. Available online: <http://www.usixml.org>.

References

- [USIX06]
UsiXML Consortium. UsiXML, a *General Purpose XML Compliant User Interface Description Language*, UsiXML V1.6.4, 1 March 2006. Available on line: <http://www.usixml.org/index.php?view=page&idpage=6>
- [USIX07]
UsiXML Consortium, *UsiXML, a General Purpose XML Compliant User Interface Description Language*, UsiXML V1.8, 14 February 2007. Available online: <http://www.usixml.org>.
- V**
- [Vand09]
Van den Bergh, J., Sahni, D., & Coninx, K., *Task models for safe software evolution and adaptation*. In Proc. of HESSD & TAMODIA 2009, Lecture Notes in Computer Science LNCS series, 5963, pp. 72 - 77.
- [vand98]
van der Aalst, W.M.P. *The Application of Petri Nets to Workflow Management*. The Journal of Circuits, Systems and Computers, 8(1):21--66, 1998.
- [vand00]
van der Aalst, W.M.P. & Kumar, A. *XML Based Schema Definition for Support of Inter-organizational Workflow*. Meeting on XML/SGML based Interchange Formats for Petri Nets, 21st International Conference on Application and Theory of Petri Nets [ICATPN 2000]. Aarhus, Denmark, June 26-30, 2000.
- [vand01]
van der Aalst, W.M.P. & Kumar, A. *XML Based Schema Definition for Support of Inter-organizational Workflow*. University of Colorado and University of Eindhoven. Technical Report, 2001
- [vand02]
van der Aalst, W. & van Hee, K., *Workflow Management: Models, Methods, and Systems*. THE MIT Press, Cambridge. 2002.
- [vand03]
van der Aalst W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. & Barros, A.P., *Workflow Patterns*. Distributed and Parallel Databases, 14(3), July 2003, pages 5-51.
- [vand05a]
van der Aalst, W.M.P., & ter Hofstede, A.H.M.. *YAWL: Yet Another Workflow Language*. Information Systems 30 (2005) 245—275
- [Vand00a]
Vanderdonckt, J., *A Small Knowledge-Based System for Selecting Interaction Styles*, Proc. of Int. Workshop on Tools for Working with Guidelines TFWWG'2000 (Biarritz, 7-8 October 2000), Springer-Verlag, London, 2000, pp. 247-262.

References

[Vand05b]

Vanderdonckt, J., *A MDA-Compliant Environment for Developing User Interfaces of Information Systems*, Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto, 13-17 June 2005), Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-31.

[vand96]

van der Veer, G.C., van der Lenting, B.F., & Bergevoet, B.A.J., *GTA: Groupware Task Analysis - Modeling Complexity*. Acta Psychologica 91 (1996) 297–322

[vand99]

van der Veer, G.C. & van Welie, M. *Groupware Task Analysis*. Tutorial notes for the CHI99 workshop “Task Analysis Meets Prototyping: Towards Seamless UI Development”. 16th May 1999, Pittsburgh PA, USA.

[vanW98]

van Welie M., van der Veer, G.C., & Eliëns, A. *An Ontology for Task World Models*. In Proc. of 5th International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS98 (3-5 June 1998, Abingdon, UK), pp. 57-70,

[vanW00]

van Welie, M., van der Veer, G.C., & Koster, A. *Integrated Representations for Task Modeling*. Tenth European Conference on Cognitive Ergonomics, pp. 129-138, 21-23 August 2000, Linköping, Sweden.

[Verb02]

Verbeek, H.M.W., Hirnschall, A. & van der Aalst, W.M.P. *XRL/Flower: Supporting Inter-Organizational Workflows Using XML/Petri-net Technology*. Web Services, e-Business, and the Semantic Web (WES): Foundations, Models, Architecture, Engineering and Applications, held in conjunction with CAiSE 2002 [The Fourteenth International Conference on Advanced Information Systems Engineering , May 27-28, 2002, Toronto, Ontario, Canada]. Workshop proceedings pages 535-552.

[Vern02]

Vernadat, F. *UEML: towards a unified enterprise modelling language*, International Journal of Production Research, 40: 17, 4309 — 4321, (2002).

[Visu07]

Visual Paradigm International Ltd. *Business Process Virtual ARCHITECT, User's guide*. Retrieved January 2, 2007.

From www.visual-paradigm.com/product/bpva/bpvadocuments.jsp

[vonG84]

von Glasersfeld, E. *An introduction to radical constructivism*. In: The Invented Reality. Ed: P. Watzlawick, New York, 1984. pp. 17-40.

References

W

- [W3C01] W3C Consortium, *XML Schema Specification, W3C Recommendation*, 2 May 2001. Available at <http://www.w3.org/XML/Schema#dev>
- [W3C05] W3C (2005) *Working Draft*. World Wide Web Consortium. 5 July 2005. <http://www.w3.org>
- [W3C10] W3C Model-based User Interfaces Incubator Group. Available on-line at http://www.w3.org/2005/Incubator/model-based-ui/wiki/Task_Meta_Model
- [Wang05] Wang, J., Kumar, A.: A Framework for Document-Driven Workflow Systems. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 285–301. Springer, Heidelberg (2005)
- [WfMC95] WfMC. *The Workflow Reference Model*. Workflow Management Coalition. Document Number TC00-1003. Document Status – Issue 1.1. 19-Jan-1995.
- [WfMC97] WfMC. *Workflow Management Coalition Terminology and Glossary*, WfMC-TC-1011, Document Status-Issue 2.0, June 1996. Specifying Task Models. (Proceedings Interac'97), Chapman&Hall, 1997, pp. 362-369.
- [WfMC99] WfMC (1999) *Terminology & Glossary*. Workflow Management Coalition. Document Number WfMC-TC-1011. Document Status – Issue 3.0. Feb-99.
- [WfMC08] Proces Definition Interface – *XML Process Definition Language*. Document Number WfMC-TC-1025. October 10, 2008. Version 2.1.a
- [Winc03] Winckler, M. & Palanque, P. (2003). *StateWebCharts: a Formal Description Technique Dedicated to Navigation Modeling of Web Applications*. In J. Jorge, N.J. Nunes, and J. Cunha (Eds.), Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS'2003 (Funchal, Madeira Island, Portugal, 4-6 June 2003). Lecture Notes in Computer Science, Vol. 2844. Berlin: Springer-Verlag, Berlin.
- [Winc04] Winckler, M., Barboni, E., Farenc, C., & Palanque, P. (2004). *SWCEditor: a Model-Based Tool for Interactive Modeling of Web Navigation*. In R. Jacob, Q. Limbourg, and J. Vanderdonckt (Eds.), Proc. of 4th International Conference on Computer-Aided Design of User Interface CADUI'2004 (Funchal, Portugal, 13-16 January 2004). Dordrecht: Kluwer Academics.
- [Winc05a] Winckler, M., Barboni, E., Farenc, Ch., & Palanque, P. (2005). *What Kind of Verification of Formal Navigation Modeling for Reliable and Usable Web Applications?*. In Proceedings of 1st International Workshop on Automated Specification and Verification of Web Sites WVV'2005 (Valencia, Spain, 14-15 March 2005).

References

[Winc05b]

Winckler, M. & Vanderdonckt, J. (2005). *Towards a User-Centered Design of Web Applications based on a Task Model*. In D. Schwabe, G. Rossi, L. Olsina, and V. Pelechano (Eds.), Proc. of Vth Int. Workshop on Web Oriented Software Technologies IWWOST'2005 (Porto, June 13th, 2005). Valencia: Technical University of Valencia, pp. 36-43. Also in CEUR Workshop Proceedings, Vol. 153. Accessible at <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-153/paper3.pdf>

[Wino86]

Winograd, T. *A language/action perspective on the design of cooperative work*. In CSCW Proceedings. ACM, New York, 1986, p.p. 203-220.

[Wodt97]

Wodtke, D., Weikum, G. *A Formal Foundation for Distributed Workflow Execution Based on State Charts*. In Proc. of the 6th International Conference on Database Theory (ICDT '97), Springer-Verlag, LNCS Series, p. 230-246, 1997.

Z

[Zelk08]

Zelkowitz, M.V., *An update to experimental models for validating computer technology*, J. Syst. Software (2008), doi:10.1016/j.jss.2008.06.040

[zurM02]

M. zur Muehlen. *Workflow-based Process Controlling. Foundation, Design, and Application of Workflow-driven Process Information Systems*. Logos Verlag Berlin. July, 2002.

Appendix A Workflow theory

The objective of *workflow management* is to support the execution of business processes. The parts of a business process that can be supported by a computer system are called *workflows*. In the glossary of the Workflow Management Coalition [WfMC97] a workflow is explained as the automation of a business process.

A *workflow specification* is a representation which supports automated manipulation, such as modeling or enactment by a workflow management system. A workflow specification mainly contains the same information as a business process description but at a more elaborated level of abstraction.

It defines a collection of tasks and the order of task invocation. Furthermore, it contains information relevant to controlling and coordination of the execution of its constituent tasks (e.g. required skills, possible actors, associated IT applications, processed data, and execution requirements).

An instance of a workflow specification is denoted as a *case*. In a case, concrete documents, information and/or tasks are passed to processing entities for action, according to the procedural rules determined in the workflow specification.

An example of a case is the process that handles an order from Paul W. The case may be distributed over several processing entities. Thus, the creditworthiness of Paul W. may be checked in the accountancy while the ordered item is already being assembled by employees from the production department. Cases are handled by a *workflow management system*.

A workflow management system (WFMS) is a computer system that implements workflow management functionality. This covers the definition of workflow specifications, their analysis, their simulation and the monitoring of the corresponding cases. Definition, analysis and simulation are done at design time. However, the core of the WFMS-functionality is the monitoring at run-time.

Monitoring of cases comprises their control and their coordination. For each monitored case, the WFMS ensures that the tasks of the case are performed in the right order, at the right time and by the right processing entities. This is done by activating tasks, assigning the embedded activities to processing entities and waiting for the activity to be completed. Which tasks are enabled at a certain point in time depends on the workflow specification. Enforcement of rules in a workflow specification by a WFMS is called enactment [WfMC97]. It is done by the workflow engine, sometimes also called workflow controller.

Appendix A

A WFMS system that is instantiated with one or more workflow specifications is called a *workflow system*, just like a database management system instantiated with one or more database schemas is called a database system.

Notations

In this section we will introduce the most widely workflow notation descriptions, Statechart Diagrams, Petri Nets, Business Process Model Notation, and UML Activity Diagrams.

Petri Nets

The concept of Petri net has its origin in Carl Adam Petri's dissertation submitted in 1962. Petri nets are graphical and mathematical modeling tools applicable to many systems. As a graphical tool, Petri nets can be used as a visual-communication aid similarly as flow charts, block diagrams, and networks. Tokens are used through the nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it is possible to set up state equations, algebraic equations, and other mathematical models governing the behavior of systems [Mura89]. Even more, Petri Nets are widely used to design, validate, and prototype user-driven interfaces [Bast90, Bast96].

The classical Petri net is a directed bipartite graph with two node types called *places* and *transitions* [vand98]. The nodes are connected via directed arcs. Connections between two nodes of the same type are not allowed.

A Place/Transition net (P/T net) is a triple (P, T, F) : P is a finite set of places, T is a finite set of transitions ($P \cap T = \emptyset$), $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation)¹.

Given a node x of $P \cup T$, the set $\bullet x = \{y | (y, x) \in F\}$ is a *preset* of x . The set $x^\bullet = \{y | (x, y) \in F\}$ is a *postset* of x . The elements in the preset (postset) of a place are called its input (output) transitions. Similarly, the elements in the preset (postset) of a transition are its input (output) places.

Given a set X of nodes, $X \subseteq P \cup T$, we define $\bullet X = \bigcup_{x \in X} \bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$. Given two sets of nodes X and Y , $X \setminus Y$ denotes the set of nodes of X that do not belong to Y .

The classical Petri net allows the modeling of states, events, conditions, synchronization, parallelism, choice, and iteration. It does not allow the modeling of data and time. To solve these problems, many extensions have been proposed. Three well-known extensions of the basic Petri net model are: (1) the extension with color to model data, (2) the extension with time, and (3) the extension with hie-

Appendix A

rarchy to structure large models. A Petri net extended with color, time, and hierarchy is called a *high-level Petri net*.

1. *Extension with color.* Tokens often represent objects (cases) in the system. If we want to represent attributes that are not easily represented by a token in a classical Petri net, the net model is extended with *color*. In a colored Petri net each token has a value often referred to as ‘color’. Transitions determine the values of the produced tokens on the basis of the values of the consumed tokens, i.e., a transition describes the relation between the values of the ‘input tokens’ and the values of the ‘output tokens’. It is also possible to specify ‘preconditions’ for the transitions, which take the colors of tokens to be consumed into account.
2. *Extension with time.* For real systems it is often important to describe the *temporal behavior* of the system. Since the classical net is not capable of handling quantitative time, a timing concept is added. There are many ways to introduce time into the Petri net. Time can be associated to tokens, places, and/or transitions.
3. *Extension with hierarchy.* In order to avoid the tendency to large and complex specifications for real systems, a hierarchy construct, called *subnet* is provided. A subnet is an aggregate of a number of places, transitions, and subsystems. These hierarchies can be used to structure large processes. In the first level a simple description of the process is given, and at the next levels another more detailed behaviors are given.

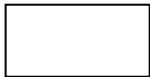
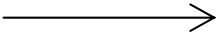
Notation	Definition	Graphical representation
<i>Transition</i>	Transitions are the active components of a Petri net. The triggering of a transition results in the state of the network being changed.	
<i>Arc</i>	Places and transitions can be linked by means of a directed arc.	
<i>Token</i>	The state of a Petri net is determined by the distribution of tokens amongst the places.	
<i>Place</i>	Places are the passive components of a Petri net. A place may contain no, one or more tokens.	

Figure A-1. Graphical representation of Petri Nets.

Appendix A

Statechart Diagrams

Harel statecharts (developed in 1987 by David Harel) are gaining some more general usage since a variant has become part of the Unified Modeling Language (UML). As it allow to model super states where activities can be part of a state. Classic state diagrams are so called "or diagrams", because the machine can only be in one state or the other. With Harel statecharts it is possible to model "and machines", where a machine is in two or more states at the same time. This is due to the possibility of having super states.

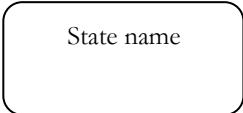
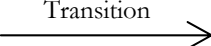
Notation	Definition	Graphical representation
<i>State</i>	A state marks a mode of the entity. The graphical representation is by using a rectangle with rounded corners and the name of the state.	
<i>Transition</i>	A transition marks the changing of the object state, caused by an event. The representation is by an arrow with the Even Name.	
<i>Initial State</i>	The initial state is the state of an object before any transitions; it is marked using a solid circle. Only one initial state is allowed on a diagram.	
<i>Final State</i>	The final states mark the destruction of the object whose state we are modeling. The final state is drawn using a solid circle with a surrounding circle.	

Figure A-1. Graphical representation of Statechart Diagrams.

Statechart diagrams are used to document the various modes ("state") that a class can go through, and the events that cause a state transition. The (UML) state diagram is essentially a state diagram with standardized notation that can describe a lot of things, from computer programs to business processes.

If an event E occurs in a state S and a condition C holds then make the transition to state T.

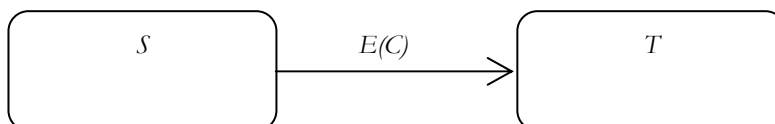


Figure A-2. A simple state machine.

Appendix A

The Harel semantics for transitions is based on the notion of an underlying event queue. If there is an event in the queue, it is processed according to the given state configuration, and a transition may be triggered. The transition is triggered and all associated actions are executed before the next event in the queue is examined.

Here is a simple example State Diagram that models the status of a user's account in a Bug Tracker system.

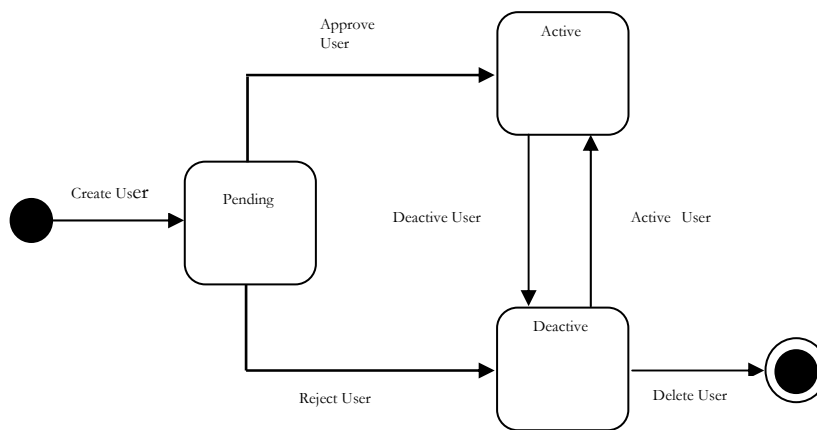


Figure A-3. An example of state diagram.

Business Process Modeling Notation (BPMN)

The Business Process Management Initiative (BPMI) has developed a standard Business Process Modeling Notation (BPMN). The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. BPMN will also be supported with an internal model that will enable the generation of executable BPEL4WS (Business Process Execution Language for Web Services).

BPMN defines a Business Process Diagram (BPD), which is based on a flow-charting technique tailored for creating graphical models of business process operations. A Business Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance. A BPD is made up of a set of graphical elements. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look-and-feel of the diagram. The four basic categories of elements are:

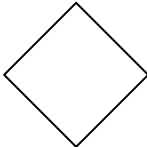
Appendix A

- Flow Objects
- Connecting Objects
- Swimlanes
- Artifacts

Business process modeling is used to communicate a wide variety of information to different audiences. BPMN is designed to cover many types of modeling and allows the creation of process segments as well as end-to-end business processes, at different levels of fidelity. Within the variety of process modeling objectives, there are two basic types of models that can be created with a BPD:

- Collaborative (Public) B2B Processes
- Internal (Private) Business Processes

We present below a list of the core modeling elements that are depicted by the notation.

Notation	Definition	Graphical representation
<i>Event</i>	An event is something that happens during the course of a business process. These events affect the flow of the process and usually have a cause or an impact. Events are circles with open centers to allow internal markers to differentiate different triggers or results. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End.	
<i>Activity</i>	An Activity is a generic term for work that company performs. An activity can be atomic or non-atomic. The types of activities that are a part of a Process Model are: Process, Sub-Process, and Task. Task and Sub-Process are rounded rectangles. Processes are either unbounded or contained within a Pool.	
<i>Gateway</i>	A Gateway is used to control the divergence and convergence of Sequence Flow. Thus, it will determine branching, forking, merging, and joining of paths. Internal Markers will	

Appendix A

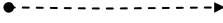
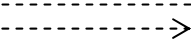
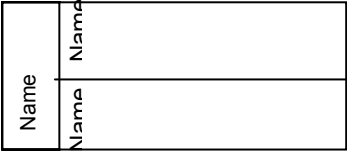
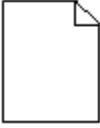
<i>Sequence Flow</i>	indicate the type of behavior control. A sequence Flow is used to show the order of those activities will be performed in a Process.	
<i>Message Flow</i>	A Message Flow is used to show the flow of message between two participants that are prepared to send and receive them. In BPMN, two separate Pools in the Diagram will represent the two participants (e.g., business entities or business roles).	
<i>Association</i>	An Association is used to associate information with Flow Objects. Text and graphical non-Flow Objects can be associated with the Flow Objects.	
<i>Pool</i>	A Pool represents a Participant in a Process. It is also acts as a “swimlane” and a graphical container for partitioning a set of activities from other Pools, usually in the context of B2B situations.	
<i>Lane</i>	A Lane is a sub-partition within a Pool and will extend the entire length of the Pool, either vertically or horizontally. Lanes are used to organized and categorize activities.	
<i>Data Object</i>	Data Objects are considered Artifacts because they do not have any direct effect on the Sequence Flow or Message Flow of the Process, but they do provide information about what activities require to be performed and/or what they produce.	 Name
<i>Group</i> (a box around a group of objects for documentation purposes)	A grouping of activities that does not affect the Sequence Flow. The grouping can be used for documentation or analysis purposes. Groups can also be used to identify the activities of a distributed transaction that is shown across Pools.	
<i>Text Annotation</i> (attached with an Association)	Text Annotations are a mechanism for a modeler to provide additional information for the reader of a BPMN Diagram.	 Descriptive Text Here

Figure A-4. Graphical representation of BPMN.

Appendix A

UML Activity Diagrams

An activity diagram is a directed graph, consisting of nodes and directed edges. It models the behavior of a system. A node represents a state of the system. In an atomic activity state the system waits for termination of an activity that has been enabled upon entry of the state. In a wait state the system waits for the occurrence of an event, e.g. some deadline occurs or a customer sends some additional information. A wait state is also used for synchronization of a thread with other parallel threads; in the wait state the system then waits for the completion of the other parallel threads. In a compound activity state another activity diagram is executed. This other activity diagram is started when the compound state is entered. When the activity diagram finishes, the compound activity state is left.

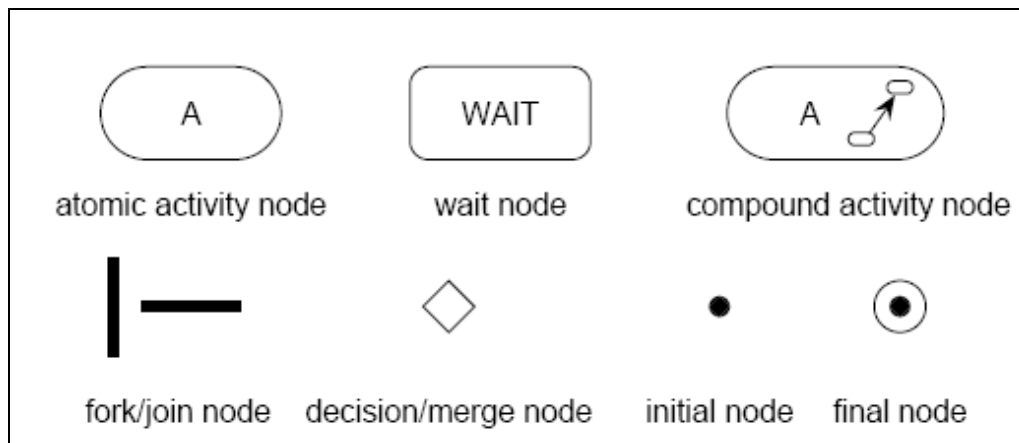


Figure A-5. Activity diagrams nodes.

The system starts in the initial state and ends in one or more final states. A final state means local termination of the corresponding thread; other parallel threads can still continue to execute. Using a fork (a bar with one incoming edge and more than one outgoing edge) a thread can be split in several parallel threads. Using a join (a bar with more than one incoming edge and one outgoing edge) multiple parallel threads can be merged into one thread. In a decision (a diamond with one incoming edge and more than one outgoing edge) one of the outgoing edges is chosen, if the incoming edge is taken. In a merge (a diamond with more than one incoming edge and one outgoing edge) the outgoing edge is taken, if one of the incoming edges is taken.

Nodes are linked by directed edges that represent sequence. We will use the term 'edge' throughout this thesis to stand for directed edge. The node that the edge leaves is called the *source*; the node that the edge enters is called the *target*. The edge always points at the target. An edge is labeled with an ECA rule $e[q]/a$, where e is

Appendix A

an event expression, c a guard condition expression, and a an action expression. Events are also called signals in UML. Each of these three components is optional. An edge with label $e[c]/a$ has the following meaning: If the system is in the source state, the event e occurs, and the guard condition c evaluates to true, then the source state is left, the actions a are performed, and the target state is entered. Since the transition is triggered by the occurrence of e , event e is called the *trigger event* or simply *trigger* of the edge in UML.

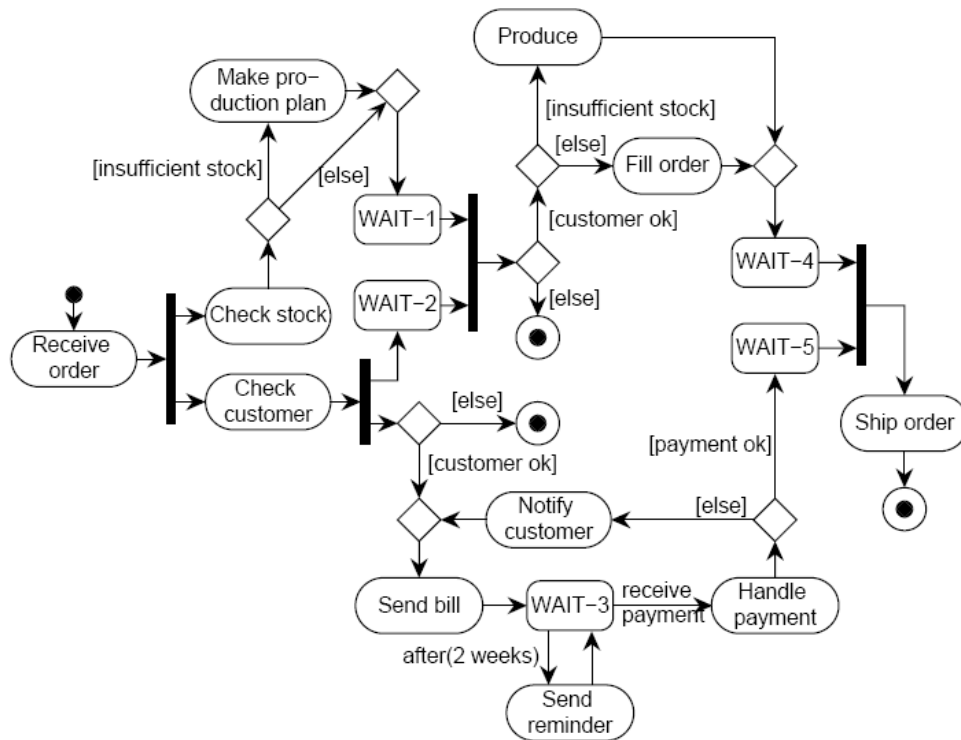


Figure A-6. An example of UML activity diagram.

Tool support

To manage the workflow, many academic and industrial research projects have been developed. The capabilities of these products are being enhanced in significant ways.

The Progression Model

The Progression model [Stav04] has incorporated some of the managing concepts of workflow to increase the flexibility in IS. It makes explicitly the steps and transactions as user undertakes when using an IS. As the user progresses towards accomplishing a task or goal, the progression model infrastructure records each

Appendix A

step and the state of the transaction and workflow. A progression is a workflow transaction and a sequence of scenes in a process to create a workflow transaction.

Progression Analyzer

User Interface

THE CAR RENTAL COMPANY

Pick-up Location or Airport Code

Pick-up Date: March 2004 ☐ AM ☐ PM

Drop-off Location or Airport Code

Drop-off Date: March 2004 ☐ AM ☐ PM

Car Type Weekend Special ☐

Transaction

Item	Value
Pick-up Location/A...	T1
Pick-up Day	
Pick-up Month	
Pick-up Year	
Pick-up Time	
Drop-off Location/...	
Drop-off Day	
Drop-off Month	
Drop-off Year	
Drop-off Time	
Car Type	
Weekend Special	
First Name	
Last Name	

Workflow

Scene	Name	Worker	Status
1	Choose Rental	Dave	In Progress
2	Enter Contact Information	Dave	Inactive
3	Enter Billing Information	Tara	Inactive
4	Send for Approval	Tara	Inactive
5	Provide Approval	Jen	Inactive
6	Confirm Reservation	Dave	Inactive

User Actions

- Enter Pick-up Location
- Enter Pick-up Airport
- Enter Drop-off Location
- Select Car Type
- Enter Pick-up Date
- Enter Drop-off Date
- Enter Drop-off Airport
- Choose Weekend Special

Transaction Actions

Workflow Actions

Markup Language Document

```
<progression>
  <scene>
    <gui>...</gui>
    <transaction>...</transaction>
    <workflow>...</workflow>
  </scene>...
```

Figure A-7. The progressing analyzing displaying the *Enter Name* scene.

The progression model is displayed to the user in the single progression section through four main panels. **USER INTERFACE** – this panel displays the widgets or interface elements that are specified in the user interface element of the markup document for the current scene. **TRANSACTION** – This panel display all the transaction items of the progression as specified in the workflow transaction element of the markup document, including the information that the user has entered up to that point in the progression. **WORKFLOW** – This panel display the status information for the progression as specified in the workflow element of the markup document. The workflow status shows the scenes of the progression in the recommended order. Each scene is associated with a name, the worker assigned to complete it, and the current status of its completion; these are all displayed in a table format. **FEEDBACK** – This panel consists of constraints, information status, and markup document. The constraints sub-panel display any information related to constraints that are violated throughout the progression. The information status sub-panel displays the information that is missing and re-

Appendix A

quired to complete the transaction. The markup document sub-panel displays the current state of the markup document. The single section also has buttons to start a new progression, open an existing progression, save the current progression, and quit the progression analyzer.

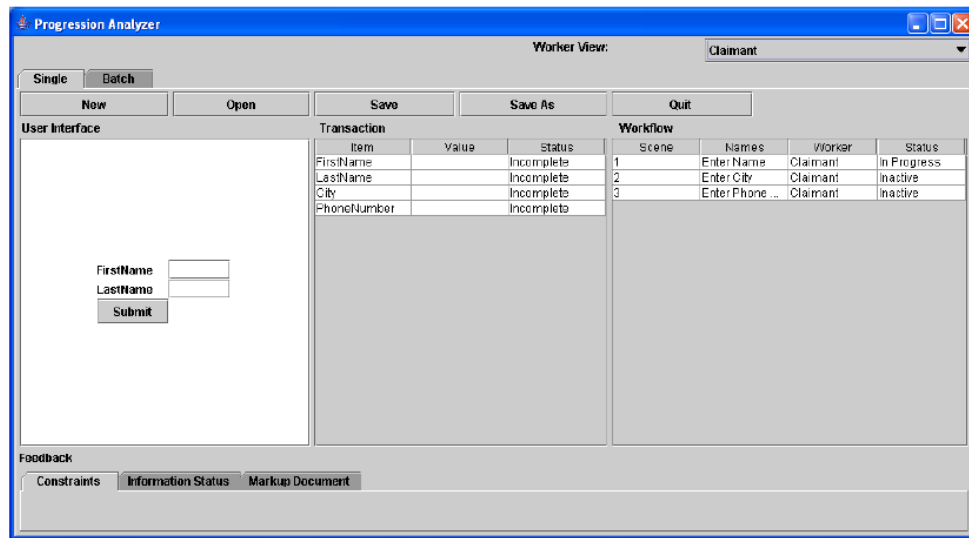


Figure A-8. The progression model in the Progression Analyzer.

Microsoft Windows Workflow Foundation

Microsoft Windows Workflow Foundation (WWF) [Espo05] is an extensible framework for developing workflow solutions on the Windows platform. It provides a single, unified model to create end-to-end solutions that span categories of applications, including human workflow and system workflow. WWF supports two fundamental workflow styles: sequential workflow and state machine workflow. A sequential workflow is useful for operations expressed by a pipeline if steps that execute one after the next until the last activity completes. Sequential workflows are not purely sequential in their execution, they can still receive external events or start parallel tasks, in which case the exact sequence of execution can vary somewhat. A state machine workflow is made up of a set of states, transitions and actions. One state is denoted as a start state, and then based on an event a transition can be made to another state. The state machine workflow can have a final state that determines the end of the workflow.

Appendix A

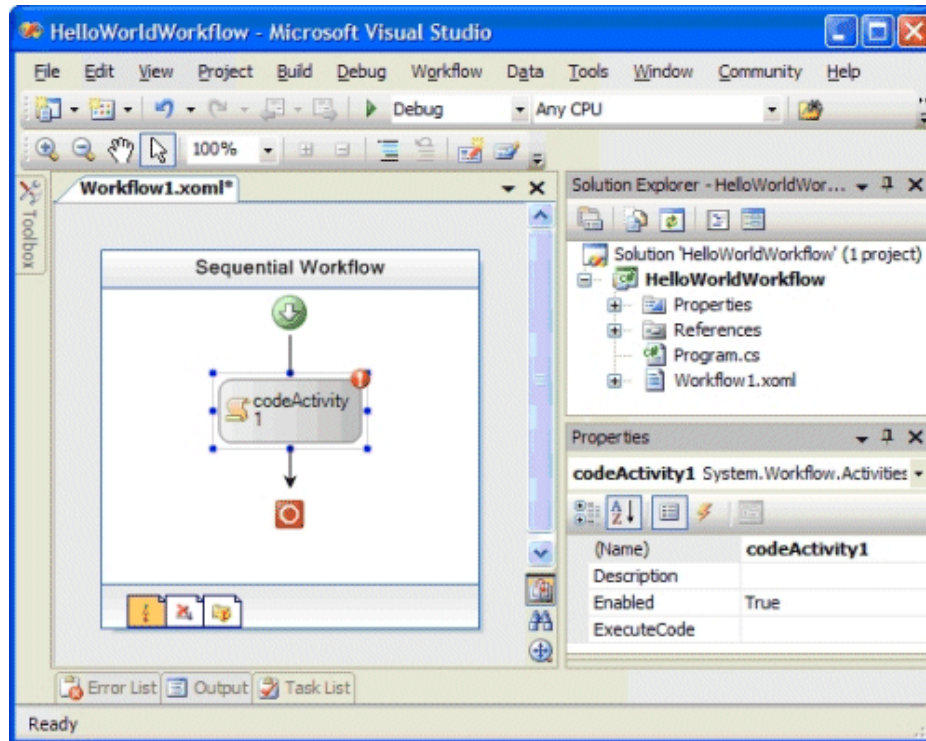


Figure A-9. The Visual Studio 2005 workflow designer. Source

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnlong/html/WFGetStart.asp>

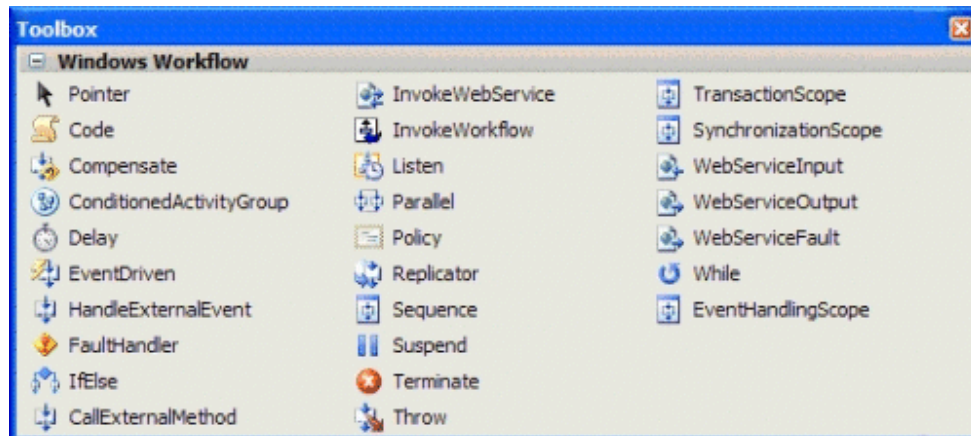


Figure A-10. Building blocks of a WPF workflow. Source

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnlong/html/WFGetStart.asp>

Appendix A

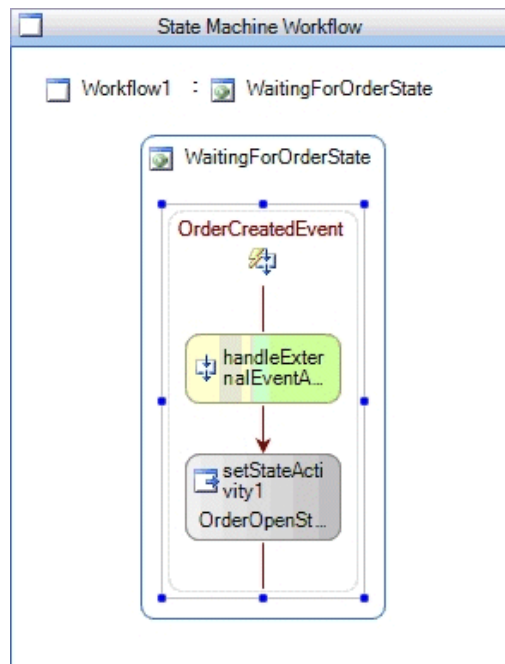


Figure A-11. An interior view of the OrderCreatedEvent EventDriven activity. Source

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnlong/html/WFGetStart.asp>

Flexo Business

The Flexo Business [Dena] is a Workflow Management System based on Petri Network formalism for process description. It interprets the workflow description, controls the instantiation of processes, sequences activities, adds work items to the user work lists and invokes application tools when necessary. Flexo Business is an extremely powerful yet simple tool that facilitates the collaborative definition of applications by modeling underlying processes, workflows, and graphical interfaces together with external connections to databases and existing services. The Flexo Workflow Engine interprets the Flexo model, enabling it to automatically run an application that interacts with databases and external services. Flexo Workflow Engine is also BPEL compatible (Business Process Execution Language) and therefore can be used in relation with other BPEL engines.

Appendix A

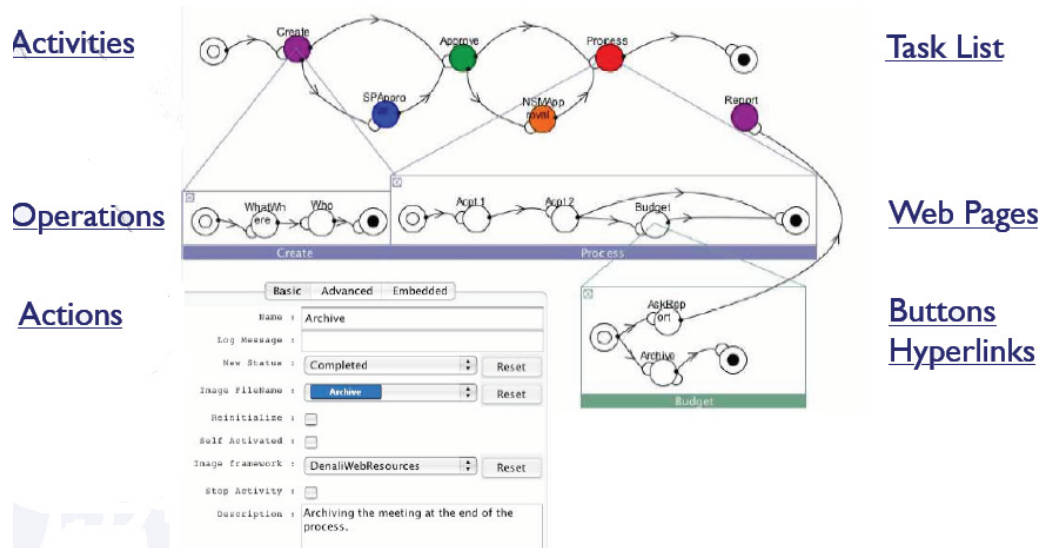


Figure A-12. Define operations & GUIs. Source

http://www.denali.be/MimeObjects/SharedContento/1000001/domsnyersslidesemjune30_1008394.pdf

Using intuitive and familiar drag-and-drop techniques from a palette of pre-configured items, business analysts can define the underlying work distribution and navigational pages of the application in seconds.

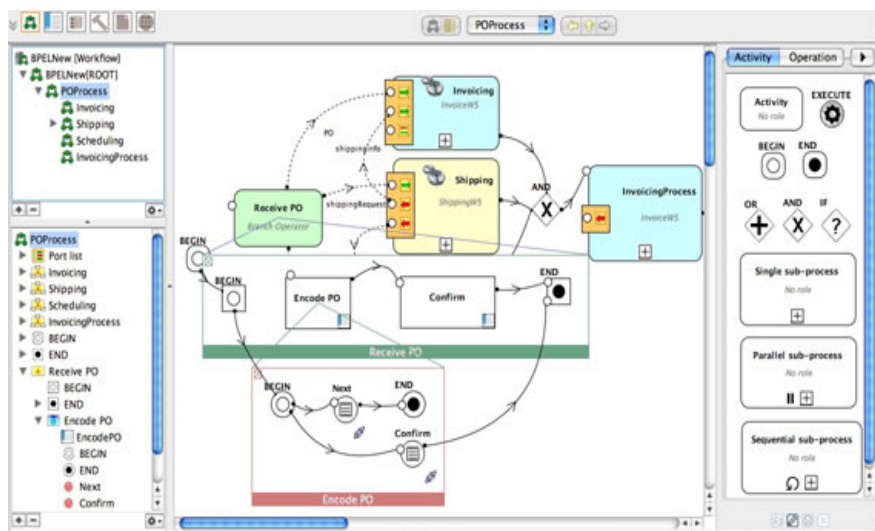


Figure A-13. The workflow modeler. Source <http://www.denali.be>

Appendix A

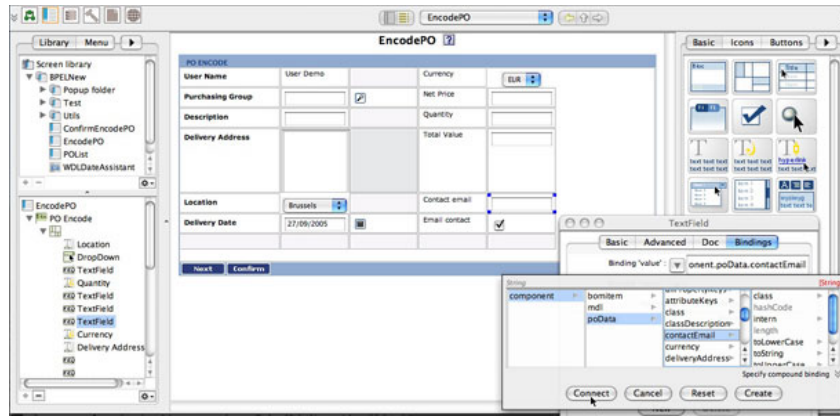


Figure A-14. Layout of graphical user interfaces. Source <http://www.denali.be>

Pages and forms are easily constructed by dragging/dropping texts, images and pre-configured interaction widgets (entry fields, drop-down lists, etc.) to their desired location. The addition of realistic data samples to the pages and the forms thus defined bring a first round of simple prototypes to life ready for improvement.

Business Process Visual ARCHITECT

Business Process Visual ARCHITECT (BP-VA) is a visual modeling tool that provides support for the Business Process Modeling Notation (BPMN).

BP-VA adopts the resource-centric interface, where context-sensitive shortcut buttons will be shown around the active diagram element. Each resource provides a functionality that you would likely to perform frequently, like creating a connection to a new/existing shape, opening the model specification, resizing a shape to fit.

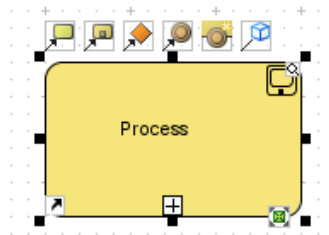


Figure A-15. Resource-centric interface.

BP-VA completely covers the BPMN, from model specification to graphical notation, including different presentation options.

Appendix A

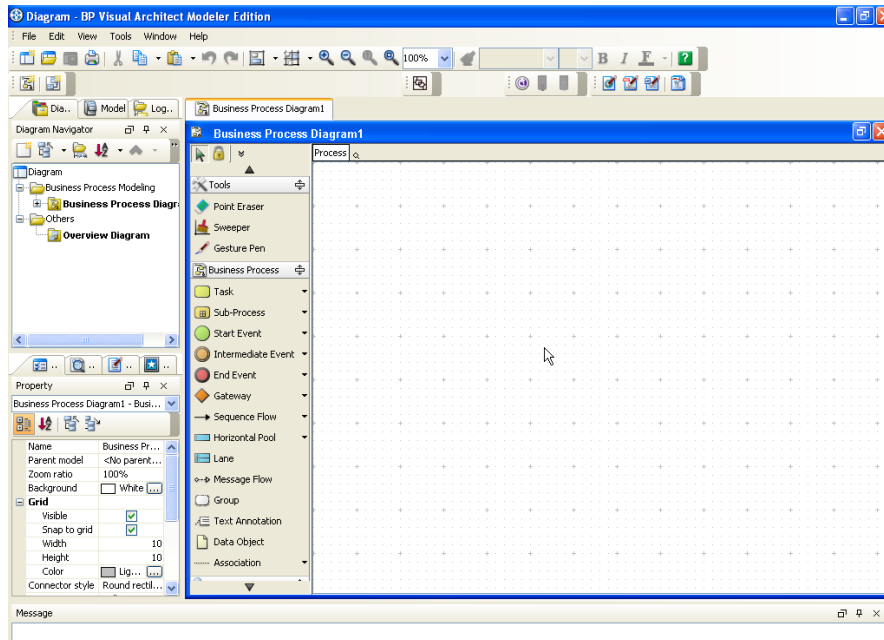


Figure A-16. User interface of BP-VA.

The Diagram pane is a tabbed view of all opened diagrams. You can click on the tab of a diagram to make it the active diagram for viewing or editing.

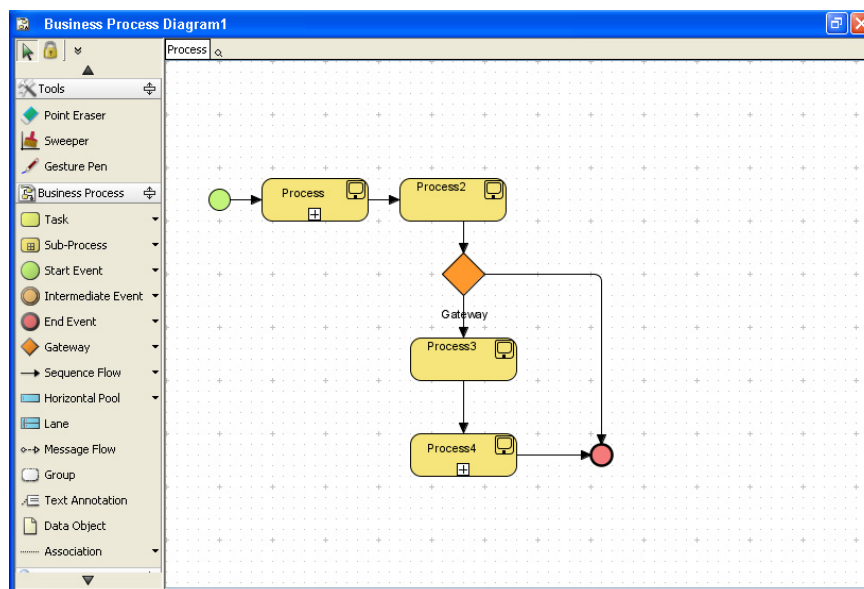


Figure A-17. Diagram pane.

Appendix A

WebSphere® MQ Workflow

WebSphere® MQ Workflow [IBM] supports long-running business process workflows as they interact with systems and people. Automates and tracks business processes in accordance with business design. Provides integration processes with rich support for human interactions. Enables use with WebSphere Business Integration Modeler and Monitor for design, analysis, simulation and monitoring of process improvements. Build time is the graphical process definition tool that is part of the WebSphere® MQ Workflow product. You can graphically define business processes and their activities to the level of detail needed for automation. Build time includes graphical support for declaring and documenting: 1) business rules on process navigation between steps, 2) business rules for role-based work assignment, 3) process interface definitions (data, programs, queues).

When you first start Build time, you see the Build time window as shown in Figure A-18. There is a *tree view* on the left of the Build time window that shows all the objects that belong to workflow models. The tabs at the top of the tree view provide a quick way to switch between the different trees. The tabs indicate that you can display object trees for *Processes*, *Staff*, *Network*, and *Implementations*. The right part of the Build time window is a *work area* that is used to display views of workflow elements. This can be the diagram view of a process or the properties that you can define for a selected object. At the bottom of the Build time window, there is a *Status bar*. The status bar shows information such as the name of the database you are using and your user ID.

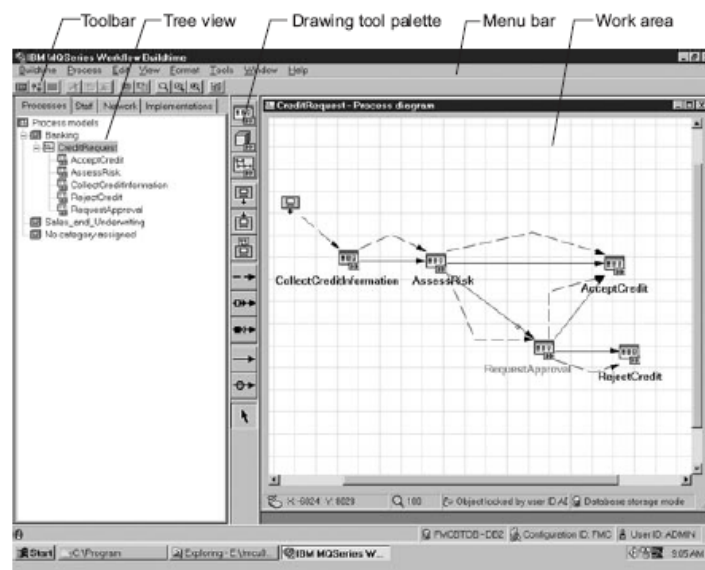


Figure A-18. Build time user interface.

Appendix A

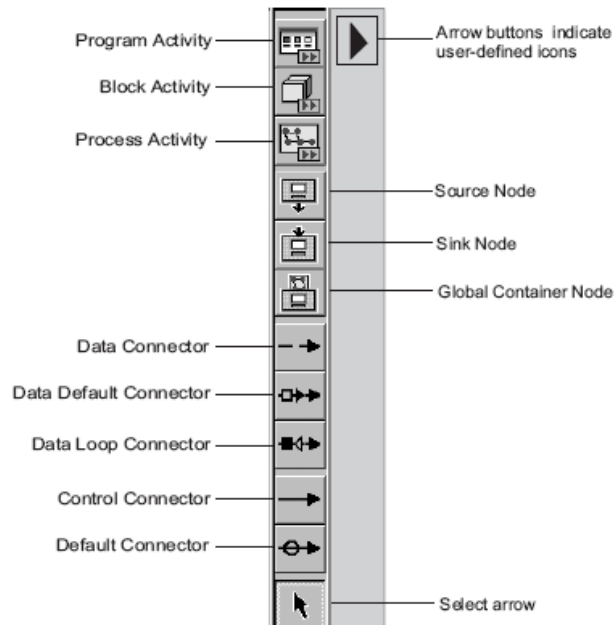


Figure A-19. The drawing tool palette.

i-Flow

Fujitsu's i-Flow is a customizable Business Process Management engine for web-centric applications. i-Flow provides an easy-to-use graphical user interface for controlling the process layer of your applications. i-Flow provides designers with the capability to chain processes together at run-time, spawning new processes as required. i-Flow incorporates advanced support for sub-processes which allows dynamic binding at run-time. Developers are able to build sophisticated dependencies between nodes and parallel threads of activity. Templates can be constructed using the browser-based Development Manager shown in the following illustration:

Appendix A

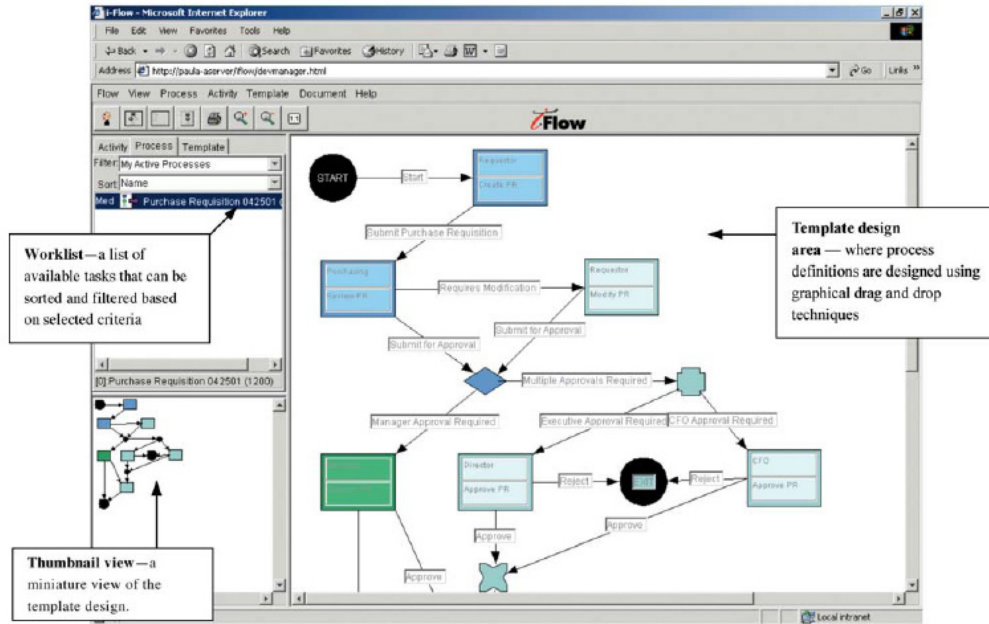


Figure A-20. The browser-based development manager.

The i-Flow provides a broad range of supporting functionality suitable for most process-oriented applications as follows:

Development Manager – used to design templates, start processes from those templates, modify processes at run-time, access tasks, and check process status and history. *Task Manager* – displays the user's assigned tasks and the forms and attachments associated with the tasks. Used to access, review and complete tasks. *New Process* – used to create new processes from existing templates. *E-mail Work Item* – When users receive an e-mail notification about a task, they click on the link in the message body to display the E-mail Work Item Client with the work item loaded automatically. *Administration* – used to import, export, archive or delete templates and modify or delete user profiles. Also it is used to create new versions of existing templates. *Server Dashboard* – systems management tool for monitoring adapter and server logs with graphical representations of i-Flow activity levels and key performance indicators. *i-Flow Reports* – used to display and print a variety of reports about processes, templates and tasks.

Nodes are the building blocks for constructing templates. i-Flow provides the following node types:

Appendix A

Start Nodes	Used to start a process	
Activities	Represent process tasks. They also specify any electronic forms associated with the task and the personnel assigned to carry out the task.	
Condition Nodes	An automated decision point where the BPM engine evaluates the specified condition and chooses the appropriate path.	
Split/Join Nodes	The Or and And nodes are typically used to split and synchronize the process flow.	  AND OR
Sub-Processes	Sub-processes allow the hierarchical decomposition and construction of nested processes, dramatically simplifying the construction of templates.	
Chained Processes	Allows a process to spawn new independent process instances at runtime, after which the calling process continues its execution.	
Delays	Used to suspend a process for a specified amount of time	
Exit Nodes	Closes the process and executes any scripts that are needed to clean up data sources or archive information and documents	

Figure A-21. Graphical representation of i-Flow.

Appendix B UsiXML, an ontology for UIs specification

UsiXML (which stands for User Interface eXtensible Markup Language) is a XML-compliant markup language that describes the UI for multiple contexts of use such as Character User Interfaces (CUIs), Graphical User Interfaces (GUIs), Auditory User Interfaces, and Multimodal User Interfaces. The conceptual framework of UsiXML relies on the Calvary's work [Calv03].

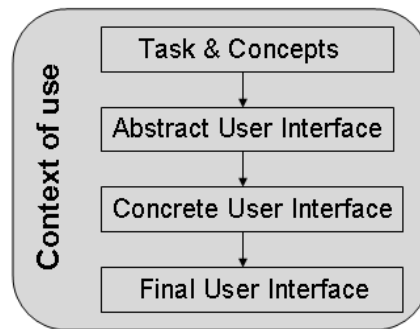


Figure B-1. Cameleon reference framework for multi-target UIs.

- *Final UI (FUI)*: is the operational UI i.e. any UI running on a particular computing platform either by interpretation (e.g., through a Web browser) or by execution (e.g., after compilation of code in an interactive development environment). The final UI has two possible representations, the code and the rendering. The code concerns the UI representation either as a set of instructions (in a procedural language) or as a set of assertions (in a declarative language), or a mix of both. The rendering of the system is a user perceivable representation of the UI.
- *Concrete UI (CUI)*: provides a specification of the user interface in terms of Concrete Interaction Objects and concrete relationships. Concrete objects and relationships provide a vocabulary that is as independent as possible of any programming language or toolkit used to implement the UI. A CUI is an abstraction of the FUI. A CUI defines widgets, layout and interface navigation and detailed behavior. Although a CUI makes explicit to the final Look & Feel of a FUI, it is still a mock-up that runs only within a particular environment. A

Appendix B.

CUI can also be considered as a reification (i.e., a concretization) of an AUI at the upper level.

- *Abstract UI (AUI)*: provides a specification of the user interface in terms of Abstract Interaction Objects (AIO) and abstract relationships. Abstract objects and relationships provide us with a vocabulary that is as independent as possible of any modality (e.g., graphical interaction, vocal interaction, speech synthesis and recognition, video-based interaction, virtual, augmented or mixed reality). An AUI can also be defined as a canonical expression of the rendering of the domain concepts and tasks in a way that is independent from any modality of interaction. The relations between the workspaces are inferred from the task relationships expressed at the upper level (task and concepts). AIOs are said to be widget-type independent. An AUI defines interaction spaces by grouping AUIs (and implicitly tasks of the task model) according to various criteria (e.g., task model structural patterns, cognitive load analysis, and semantic relationships identification). A set of abstract relationships is provided to organize AIOs in such a way that a derivation of navigation and layout is possible at the concrete level. An AUI is considered as an abstraction of a CUI with respect to modality.
- *Task & Domain (T&D)*: describe the various tasks to be carried out by the user in interaction with the system along with the domain-oriented concepts as they are required by these tasks to be performed. Domain objects are considered as instances of classes representing the concepts manipulated.

Task Model

A task model describes the various tasks to be carried out by a user in interaction with an interactive system. A task model is therefore composed of *tasks* and *task relationships*.

Tasks are, notably, described with a name, and a type. Task *type* may be: users, interactive, system or abstract. A user task refers to a cognitive action like taking a decision, or acquiring information. User tasks are notably useful to predict a task execution time. An interactive task involves an active interaction of the user with the system (e.g., selecting a value, browsing a collection of items). A system task is an action that is performed by the system (e.g., check a credit card number, display a banner). An abstract task is an intermediary construct allowing a grouping of tasks of different types. Tasks can also have attributes. A task *frequency* attribute is an assessment of the relative frequency of execution of a task. A task *importance* attribute assesses the relative importance of a task with respect to main user's goals. Frequency and importance are interesting attributes when it comes to

Appendix B.

adapt a UI to a constraining context imposing a UI to be pruned of some of its elements (e.g., as display space decreases it may be interesting to filter out widgets that allow the execution of unimportant tasks).

Action type and *action item* enable a refined expression of the nature of leaf tasks (sometimes called action tasks or leaf tasks). The taxonomy is twofold: a verb describes the type of activity at hand; an expression designates the type of object on which the action is operated. By combining these two dimensions a derivation of interaction objects supposed to support a task becomes possible.

Task relationships are of two main types: decomposition and temporal.

- *Decomposition* enables representing the hierarchical structure of a task tree.
- *Temporal* allows specifying a temporal relationship between sibling tasks of a task tree. LOTOS [Pate97] operators are used here.

Domain Model

A *domain model* describes the real-world concepts, and their interactions as understood by users and the operations that are possible on these concepts [DSou99]. He selected UML class diagrams as the basis of expression for his domain model. We considered UML class diagrams as Extended Entity Relationship model (EER) [Teor86].

The Domain model concepts are:

- *domainClass*. Classes describe the characteristics (attributes and methods.) of a set of objects sharing a set of common properties.
- *Attribute*. Attributes enable a description of a particular feature of a class.
 - The *type* of an attribute refers to common data types found in most programming language i.e., Boolean, char, string, integer, float.
 - The *cardinality* of an attribute indicates the number of values an attribute may be associated with. The cardinality can be specified by providing two integers: a minimal cardinality and a maximal cardinality.
 - An original typology allows characterizing a type of *domain* for an attribute. Indeed, **attributeDomainCharacterization** takes the value of: interval, continuous interval, discrete interval, linear interval, circular interval, set[n] (where n is the number of possible values in an attribute domain). When used in combination with a task model, this typology helps to map domain attributes to a type of interaction object by which it will be rendered. For instance, a “choose element” task on an

Appendix B.

attribute with a circular interval enables the derivation of a (multi-state) toggle button.

- *Methods* (in this context) are presences which are called either by objects of the domain or by user interface components. Methods manipulate object's attributes. Methods are, here, described with their signature i.e., with their name, type, and parameters.
- *Objects* are instances of a class. An object is composed of attribute instances which may have values and define the state of an object.
- *domain class relationships* describe various types of relationships between classes. They can be classified in different types: *generalization*, *aggregation*, *usage*, *materialization*, *instantiation* and *ad hoc*. Class relationships are described with several attributes enabling the specification of role names and cardinalities.

Abstract User Interface Model

Abstract User Interface (AUI) model is a user interface model that represents a canonical expression of the renderings and manipulation of the domain concepts and functions in a way that is as independent as possible from modalities and computing platform specificities.

The AUI is populated by *Abstract Interaction Objects* (AIO) and *abstract user interface relationships*. These concepts constitute a vocabulary that is independent of the modality and the computing resources for which a system is targeted at.

A modality (also called interaction technique) can be defined more precisely, after [Niga95], as the coupling of a physical device d with an interaction language L : $\langle d, L \rangle$. The language supports, at the concrete level, two modalities: speech (i.e. auditory) input and output and graphic (i.e., graphical) input and output.

Abstract Interaction Object (AIO) may be of two types *Abstract Individual Components* (AIC) and *Abstract Containers* (AC).

An *Abstract Individual Component* (AIC) is an abstraction that allows the description of interaction objects in a way that is independent of the modality in which it will be rendered in the physical world. An AIC may be composed of multiple facets. Each facet describes a particular function an AIC may endorse in the physical world. Four main facets are identified:

- An *input* facet describes the input action supported by an AIC.
- An *output* facet describes what data may be presented to the user by an AIC.
- A *navigation* facet describes the possible container transition a particular AIC may enable.

Appendix B.

- A *control* facet describes the links between an AIC and system functions i.e., methods from the domain model when existing.

A single AIC may assume several facets at the same time. The AIO that reifies this multi-facetted AIO will assume all those ‘functionalities’. For instance, a CIO may display an output while accepting an input from a user, ensure a transition between windows and trigger a method defined in the domain model.

An *Abstract Container* (AC) is an entity allowing a logical grouping of other abstract containers or abstract individual components. AC are said to support the execution of a set of logically/semantically connected tasks. Actually AC may be reified, at the concrete level, into one or more graphical containers like windows, dialog boxes, layout boxes or time slots in the case of auditory user interfaces. However there is no concretization of these objects for 3D UIs.

Abstract User Interface Relationships (AUI relationship) are relationships that can be drawn between abstract interaction objects of all kinds.

Five types of abstract relationships may be defined at this level:

- *Decomposition* relationship allows specifying a hierarchical structure of abstract containers and abstract individual components.
- *AbstractAdjacency* relationship indicates that two AIO are logically adjacent.
- *Spatio-temporal* relationship allows a specification of a very precise layout in time or space in a way that is independent of any modality.
- *Dialog control* relationship allows a specification of a flow of control between the abstract interaction objects.
- *Mutual emphasis* relationship allows specifying that two components should be somehow differentiated at the concrete level. This relationship may be useful in a user interface where the probability of confusing two UI elements is high (e.g., in an airplane cockpit, a field displaying the angular speed and the absolute speed).

Concrete User Interface Model

The Concrete User Interface Model (CUI) represents a concretization of an AUI model. A CUI is populated by Concrete Interaction Objects and Concrete User Interface relationships between them. The CUI model is a UI model allowing a specification of an appearance and behavior of a UI with elements that can be perceived by users.

Appendix B.

By definition, a CUI is modality dependent as any CUI instance refers to the interaction modalities that have been selected for this UI. In contrast to its modality dependence, a CUI remains toolkit independent as no CUI instance does refer to any physical element (i.e., toolkit elements or widget) of the computing platform. Nonetheless, a CUI description can be detailed enough to allow a complete rendering of a user interface.

A CUI model is composed of *Concrete Interaction Objects* (CIO) and *cui relationships*.

A Concrete Interaction Object (CIO) is defined as an entity that users can perceive and/or manipulate (e.g., a push button, a list box, a check box, a sound). The actual specification realizes an abstraction of widget sets found in popular toolkits: 2D graphical (Java Swing, HTML 4.01, Flash) and auditory (earcons and VoiceXML 2.0). In other words, CIOs allows an expression of UI elements that is independent of their actual rendering.

Graphical and auditory CIOs are further decomposed into containers and individual components. We have just summarized the main characteristics of the actual model more information can be found in the [USIX06] documentation.

In Graphical containers (GC) attributes used are as abstract as possible in order to respect the independence on implementation.

Graphical Individual Components (*GIC*): text components are differentiated in two types, for input (an input field, a password field, a multi-line input field) and output (a label, a complex textual output as a rtf file) purposes.

Vocal Containers represent a logical grouping of other auditory containers or auditory individual components. *Vocal individual components* are of five types: auditory output which may consist in music, voice or a simple “earcon” (i.e., an auditory icon), auditory input which is a mere time slot allowing the user to provide an auditory input using her voice, or any other physical device able to produce sound, vocal navigation (Specifies a transition to another vocalForm), break (Interrupts the execution of the current vocalContainer) and exit (Terminates the execution of the vocal interface).

Similarly to Concrete Interaction Objects they are divided into *vocal relationships* and *graphical relationships*. *Dialog control relationship* can be defined between both types of interaction objects [USIX06].

Vocal relationships are of three types: *vocal transition* that enables to specify a transition between two auditory containers; *vocalAdjacency* that indicates a time adjacency between two auditory components; and *vocalContainment* that allows adding or deleting *vocalIndividualComponents* from a *vocalContainer*.

Appendix B.

Graphical relationships are of five types: *Graphical transition* specifies navigation links between the different containers populating the UI, *alignment* that may also be specified among any individual component belonging to the same window, *adjacency* indicates that two components are topologically adjacent, *emphasis* enables to specify that two or more graphicalIndividualComponents must be differentiated in some way (e.g., with different color attributes) and *containment* analog to the vocal containment, allows to specify that a graphicalContainer embeds one or more graphicalContainers or one or more graphicalIndividualComponents.

Dialog control allows a specification of a flow of control between the concrete interaction objects. As so a dialog control may be specified independently of a task model. LOTOS operators are used for this purpose. For instance a relationship `CIC1.EnterCountry []> CIC2.EnterProvince`, indicates that `CIC2` cannot be initiated while `CIC1` is not terminated and that `CIC1` has provided a value for the data on which the two component synchronize with.

Any CIO may be associated with any number of *behaviors*. A *behavior* is the description of an event-response mechanism that results in a system state change. The specification of a behavior may be decomposed into three types of elements: an *event*, a *condition*, and an *action* [USIX06].

An *event* is a description of a run-time occurrence that triggers an action. They consist of any system event (i.e., issued from a process belonging to the domain), user interface event (i.e., issued in the context of the user interface). A limitation on the events is that they cannot make any reference to coordinates, which is imperative in 3D event handling. Events can be composed into more complex event expressions using a subset of the LOTOS operators introduced earlier. However, as it is not part of the language, the behavior description is straightforward from the actual [USIX06] specification.

A *condition* is the expression of a state that has to hold true before (pre-condition) or after (post-condition) an action is performed. A condition may be positive or negative. An *action* is a process that results in a state change in the system. An action can be of three types: a *method call*, a *transformation system*, or a *transition*.

A *method call* is a call to a method that is external to the UI. If a domain model exists, all method calls must reference a method belonging to this model. A method call is normally specified with the name of the method (under the form `Class.methodName`), but other referencing techniques are not forbidden. The method call parameters can be specified by making a reference to the value of a property of an object belonging to the CUI.

Appendix B.

A *transformation system* is the expression of any property change at the UI level. We use a mechanism to specify property changes on the UI. This mechanism is similar to the one that will be introduced in Chapter 4. To avoid too much forward reference, it can be said that a transformation system can be explained as follows: when a pattern is found in CUI specification, changes should occur on the elements matching the pattern. A transformation system might be, for instance, “when a green button is found in the specification, change the color property of this button to red” or “For all text components belonging to the main window, increase their font by a factor of 2”.

A *transition*, also called *navigation*, is a description of a change in the container’s visibility property of a user interface system. A transition has a source (a navigation individual component) and a target (generally a container). Depending on the type of modality, transitions may be of different types (see above in this Section).

Appendix C Cognitive dimension criteria evaluation

Viscosity

Viscosity is a property of the system as a whole. It refers to resistance of the system to change. This means that it becomes hard after modifying our model to get a desire state. Changes may be related to different operations, such as: adding, removing, modifying, consequently viscosity may be very different depending on the operation and the operator. For instance, viscous is the operation of removing an organizational unit while fluid could be adding a task to a process. It is known that viscosity might be problematic in an exploratory design if not tackled correctly [Gree98] for at least one reason: redesigning in a graphical editor usually requires much tedious work, and frequently many similar alterations need to be made to different parts of the picture.

Adding, removing and/or modifying text in the model elicitation tool is considered less viscous for all the advantages that it offers, such as: automatic identification of action verbs and the *transcription* of textual scenarios into a task spreadsheet.

Adding, removing and/or modifying a task in the task spreadsheet editor is considered less viscous there is no affection on the task list when an operations is done on the task identification, on the contrary the *transcription* of the changes to the workflow editor is automatically done.

Adding, removing, and/or modifying a job from the job editor is considered viscous because of its impact in the model might require manual adaptation of the process model to determine the role in charge to perform the task previously assigned to other roles. Similarly, adding, removing, and/or modifying a user stereotype is considered viscous because of its impact in the model. Likewise, when operations are on organizational units where a whole restructuration of the workflow might be needed if an organizational unit disappear. In any case organizational changes have a viscous nature per se and it is good to have a viscous system supporting operations related.

The process modeling is an activity that we limit to the structuration of the workflow in Petri Nets. When operations over the Petri net take place we could

Appendix C

imagine different scenarios with viscous or less viscous impact. Adding, removing or modifying a process of the Petri Net might require lines connecting it to other boxes will have to be moved (viscosity problem known as *knock-on*). Then most each line will have to be redrawn individually (problem known as *repetition*). Allocating a task has been explored in deep. Task allocation patterns have constraints while combined. This has been explored and the tool checks the validity of any attempt to add, change, and remove an allocation pattern. The problem becomes viscous when such changes need to be propagated, *knock-on*, and new jobs satisfies the new allocation model. The task model is viscous due to its hierarchical structure. Deleting leaf tasks might not produce a lot of work in reorganizing the tree structure but when the deletion is done on a parent node then all the lines connecting it to other nodes will have to be moved, *knock-on*. Then most each line will have to be redrawn individually, *repetition*. As a whole, we consider the *viscosity* of FlowiXML as acceptable as most of the operations it supports are less viscous and those which are viscous are due to the intrinsic nature of their representation (Petri Nets and Task Models) and the operations (organizational changes).

Hidden Dependencies

Hidden dependencies refer to a one-way pointer, where A points to B but B does not contain a back-pointer to A. This problem is of interest because there a lot of dependencies in the workflow that they might be explicit as much as possible.

The text of the requirements has hidden dependencies, as it is possible that deleting text might imply to delete task, organizational units, and so. It is hard to imagine how from a textual description possible dependencies might be highlighted to prevent harmful. Similarly, to the previous aspect, task dependencies cannot be easily foreseen at this level. However, the spreadsheet provides information that is relevant to the user to prevent him from possible dependencies, such as: rationale of the task. Of course the cognitive load to recognize that dependency is significant, particularly for non-experts. Thanks to graphical notation selected in FlowiXML it is clear and explicit dependencies between job, organizational unit and processes. Petri Nets make the dependencies explicit. The task model that details the work to be performed in a task of the process model is hidden in the process model view; this information is not fully available, unless the task model editor is launched. Task allocation patterns have been explored to identify their dependencies which become explicit. Task models use a hierarchical structure with links showing dependencies explicitly. Overall the hidden dependencies are not a problem in FlowiXML there were avoid thanks to the explicitness of the notations (task model and process model). However, intermodel dependencies have not been fully considered. We support forward *transcription* and keep consistency from

Appendix C

the model elicitation until the process model. When a change is made on other models such as the process model the backward traceability is not supported until the requirements elicitation model. A solution to this problem is foreseen by having a rule like: When a new task I added to the process model then create text in the model elicitation tool: 'The **new task** is performed in the **organizational unit X** by the **job a**, after the **precedent tasks**, and gives input to the **next task**. This is not a full textual description but then it could be refined by the user.

Premature Commitment

Constraints on the order of doing things force the user to make a decision before the proper information is available. It is known [Gree98] that experts recognize potential problems much earlier, perhaps not from looking ahead but by familiarity with likely sources of difficulty. We notice that this is something that is problematic because our methodology, ideally, follows a series of steps from which decisions made have an impact. This is why we recognize this feature and act to prevent the conflict. From the textual scenario is vital to identify most relevant elements of the structure but novice users lack to understand and highlight the correct experienced this type of problems with the user of the system. Clearly differentiating a task, from process and from workflow is vital at early stages of the development. This is why we provide guidance, with our workflow identification criteria, to non-expert users of the methodology to properly select among the options; this solution is known as *decoupling*. Planning task allocation patterns is vital before selecting them. We provide guidance with our tables showing the constraints and dependencies between different patterns. Workflow designers have information that could help them to make the right decision about the type of resource allocation pattern to be assigned. Overall FlowiXML and the nature of our methodology force some *premature commitment* while modeling the workflow. Research has shown that expert designers frequently treat potential trouble-spots differently, putting them aside for later treatment or attacking them early [Gree98]. For novice experts we provide methodological guidance trying to avoid problems derived from wrong decisions.

Secondary notation

Extra information carried by other means than the official syntax. All concepts from the workflow model can be exported to a XML-format that can be processed to check the reachability of the tasks and completeness of the model. The organization also can be viewed as a hierarchical diagram. Thus, FlowiXML addresses the need of the secondary notation for most of their modules by expressing the graphical notation as a XML-based format.

Visibility

Extra information carried by other means than the official syntax. The *exploratory design* is encouraged to use *juxtaposition* (giving the ability to put two or more items visible at the same time) as a seed for problem-solving [Gree98].

All aspects of the organizational modeling are visible and available to be used. In case that a modification takes place menus offer the option to open the appropriate editor. Menus are always contextualized within the different windows of each tool. For instance if a new user stereotype is added, the workflow designer can compare with an existing workflow model what is the job that the user stereotype needs to be allocated to a task. The process model offers a map view that allows the designer for a quick scan of the model. This miniature map-view reduces the visibility problem in large problems. Task modeling using a hierarchical structure presents a visibility problem when the task model deep (level of the tree) increases. Even that it is not yet implemented, we could imagine a similar map-view as the one used for the process model to visually scan the task model.

Selecting a resource allocation pattern is a complex task that demands perfect understanding and knowledge about the patterns. We have visibility problems for this activity because we do not provide any possibility to the workflow designer to compare a pattern assigned with other task, *side-by-side viewing*. Although we tried to organize and present them in a consistent way via the WUIPs we did not find clear understanding by the users. Also, it would be nice to provide some guidance to remind the user the meaning of the patterns and their applicability, although we provide this information on paper, which reduces to some extent the visibility problem. So as for the table of constraints for resource allocation patterns, to prevent the user from the consequences of their decision they can check the table on paper, ideally this should be presented digitally on the screen within the tool.

FlowiXML considers the importance and relevance of visibility. We consider that visibility is acceptable with some aspects still to be improved. For instance, Problems might be related to workflow diagrams or task model diagrams when they grow. So far, from the case studies we have not yet identified this problem but with more complex problems they might arise.

Abstractions

Abstractions refer to the number of models and the nature they have in order to help workflow designers to design workflow models. We consider that we provide a sufficient number of abstraction that do not hide any information, thus the user does not need to think about building new concepts on top of our notation to design their workflow models. This means that no new constructions are expected

in order to create new models. However, the workflow designer must integrate concepts to build a workflow model, for instance, an organizational unit is composed of jobs, user stereotypes, process models, task models, and this is something that must be done by hand but does not represent new abstraction to our system.

Closeness of Mapping

Relying on a notation close to the domain of application is important. In FlowiXML we did survey different notations, so as methodologies, and we tried to add here our solution to meaningful concepts closed to the domain of application. This is the case of the Petri nets for workflow representation, the notation of the task model, the task resource allocation patterns.

Consistency

Consistency is one of the most important aspects of usability. We can discuss consistency at different levels with regard to FlowiXML. First, there consistence in the way all models and concepts are represented, meta-models using UML class diagrams. Second, all models use the same UIDL thus keeping consistency at the language representation. The notation selected for the different models is consistent with existing knowledge on the different topics. Finally, the tool implemented lacks of consistency for the nodes manipulation in the graphical editors for Petri Nets and Task Models. This inconsistency is related to selection and manipulation, release operations.

Diffuseness

The notation for FlowiXML must not be diffuse. We consider that the different notation used and not diffuse, composed of a reduced set of components that make them easy to understand and, what is more important, to apply Modeling resource allocation patterns however did not show to be clear, even with the WUIPs. The understanding and knowledge require to apply them requires to keep in mind a long description and example of their applicability. Unfortunately, we did not find a simpler representation.

Error-proneness

Error-proneness evaluates the notation in terms of the errors it produces. Different could be the sources of errors but directly attributed to the notation we found that to evaluate. The cognitive load due to diffuseness or consistency problems might be source of errors. Not being consistent with the interaction with nodes prove to be a source of users dissatisfaction when using the task model editor, it was hard to manipulate the node. Tool by tool this problem was not identified,

Appendix C

for instance, Petri Nets come along with three basic elements, connector, transition, and place, where connections are just possible between a state and a place, thus, the error were not possible. Related to the models, task attributes we invite users to use the task life-cycle to understand task' attributes.

Hard mental operations

High demand on cognitive resources was considered as a result of considering other criteria, such as: viscosity, consistency, diffuseness, reduced but yet extent set of abstractions. Nevertheless, as discussed in other sections, the results are not perfect. Particularly with respect of the task allocation patterns, from the external evaluation we notice that they were hard to understand and off course to use correctly. However, there were some cases in which users showed that their use was straightforward. Without being an expert in psychology, we adventure to say that cognitive capacities for abstraction made a difference when using FlowiXML.

Progressive evaluation

The evaluation of the workflow modeling can be assessed at any time via the different tools. We provide mechanism to check the reachability of the workflow model, completeness of the model, and provide a checklist, this serves to evaluate the progress of the workflow modeling.

Provisionality

The provisionality refers to the degree of the premature commitment, i.e., how hard is going back from our actions. Considering the premature commitment that we have identify for FlowiXML, which we considered to be important, we consider the degree of commitment to be reasonable. We meant that it is possible to go back in the decisions made but in some cases it might imply more than just deleting and adding a concept.

Role-expressiveness

This criterion refers to the facility that the notation to split into parts. In FlowiXML the designer can jump to see the work from the organizational point of view and then look at details of the organizational units or the details of the process. Even that this criteria is one that was not clearly detailed, at the time of producing their document [Gree98], we considered that we satisfy it.

Appendix D Workflow editor user guide

Workflow Tools

Several tools are included in the main program. Those are: a model elicitation tool, a task spreadsheet, a job edition tool, a userstereotype editor, and a workflow editor. Each program allows defining a subset of the data's used to define a workflow, and can be perceived as a “view” of those data's. The consequence of this is the fact you won't have to introduce the same information twice, depending on the tool. For example, if you define a task in the task spreadsheet, the same task will be put in the workflow editor scheme, becoming directly available for further treatments.

The different options are: File/ [new] [load] [save] [export xml]

Save and load are made into a .wft format. Export requires to introduce the name of a directory (not a file!) in which the files corresponding to the different models will be written.

Keep in mind that saving is only useful in this part of the program. Every time you use one of the sub programs (model elicitation, task spreadsheet,), information will be kept at this level of the program.

Model Elicitation Tool

The Identification Tool allows the user to identify tasks, units, jobs and resources from a textual scenario. When drawing the workflow, it is possible to reuse that information. The model elicitation is base on pattern-matching. So once a task is defined, for example “fill the lab papers”, each occurrence of those terms will be colored. The color depends on the class to which the information belongs.

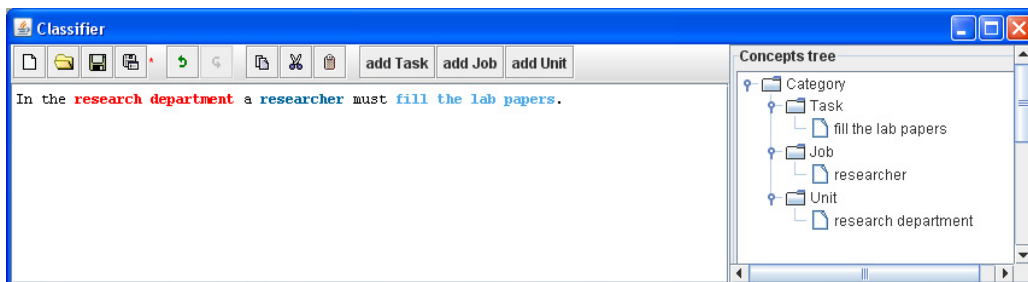


Figure D-1. Model elicitation tool.

Appendix D

The use of the elicitation tool is quite simple. At first you select a portion of text. Then you use the right click on the mouse. A menu will appear, allowing you to choose which kind of element is concerned. On the right part, a tree allows you to visualize the information you elicited. For attributes that are (by definition) relative to another concept, you have to select this concept (for example, a definite task) in the tree at first, by a left-click action with your mouse.

When adding a task, an option window will invite you to introduce a task type. This is additional information about the task. It may either be custom (user types it), predefined (choice inside of a selection), or pattern (a pattern can be seen as a set of different types, and will result in the creation of as many tasks as there are different types).

Most of options are disabled, apart from operations about the text itself (new, load, save). The text is not saved by default, so the principle is to load one (or more) scenarios as to successively elicitate (i.e. extract) the different parts of the workflow.

Task spreadsheet

Each task is defined by the following attributes: id, name, predicate (task id of previous task), definition, nature and rationale.

It is possible to export the list of tasks into an excel file.

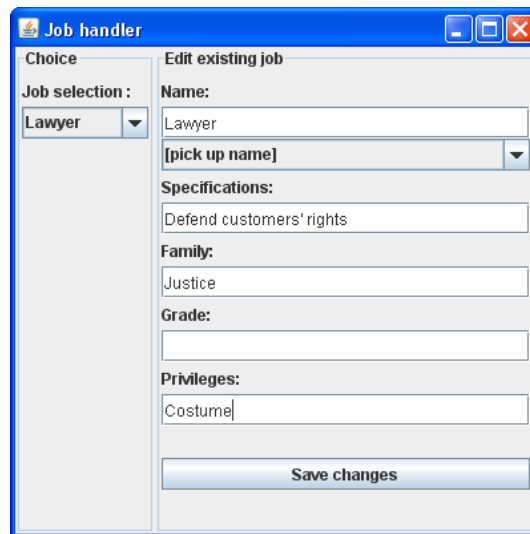
Three different actions are available: add task, delete task and sort table. Sorting table should order them on basis of the task id.

!!! In task table program, click on another square in order to finish current square edition. Otherwise last square changes are not taken into account.

Job Edition

The Job Edition is used to define jobs, whose attributes are specifications, family, grade and privileges.

Appendix D

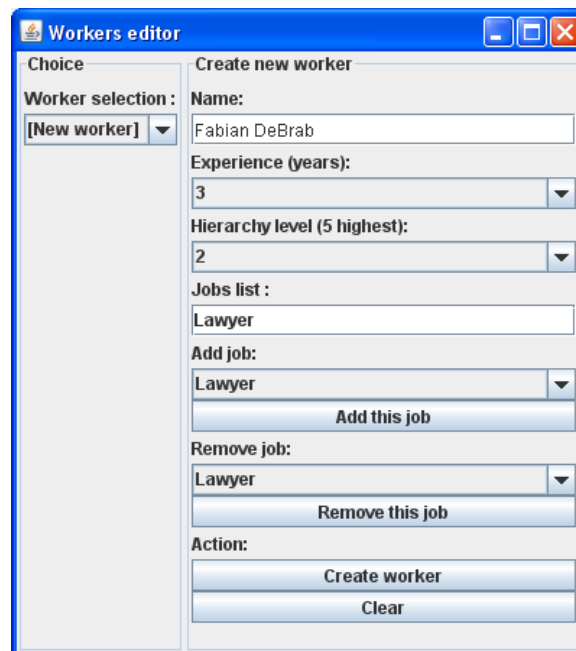


The screenshot shows a window titled "Job handler" with a blue title bar. On the left is a vertical pane labeled "Choice" containing a "Job selection :" dropdown menu with "Lawyer" selected. The main area is titled "Edit existing job" and contains several form fields: "Name:" with a text box containing "Lawyer" and a "[pick up name]" dropdown; "Specifications:" with a text box containing "Defend customers' rights"; "Family:" with a text box containing "Justice"; "Grade:" with an empty text box; "Privileges:" with a text box containing "Costume"; and a "Save changes" button at the bottom.

Figure D-2. Job editor.

UserStereotype Edition

Workers Editor allows to describe user stereotypes (i.e. workers), through the use of information about their experience, hierarchy level and jobs that can be performed.



The screenshot shows a window titled "Workers editor" with a blue title bar. On the left is a vertical pane labeled "Choice" containing a "Worker selection :" dropdown menu with "[New worker]" selected. The main area is titled "Create new worker" and contains several form fields: "Name:" with a text box containing "Fabian DeBrab"; "Experience (years):" with a dropdown menu showing "3"; "Hierarchy level (5 highest):" with a dropdown menu showing "2"; "Jobs list :" with a text box containing "Lawyer"; "Add job:" with a dropdown menu showing "Lawyer" and an "Add this job" button; "Remove job:" with a dropdown menu showing "Lawyer" and a "Remove this job" button; and "Action:" with "Create worker" and "Clear" buttons.

Figure D-3. User stereotype editor.

Workflow Editor

The purpose of this part is to help user to be familiar with the user of the Workflow Editor tool. This tool offers to design a workflow based on a multi-layer methodology. It is based on a workflow model, a process model and a task model.

The methodology used to model a workflow is outside the scope of this manual.

Workflow drawing

The editor behavior is partially made of a visual part close to Microsoft Paint. The graphical elements are the places, transitions, organizational units, job boxes and job groups.

Insertion of one of those elements is made using the left menu, by choosing the element in the menu and secondly clicking in the central zone. If user moves the mouse before relaxing the button, the element will move during this drag & drop action.

The elements are:

- Places – linkable with transitions, can be put inside an organizational unit.
- Transitions – linkable with places, can be put inside an organizational unit.
- Organizational units. May not overlap any other element. A unit may be the container of another unit, and every other element.
- Job box. Can be put inside a job group or directly inside an organizational unit. The range of action of the contained resource is defined by the lower organizational unit in which it is located.
- Job group. The purpose of this element is to put all the job boxes inside a dedicated container. As container it may not overlap elements. A job group may only contain job boxes.

Some operations are available:

- Edition – in this mode user may change the location of graphical elements and have an access to the edition menu on the right of the screen. Every selected element is underlined in yellow, while you can access its related information and options by the right part of the workflow editor. Moving elements is made through drag & drop operation, needing the new element location to be empty. Overlapping existing elements will be forbidden.

Appendix D

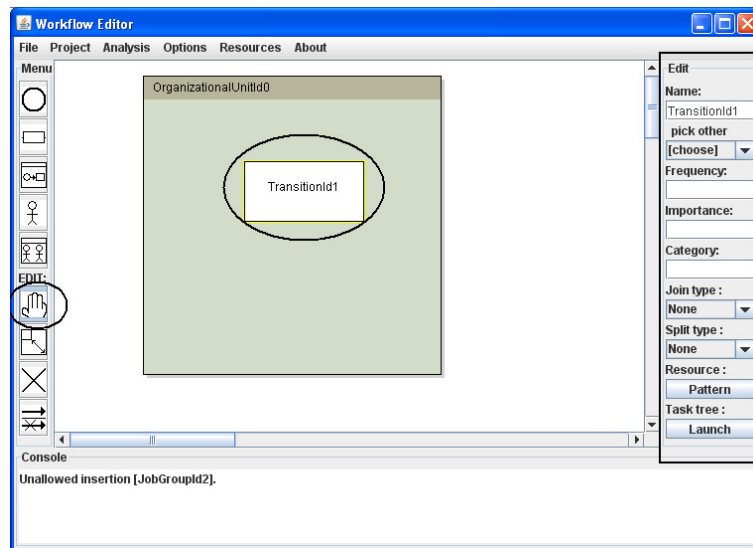


Figure D-4. Workflow editor.

- **Resize** – to use this option, click anywhere inside of an element and then drag the mouse until the pre-visualization rectangle fits the zone you want. You don't have to select an edge of the box, clicking inside of an element is sufficient to start the resizing operation. Keep in mind resizing may be unallowed if the element is a container and some of its contained elements would be left outside of the newly resized container. The same problem will appear when an element overlaps an existing one.

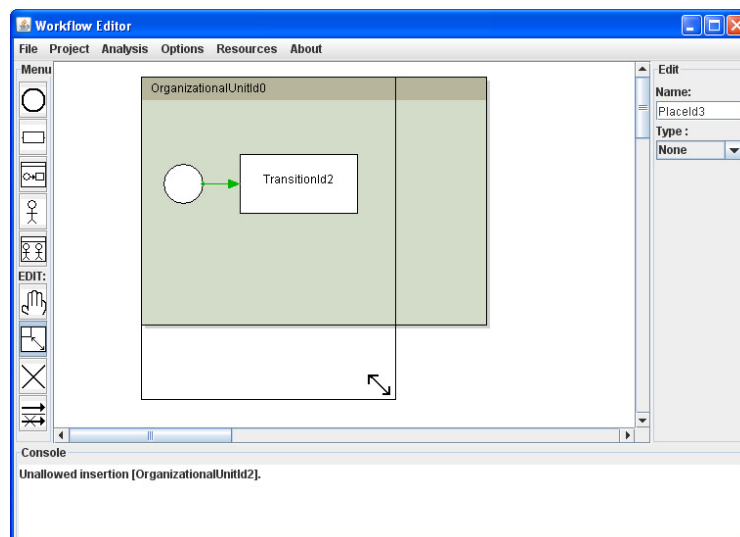


Figure D-5. Workflow editor (2).

Appendix D

- Delete – one click on the left menu button to be in the “delete” mode, another one on the selected element to remove it. When a container element is chosen, all of its subcomponents are also removed. An “undo” option will help the user who made a mistaken delete operation.
- Link/unlink – For linking two elements; first select this option on the left menu. Click on the first element to link, and then on every waypoint you would like to add to the link path. When you make another click on a linkable element (i.e. place or transition) both elements are linked. Follow the same procedure to remove an existing link: at first click on the source element, at last on the destination element.

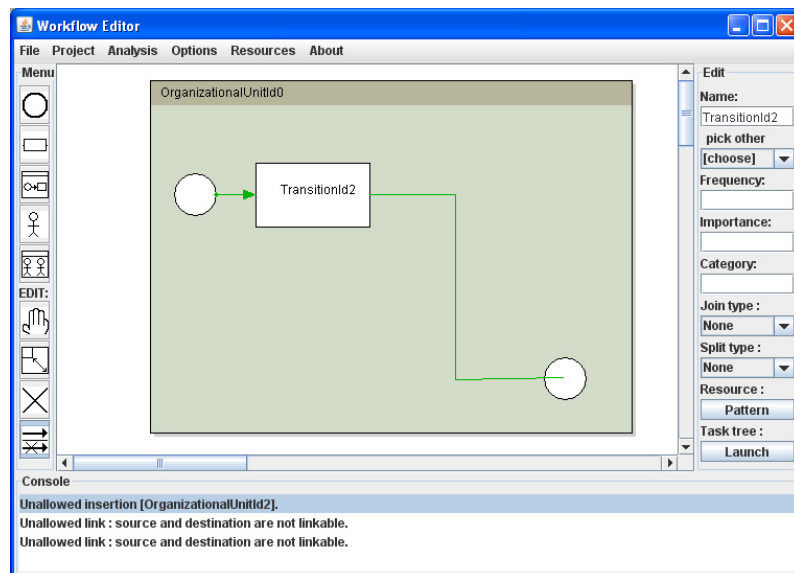


Figure D-6. Workflow editor (3).

Editing graphical Elements

Every kind of graphical elements has a devoted menu on the right of the screen. Its purpose is to allow the workflow designer to edit characteristics of the selected element.

- Places can be of several types: start, normal or final.

Appendix D

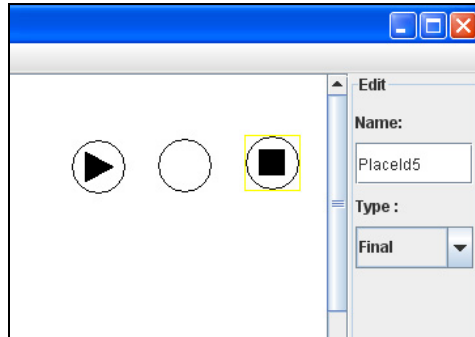


Figure D-7. Type of places.

- Transitions represent atomic processes (i.e. one task). They have some parameters: frequency, importance and category. Two routing patterns may be applied, respectively for the incoming links and outgoing links. Those patterns are sequential (none / applied by default), and, xor, and/xor.

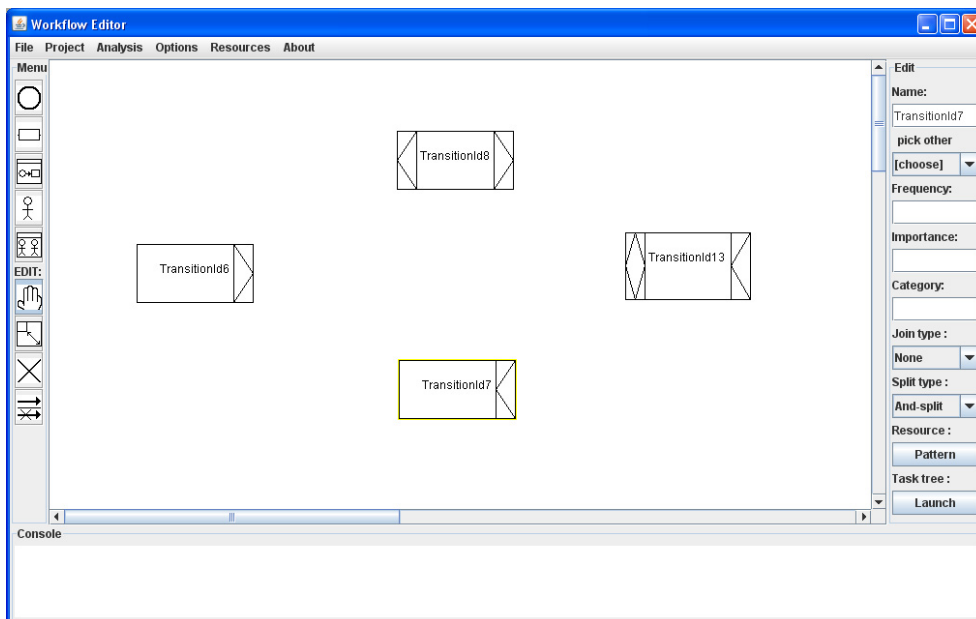


Figure D-8. Routing constructs.

Two more options are available for transitions through the edit menu:

1. The resource pattern tool, used to define what resource will be allowed to be in charge of task. A list of allowed jobs is used to specify that at least one of the chosen job is required to be able to perform

Appendix D

the task. A creation pattern is used to restrict the range of resources that may accomplish the task performance. A distribution type (allocation/offer to single or multiple resources) and a distribution time (i.e. when the resource will be made aware of its work to be done) are also specified.

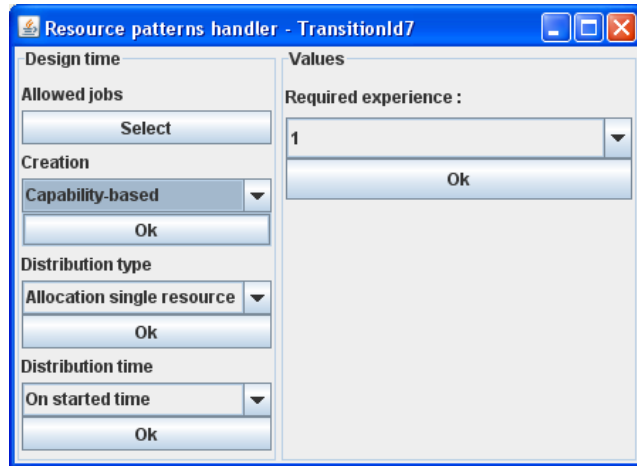


Figure D-9. Resource pattern tool.

2. The concurrent task tree editor. This tool, which is an embedded version of Francisco Montero IdealXML. In order to get a decent presentation the user has to define relationships between tasks. When using the Editor exporting option, a directory will be filled with all the different task trees that have been designed.

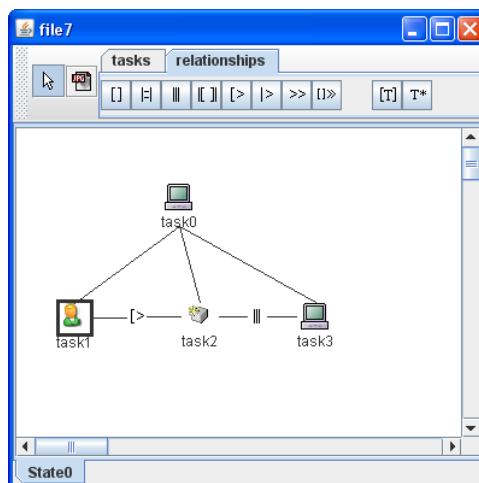


Figure D-10. Task editor.

Appendix D

- Organizational units are used to define where the work takes places inside the organization. The availability of resources depends on the fact a job box belongs to a unit (or one of its sub-units) or not. The different attributes that can be edited are objectives, group type, location, hierarchy level, relationships and rules.

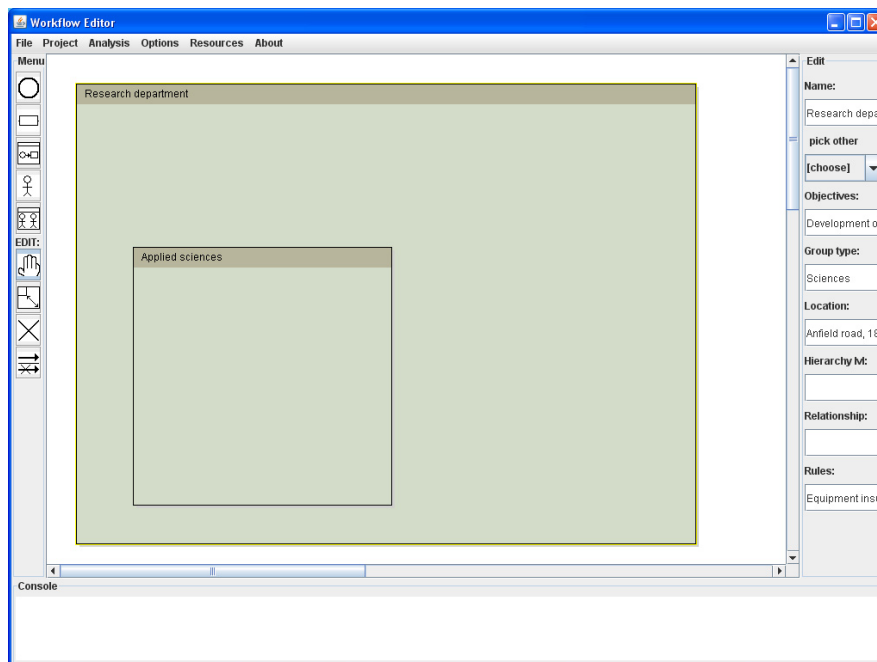


Figure D-11. Representing organizational units.

- Job box is the way to represent available resources able to perform a defined job. For example, four people who are lawyers can be put in a job box containing lawyers. Those resources are available to perform tasks belonging to the same organizational units, and also for every sub-unit of this unit. The edit menu allows defining which job is involved and what resources are put inside the box.

Appendix D

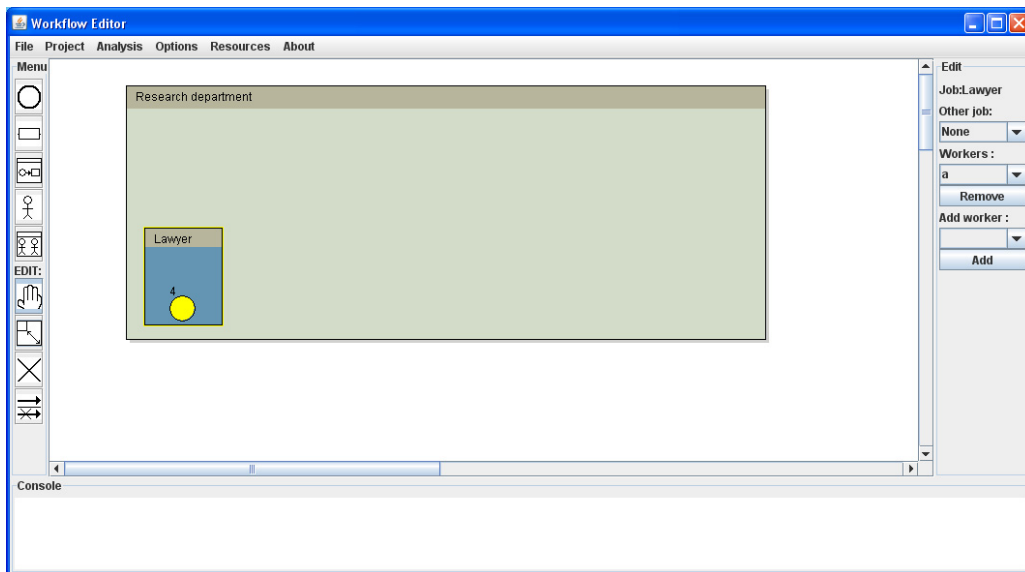


Figure D-12. Representing jobs.

- Job group is a container of job boxes. Its purpose is to have an easy-to-understand view of grouped resources.

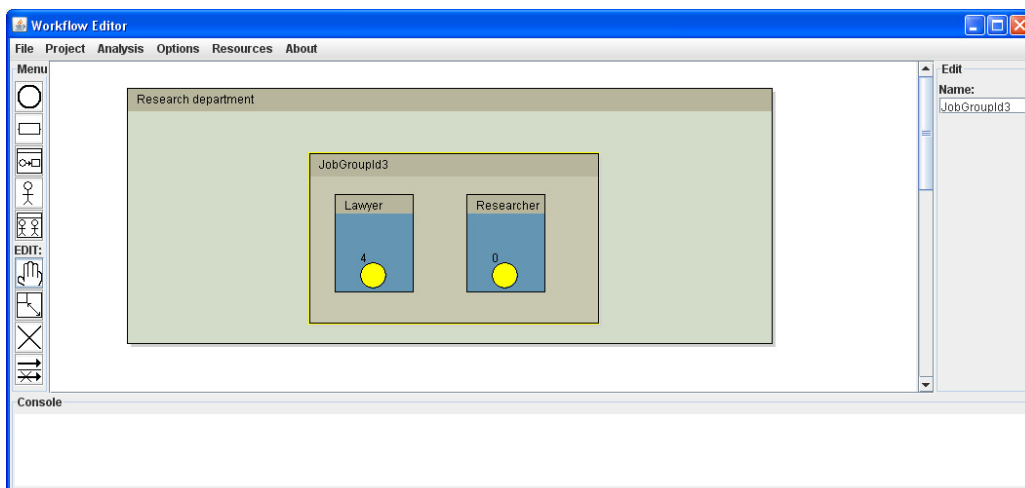


Figure D-13. Representing jobs (2).

North menu

This menu is made of several part, enabling the Workflow Editor options.

Appendix D

- File menu offers the classical options: new, load, save and quit. It also gives the possibility to export the workflow scheme in .jpg image format. The last option is to export in the UsiXML compliant format. It requires the selection of a directory in which the different models will be exported. The concurrent task trees are also exported during this step.

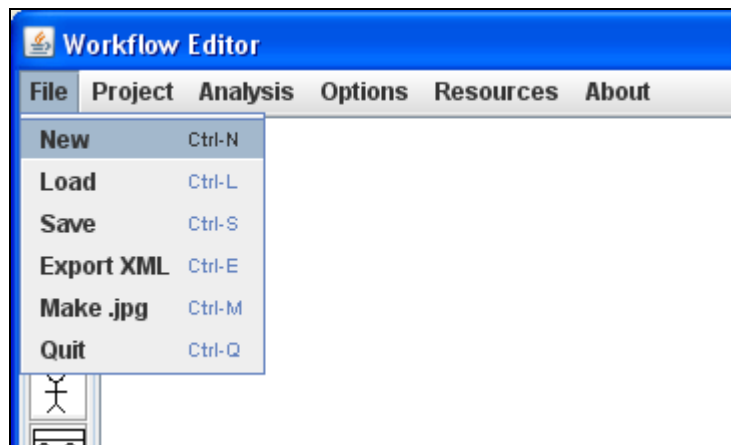


Figure D-14. File menu.

- Project menu has four parts: project information, milestones, identification tool and organizational tree.

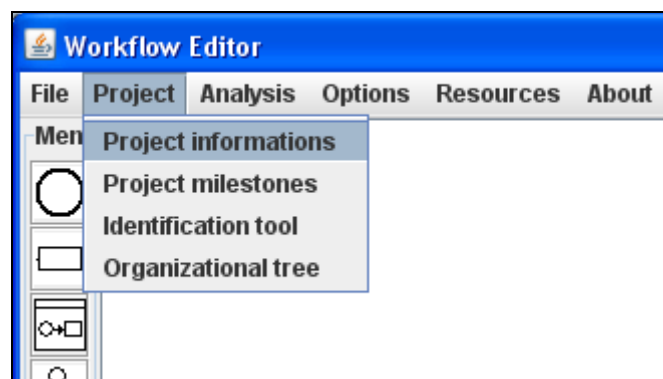


Figure D-15. Project menu.

1. Project information is a small window in which workflow designer may introduce the name of the project, the modeled organization and his own name (the most important field of course).

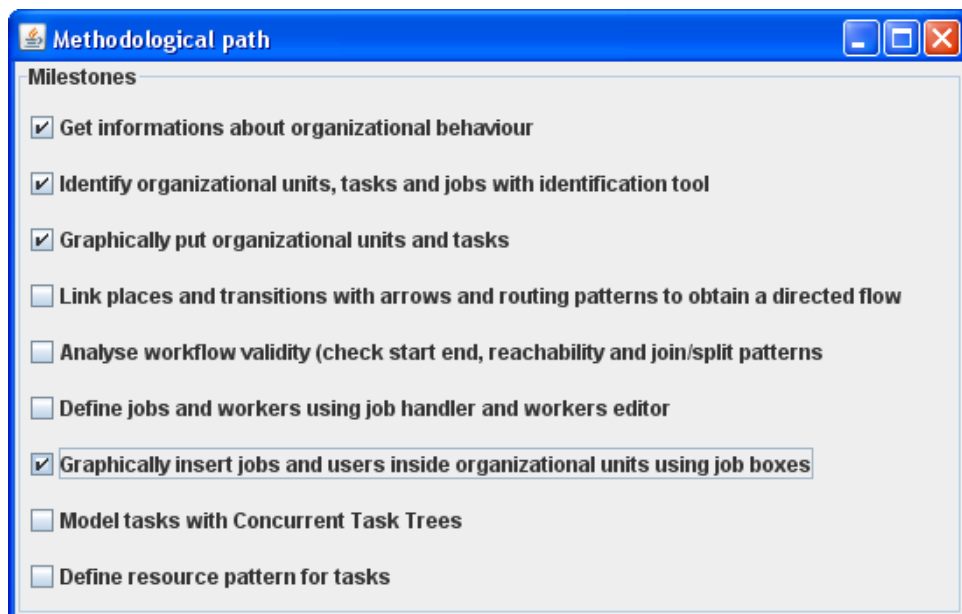
Appendix D



Project	Informations
Project name :	project name
Organization :	organization
Workflow author(s) :	workflow author

Figure D-16. Project information.

2. Milestones are used to get a methodological path reminder. The Workflow Editor user may model the workflow using many different step orderings. It is a practical way to note what you still have to do, telling you where you are (after an holiday break for example).



Methodological path

Milestones

- ☒ Get informations about organizational behaviour
- ☒ Identify organizational units, tasks and jobs with identification tool
- ☒ Graphically put organizational units and tasks
- ☐ Link places and transitions with arrows and routing patterns to obtain a directed flow
- ☐ Analyse workflow validity (check start end, reachability and join/split patterns)
- ☐ Define jobs and workers using job handler and workers editor
- ☒ Graphically insert jobs and users inside organizational units using job boxes
- ☐ Model tasks with Concurrent Task Trees
- ☐ Define resource pattern for tasks

Figure D-17. Milestones.

3. The organizational tree is used to have an overview of the organizational structure in terms of organization units.

Appendix D



Figure D-18. Organizational tree.

- Analysis menu is the way to have a structural test of the implemented Petri net. The first test checks the presence of one start place and at least one end place. The second one verifies if every linkable element (i.e. places and transitions) is reachable through a token that would start from the start place. The last verification ensures a routing pattern is applied to every transition when needed.

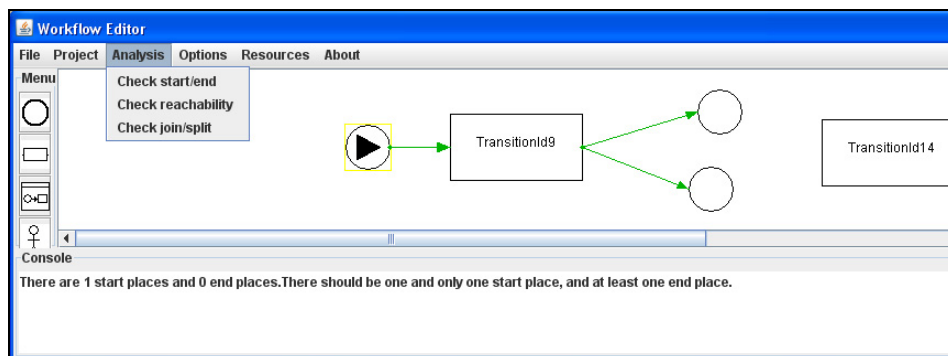


Figure D-19. Analysis menu.

- Options are the way to adapt the size of the drawing part (i.e. the central one). This menu also provides a way to undo the last delete action. This can prove useful in case of accidental deletion of the main organizational unit, containing 100 tasks, 43 jobs and that cost you 6 hours to model.

Appendix D

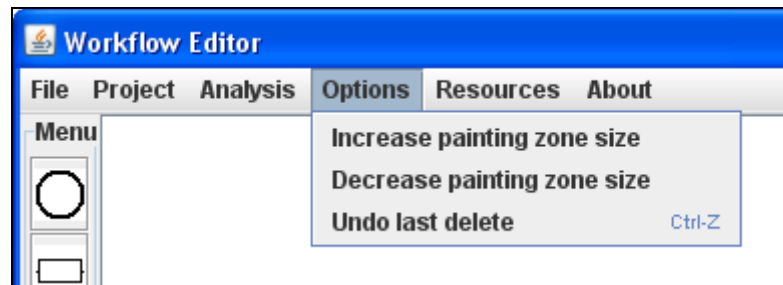


Figure D-20. Options menu.

Last remark

Save frequently, on different files. Some problems may occur while using the program.

Appendix E Workflow user interface patterns

Context

In this appendix, we focus on the assignation of tasks perspective, i.e. the manner in which tasks are advertised and ultimately bound to specific resources for execution.

Developing UIs for WfIS represents new challenges today because the user interaction takes place in two different logical levels synchronously. At the higher level, the workflow manager specifies and monitors the workflow execution. At the lower level, the workflow users are carrying out their allocated tasks whose current status is then communicated to the workflow manager. These results into two UI categories: UIs for the end user (referred to as the *userStereotype*) and UIs for the workflow manager. There is some dependence between these two categories: any change of state of the worker UI should be reflected into the workflow manager UI. For instance, when a worker has finished a task, this information is propagated to the workflow UI. Similarly, when a task should be offered to a single or multiple resources, or delegated, the workflow UI is informed.

Once the conceptual model has been defined, this chapter is aimed at defining a library of UI patterns for workflow information systems that covers both UI categories.

Workflow users

When a workflow is designed, several *userStereotypes* are involved in the process:

- The *Workflow designer* is in charge of the conceptual process and the drawing of the organization. Direct communication is needed to the *workflow manager* of the organizations to have the global view of the processes to be modeled. The results provided are mock-up of the workflow of the processes to be modeled.
- The *Workflow manager* is a person who is responsible for the handling of a whole workflow. Through the modeling the knowledge required to model the organization is captured with meetings with supervisors, workers, etc.
- The *Process manager* is the person who is responsible for a particular process.

Appendix E

- The *end user* is any *userStereotype* that belongs to the organizational model (internal) or is part of other organization (external). It can be the manager, an economist, the lawyer, any valid user declared in the workflow specification.

The above roles could be assigned to different persons or combined into one single person. Apart from the conception of a workflow system, the final use of the system could impact several levels in the hierarchy of the organization.

Task life cycle

As pointed before, the way in which tasks are advertised to resources is essential; therefore we need to consider the different states that task goes through, from creation to termination. There are different approaches to task life cycle [Peti94] [Russ05], from those that contain the basic states to those that contain all the imaginable states.

We propose a life cycle (Figure E-) of the task from the time that it is created to final completion (or failure or cancel).

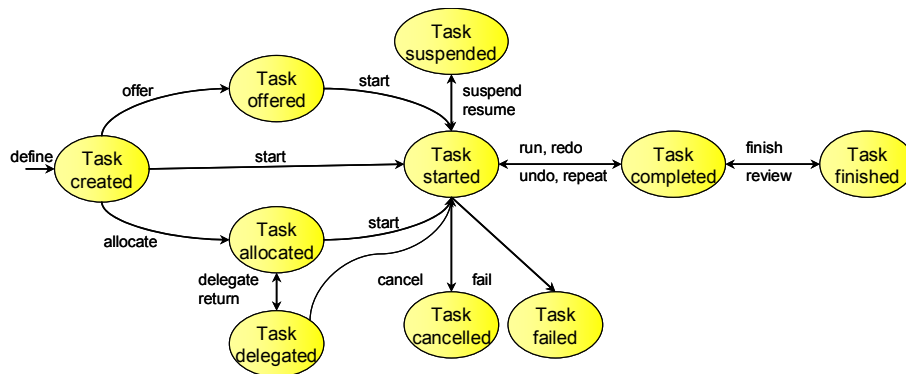


Figure E-1. Task life cycle.

Defining a task consists in specifying its goals, pre-condition, post-conditions, required skills, and required resources. Once a task has been properly defined, it comes into existence in the *created* state. At this point, a task could be effectively *offer* or *allocate*; a task is said to be *started* when the human resource to which the task has been allocated has initiated its execution. A task may be allocated to such resource, but it starts later on. If this allocation is not straightforward, the task can be offered to a single resource or to multiple resources. Once a task is allocated, it could be *delegated* to another resource (e.g., due to unavailability). If the resource which delegated the task wants to receive the results in return, the task is then returned. Otherwise, it can start directly. Subsequent states in the task distribution

are *started*, which indicates that a resource has commenced executing it; *suspended* which denotes that the resource has chosen to cease execution of the task for a period, but does intend to continue working on it at latter time; *failed* which identifies that the task cannot be completed; *cancel* which identifies that the resource, by any reason, will not work on it any further; *completed* which identifies a tasks that has been successfully executed; and *finished* which identifies when the goal is reached.

Defining workflow user interface patterns

A *pattern* is referred to as “the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts” [Rieh96]. Workflow patterns refer specifically to recurrent problems and proven solutions related to the development of workflow information systems. *Workflow resource patterns* have been identified that capture the different manners in which resources are presented and used in workflows [Russ05]. The rationale for identifying these patterns was the need to master the many ways according which work can be distributed.

Model-based UI design [Calv03] [Kris07] [Pate99] is intended to assist designers in obtaining UIs with a formal method, preferably one that is computer-supported; model-based tools have been investigated since the late 1980’s. The goal of these tools is to allow the designer to specify the UI at a level of abstraction that is independent from any implementation. In order to structure the development life cycle of a workflow UI, we are relying on UsiXML [Vand05b].

UsiXML is both the UIDL and a UI engineering methodology articulated on three axes: models and their specification language, method, and tools that support the method based on the underlying models. An overview of this methodology is presented in Appendix B, for the complete definition we refer to [www.usixml.org].

We adopted the following steps for defining the Workflow User Interface Patterns (WUIPs):

- *UI pattern definition*: from each workflow resource pattern a WUIP is created and defined.
- *Incorporation in the model-driven engineering method*: for each initial pattern definition resulting from the previous step, a task model has been specified using CTT notation [Pate99] in order to depict the pattern.
- *Final WUIPs*: from the task models resulting from the previous steps, abstract UIs and, consequently, concrete UIs have been defined in terms of the UIDL

Appendix E

(here, UsiXML) so as to form corresponding abstract and concrete UI models.

Applying the above methodology resulted in 42 WUIPs. We give below only a snapshot of some of these patterns for facilitating the understanding.

<i>Name:</i>	Direct allocation
<i>Identifier:</i>	R-DA
<i>Synopsis:</i>	The ability to specify at design time the identity of the resource that will execute a task
<i>Strengths:</i>	To prevent the problem of non-suitable allocation
<i>Weakness:</i>	No opportunity to change the resource if he is not available to perform the task
<i>Opportunities:</i>	To ensure task is routed to specific resource
<i>Problem:</i>	This pattern effectively defines a static binding of tasks to a single resource
<i>Solution:</i>	Probably the use of deadline and escalation mechanisms when the resource becomes overload and cannot deal with his assigned workload in a reasonable timeframe
<i>Example:</i>	“Ask reviewers preferences” task must only be undertaken by “Joshua Brown”

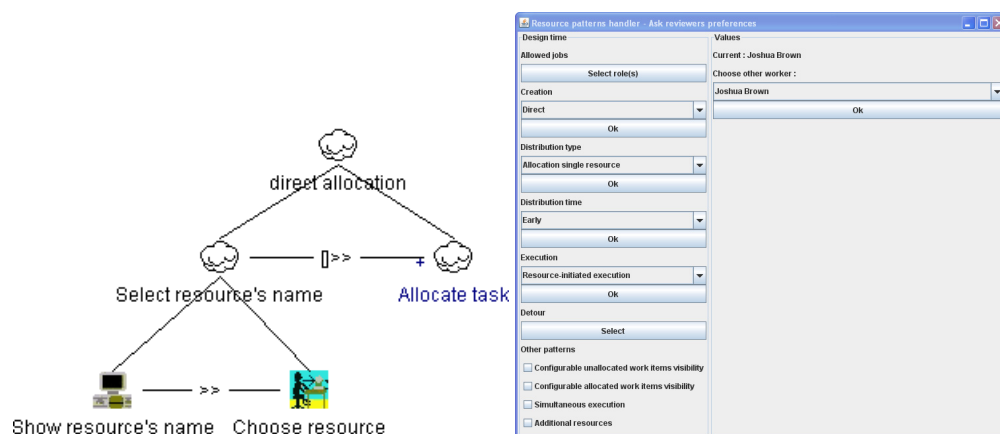


Figure E-2. Direct allocation pattern.

Appendix E

<i>Name:</i>	Deferred allocation
<i>Identifier:</i>	R-FBA
<i>Alias:</i>	Put-off
<i>Synopsis:</i>	The ability to defer specifying the identity of the resource that will execute a task until runtime
<i>Strengths:</i>	To defer the need to identify the resource for a specific task until run time
<i>Weakness:</i>	The identity of the resource in charge of the task is unknown
<i>Opportunities:</i>	The resource identity can be changed dynamically during the workflow execution to ensure that the resource is the most appropriate to develop the task
<i>Problem:</i>	The identification the resource for a specific task in runtime
<i>Solution:</i>	Nominate a data field from which the identity of the resource to which a task should be routed can be determined at runtime. It is important to ensure that the data field contains a valid resource name
<i>Example:</i>	During the execution of “Pay a payment order”, instances of the task will be executed by the resource named in the next-resource field

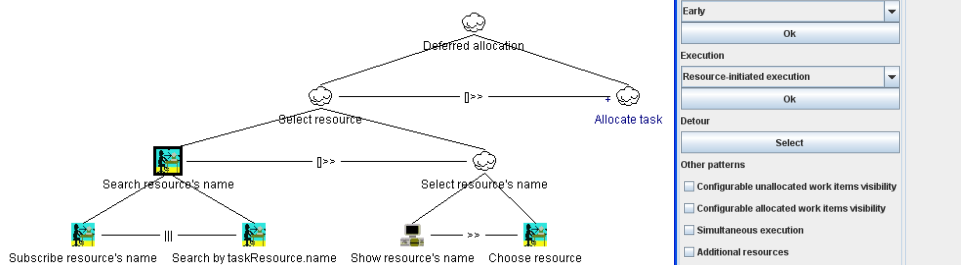


Figure E-3. Deferred allocation pattern.

Appendix E

<i>Name:</i>	Authorization
<i>Identifier:</i>	R-RA
<i>Alias:</i>	Permission
<i>Synopsis:</i>	The ability to specify the range of resources that are authorized to execute a task
<i>Strengths:</i>	To ensure that unexpected events (delegation or reallocation) that may arise during the workflow execution, do not lead to unexpected resources being able to undertake tasks
<i>Opportunities:</i>	To define a security framework over a workflow implementation that is independent of the way in which tasks are actually routed in runtime
<i>Weakness:</i>	The range of resources that are authorized to execute a task is limited
<i>Solution:</i>	Where a resource is not able to execute a task, the workflow manager can do a direct allocation
<i>Example:</i>	Only the “Social worker” is authorized to execute instances of the “Apply the Final Interview” task

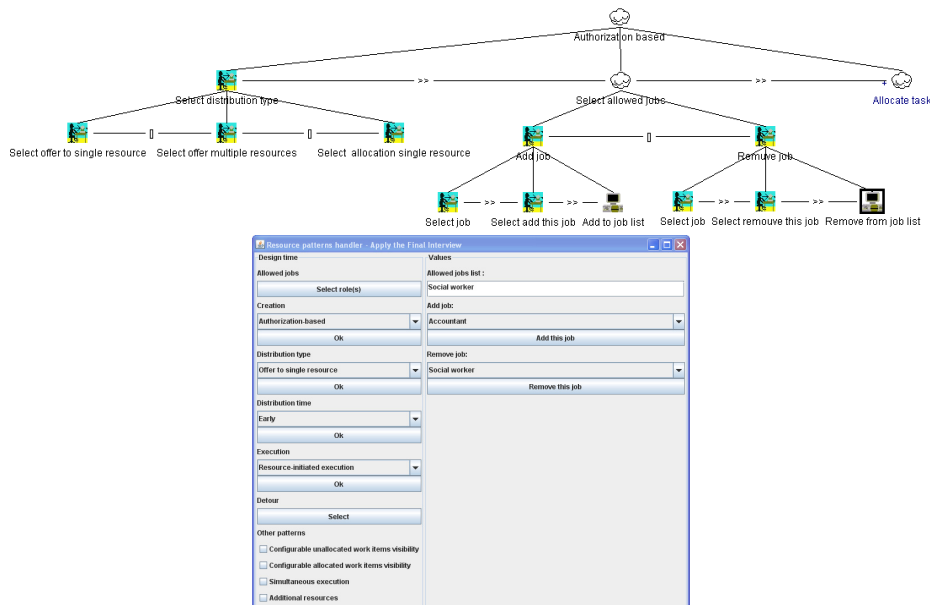


Figure E-4. Authorization pattern.

Appendix E

<i>Name:</i>	Separation of duties
<i>Identifier:</i>	R-SOD
<i>Alias:</i>	Split
<i>Synopsis:</i>	The ability to specify that two tasks (task1, task2) must be allocated to different resources in a given process
<i>Strengths:</i>	This ensures that a task cannot be executed by the same resource that executed another task within the same process
<i>Opportunities:</i>	This allows for the enforcement of audit controls within the execution of a process.
<i>Weakness:</i>	Availability of resources
<i>Solution:</i>	Look for a resource for task 1 and assign it, look a resource for task 2 (resource 1 $\neq$ resource 2) and assign it
<i>Example:</i>	“Submit paper” task must be allocated to a different resource to that which executed the “Prepare final submission” task

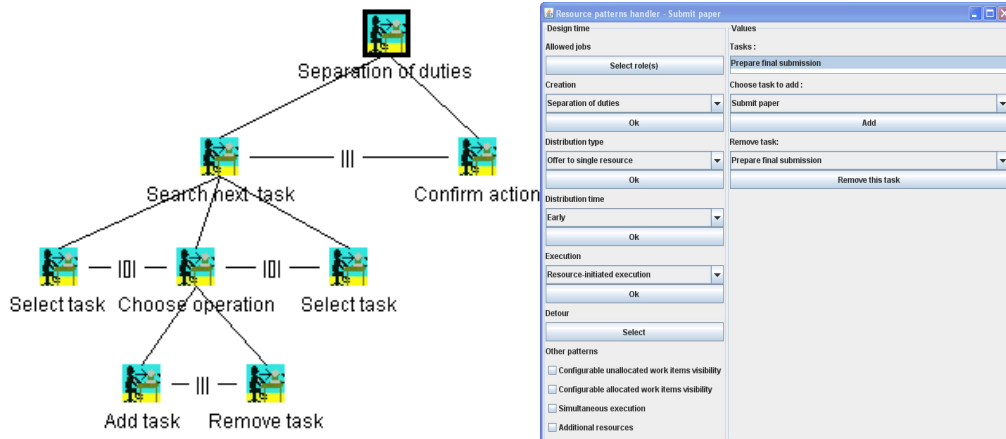


Figure E-5. Separation of duties pattern.

<i>Name:</i>	Case handling
<i>Identifier:</i>	R-CH
<i>Synopsis:</i>	The ability to allocate the tasks within a given process to the same resource

Appendix E

- Strengths:** Tasks in a process can be allocated exclusively to the same resource which must complete them all to ensure continuity
- Opportunities:** R-CH can serve as a guide to know how tasks within a given process should be routed with an initial resource being identified as having responsibility for all tasks
- Weakness:** It is possible for a resource to have an overload of work
- Solution:** One consideration is the delegation option
- Example:** All task in “Make doors” are allocated to the same “carpenter”

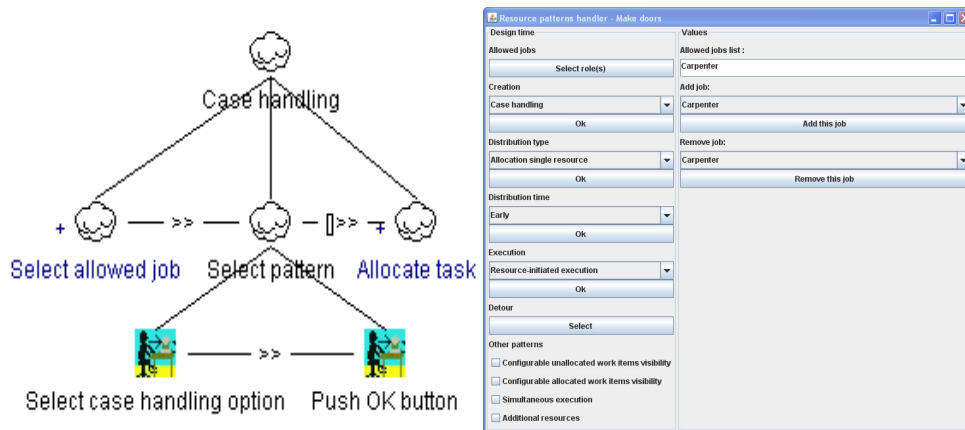


Figure E-6. Case handling pattern.

Note: After the selection of job, it could be possible to assign tasks to the same resource by name, in this case a combination with direct allocation pattern. This combination could be applied in future. Task “Select allowed job” is presented in “authorization” pattern.

- Name:** Retain familiar
- Identifier:** R-RF
- Alias:** Preservation
- Synopsis:** Where several resources are available to undertake a task, the ability to allocate a task within a given process to the same resource that undertook a preceding task.
- Strengths:** It only comes into effect when there are multiple resources available to undertake a given task and where this occurs; it favors the

Appendix E

allocation on the task to the resource that undertook a previous task in the process

Opportunities: As the resource is already aware of the details of the previous tasks, it saves familiarization time at the commencement of the next task

Weakness: Availability of the resource

Solution: Advertise the allocation of tasks to resource on time

Example: In a hospital where are several “Pediatrician” to undertake “Consult newborn baby” task, it should be allocated to the same “pediatrician” who “Attend in the childbirth” task

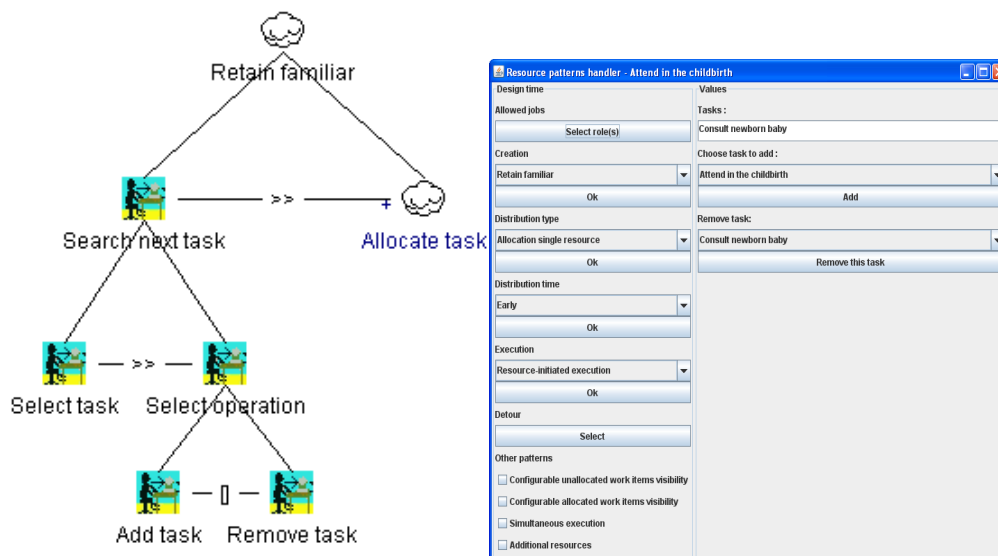


Figure E-7. Retain familiar pattern.

Name: Capability-based allocation

Identifier: R-CBA

Alias: Competence

Synopsis: The ability to offer or allocate instances of a task to resources based on specific capabilities that they possess

Strengths: This allocation allows selecting the most appropriate resource to undertake a given task

Appendix E

- Opportunities:* To ensure that a task is developed by the appropriate resource
- Weakness:* It is possible to identify more than one possible resource to which a task may be assigned or no to find any resource
- Solution:* Them can avoided through more precise definition of capability functions
- Example:* Instances of the “Teach to fly airplanes” task should be allocated to the “Pilot” with 5 years experience in teaching, more than 5,000 flight hours

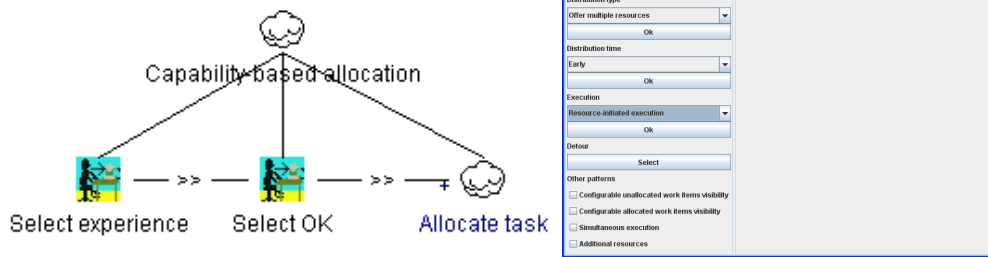


Figure E-8. Capability-based allocation pattern.

Note: there are other capability functions to be added in the future

- Name:* History-based allocation
- Identifier:* R-HBA
- Alias:* Record
- Synopsis:* The ability to offer or allocate tasks to resources on the basis of their previous execution history
- Strengths:* It involves the use of information on the previous execution history of resources when determining which of them a task should be offered or allocated to
- Opportunities:* This is an analogue to common human record; it considers factors such as who has the least numbers of failures
- Weakness:* The difficulty is the necessity of register additional information about the resource and the task.

Appendix E

Solution: To gather and manage the least amount of execution history for each resource that is required to facilitate the chosen work distribution strategy

Example: Allocate the “Assemble the microchips” task to the “Computer hardware engineer” who has successfully completed the most of these tasks

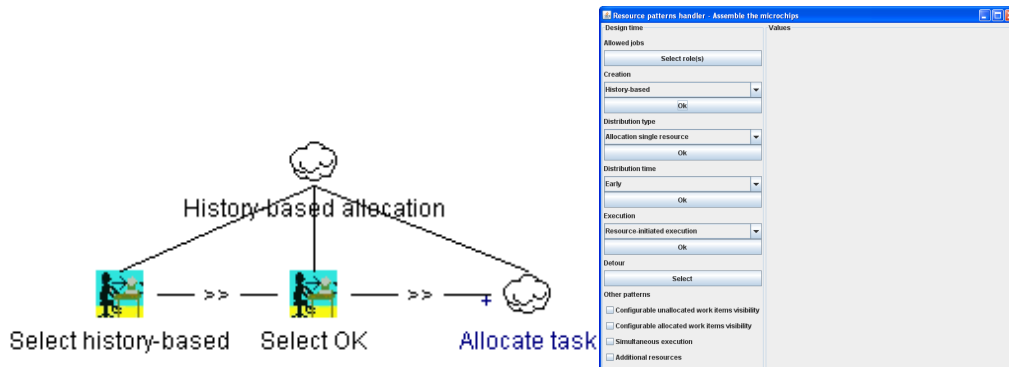


Figure E-9. History-based allocation pattern.

Name: Hierarchy level-based

Identifier: R-HLB

Alias: Position-based

Synopsis: The ability to offer or allocate instances of a task to resources based on their hierarchic level within the organization and/or their relationship with other resources

Strengths: It helps when the allocation or offer of tasks must be made in the context of the organizational structure and the relative position of individual resources both in the overall hierarchy and also in terms of their relationships with other resources

Opportunities: The provide a flexible and realistic basis for managing work in organizational setting

Weakness: The possible confusion of this pattern with authorization pattern

Solution: With this pattern the allocation of tasks is based on a hierarchical organizational model, in the case of authorization pattern, it is based on a permission to develop a task that is granted to the resource.

Appendix E

Example: “Reduce wage bill” task is allocated to a “Financier” with has a 5 level, i.e. the “Financial Manager”

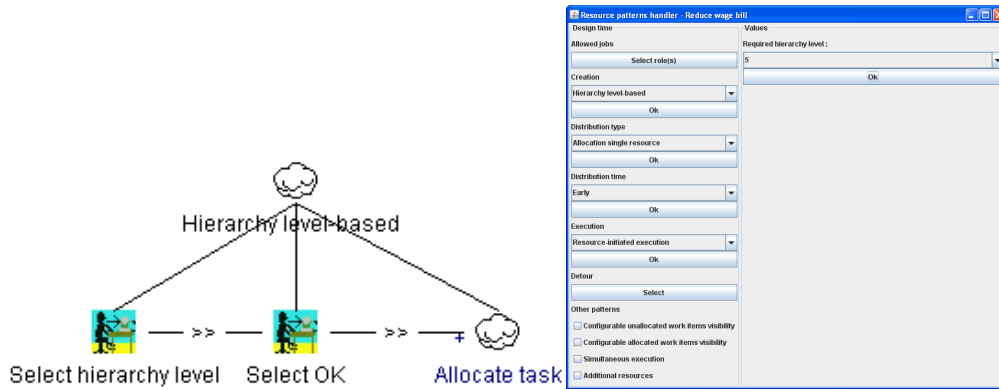


Figure E-10. Hierarchy level-based pattern.

Name: Distribution by offer single-resource

Identifier: R-DBOS

Alias: Individual tender

Synopsis: The ability to offer a task to a selected individual resource

Strengths: To offer a task to a single distinct resource for potential execution

Opportunities: The system should notify to the resource that task exists that he may be wish to undertake

Weakness: The possibility that the resource is not committed to execution the task

Solution: Notify to other resources of the potential task exists

Example: “Install the stopcock” task is offered to a selected “Plumber”

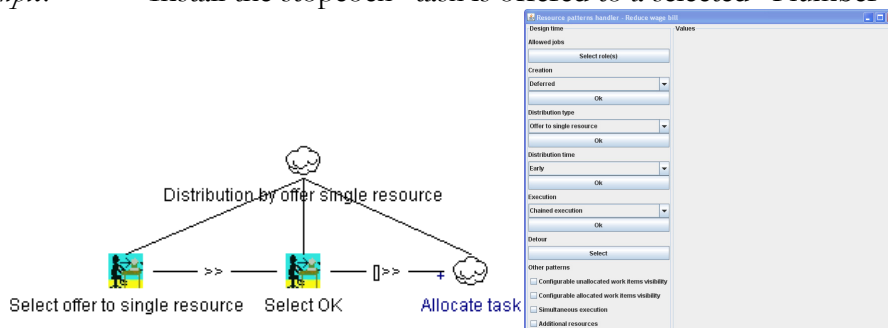


Figure E-11. Distribution by offer single-resource pattern.

Appendix E

<i>Name:</i>	Distribution by offer multiple-resources
<i>Identifier:</i>	R-DBOM
<i>Alias:</i>	Multiple tender
<i>Synopsis:</i>	The ability to offer a task to a group of selected resources
<i>Strengths:</i>	It provides a means of advising a suitably qualified group of resources that a task exists but leaves the onus with them as to who actually commits to undertaking the activity
<i>Opportunities:</i>	Support the notion of work groups and allow tasks to be allocated to them
<i>Weakness:</i>	It is possible that a task that has been accepted by a resource remains visible for execution to the rest of group
<i>Solution:</i>	There are two possible solutions, when a multiply-offered task is accepted by one of the resources to which is offered, it is removed from the agenda of all other resources, or, it could be remained on the group agenda but is not able to be selected for execution by other resources
<i>Example:</i>	“Prepare athlete” task is offered to multiple “athletic trainer”

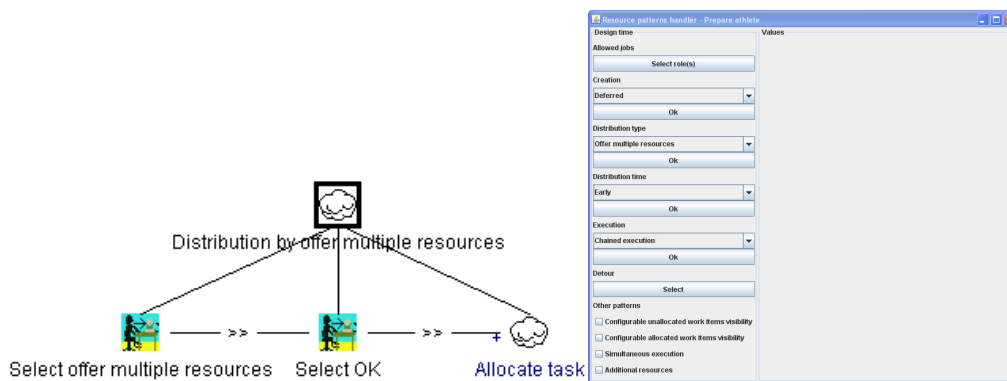


Figure E-12. Distribution by offer multiple-resources pattern.

<i>Name:</i>	Distribution by allocation single-resource
<i>Identifier:</i>	R-DBAS
<i>Alias:</i>	Assignation

Appendix E

<i>Synopsis:</i>	The ability to directly allocate a task to a specific resource for execution
<i>Strengths:</i>	The directly assignation of a task to a resource without first offering it to other resources or querying whether the resource will undertake it
<i>Opportunities:</i>	The main focus is on maximizing work throughput by keeping the resource busy
<i>Weakness:</i>	There is the possibility that a resource to which has been assigned a task does not want to execute it
<i>Solution:</i>	The option that a task could be assigned to other resource
<i>Example:</i>	“Develop the final test” task should be allocated to the “Chemical engineer”

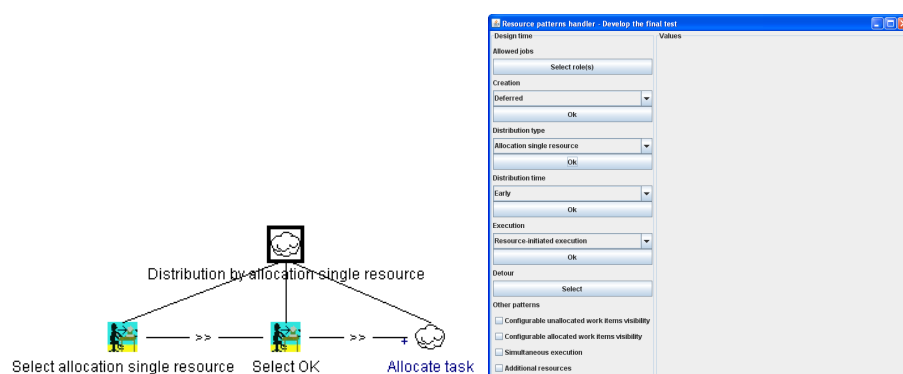


Figure E-13. Distribution by allocation single-resource pattern.

<i>Name:</i>	Early distribution
<i>Identifier:</i>	R-ED
<i>Alias:</i>	Before start distribution
<i>Synopsis:</i>	The ability to advertise and potentially allocate tasks to resources ahead of the moment at which the tasks are enabled
<i>Strengths:</i>	It provides means of notifying resources of upcoming tasks ahead of the time at which they need to be executed

Appendix E

- Opportunities:* It is useful where resources are able to provide some form of booking tasks indicating they will execute and complete them at some future time
- Weakness:* None observed
- Solution:* Not apply
- Example:* “Organize the annual college reunion” task is allocated to the “Secretary” of ex-students department at least three months ahead of the time that it will commence

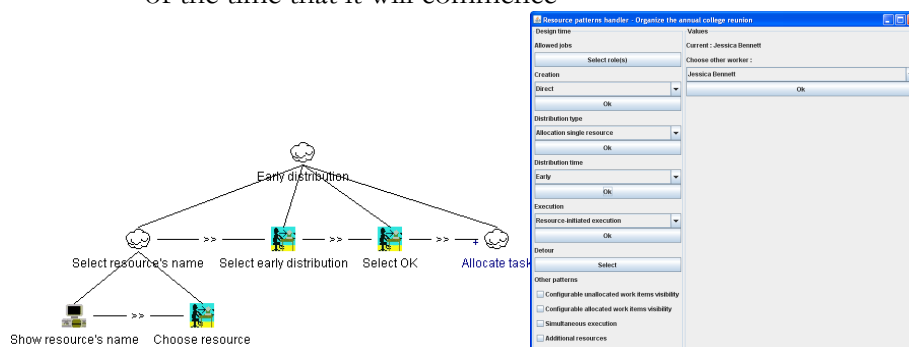


Figure E-14. Early distribution pattern.

- Name:* Distribution on enablement
- Identifier:* R-DE
- Alias:* On started time
- Synopsis:* The ability to advertise and allocate tasks to resources at the moment they are enabled for execution
- Strengths:* It is an effective mechanism for tasks distribution in a workflow system by placing them on the resources' agendas as offer or allocation
- Opportunities:* The enablement of a task serves as the trigger for the workflow engine to make it available to resources for execution
- Weakness:* None observed
- Solution:* Not apply
- Example:* “Prepare the payroll” task is allocated to the “Accountant assistant” at the time it is required to commence

Appendix E

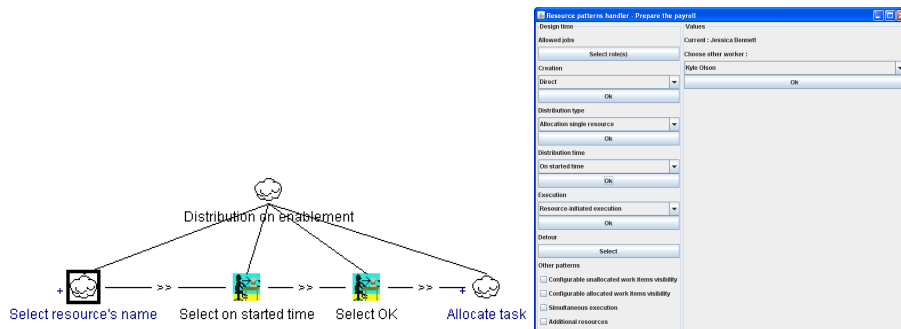


Figure E-15. Distribution on enablement pattern.

- Name:* Late distribution
- Identifier:* R-LD
- Alias:* After start distribution
- Synopsis:* The ability to advertise and allocate tasks to resources after the task has been enabled
- Strengths:* This pattern is undertaken with the aim of preventing resources from becoming overwhelmed by the apparent workload even though they may not be required to undertake all of it themselves
- Opportunities:* It is possible to reduce the current volume of work in progress within a workflow
- Weakness:* None observed
- Solution:* Not apply
- Example:* “Pack milk boxes” task is allocated to a Packer after they have been sealed to be delivered

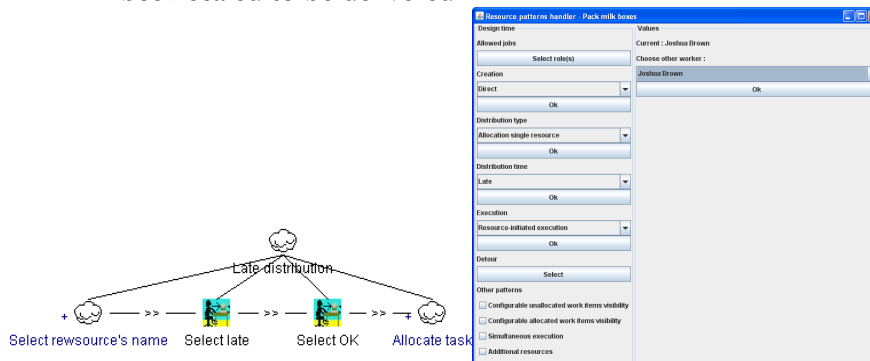


Figure E-16. Late distribution pattern.

Appendix E

<i>Name:</i>	Random allocation
<i>Identifier:</i>	R-RMA
<i>Synopsis:</i>	The ability to offer or allocate tasks to suitable resources on a random basis
<i>Strengths:</i>	It provides a non-deterministic mechanism for allocating tasks to resources
<i>Opportunities:</i>	Random allocation is used to reduce bias in trials
<i>Weakness:</i>	It is possible that a task is not allocated to a suitable resource
<i>Solution:</i>	This pattern can be used in combination with other patterns to ensure the correct allocation of tasks
<i>Example:</i>	“Give maintenance to sewing machine” task is allocated to a “Sewing machine operator” on a random basis

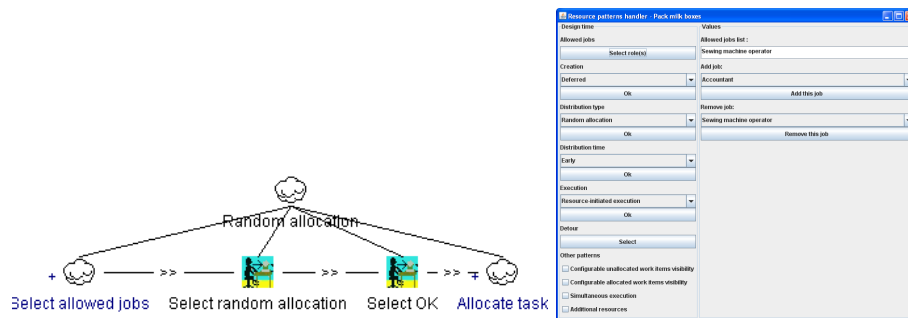


Figure E-17. Random allocation pattern.

<i>Name:</i>	Round robin allocation
<i>Identifier:</i>	R-RRA
<i>Synopsis:</i>	The ability to allocate a task to available resources on a cyclic basis
<i>Strengths:</i>	It assigns tasks to each resource in equal portions and in order
<i>Opportunities:</i>	It provides a means to ensuring that all resources are allocated the same number of tasks
<i>Weakness:</i>	It requires details of each resource allocations to be maintained so that a decision can be made as to which resource should be used when the next allocation decision is made

Appendix E

Solution: There are two possibilities: to use an external program to manage the allocation decision and keep track of the previous allocations, or to combine with history-based allocation pattern

Example: Tasks corresponding to “Assemble portable computer” process are allocated to each available “Computer hardware engineer” on a cyclic basis

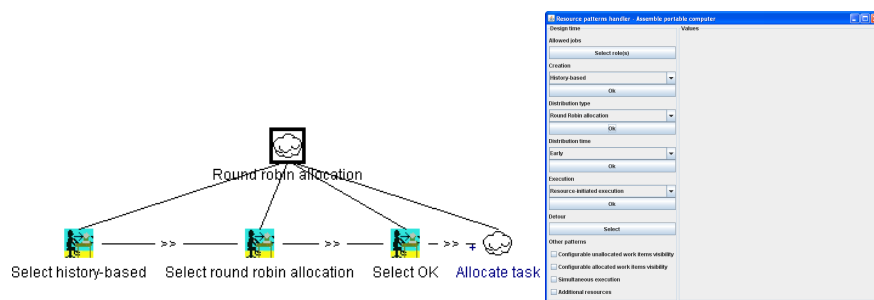


Figure E-18. Round robin allocation pattern.

Name: Shortest queue

Identifier: R-SHQ

Alias: Little work

Synopsis: The ability to allocate a task to the resource that has the least number of tasks allocated to it

Strengths: Tasks are allocated to the resources that is able to undertake them in the shortest possible timeframe

Opportunities: Possibility to execute task in the shortest time due that the resource has few tasks to doing

Weakness: None observed

Solution: Not apply

Example: “Show new apartment” task is allocated to the “Sales representative” who has the least number of tasks to doing

Appendix E

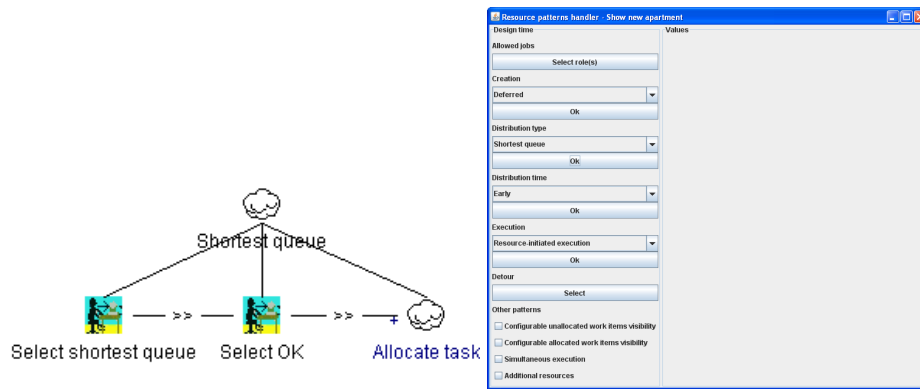


Figure E-19. Shortest queue pattern.

Summary

This appendix introduced a collection of user interface design patterns that are particularly applicable to user interfaces of workflow information systems. In order to obtain these patterns, the task life cycle gave rise to a series of pertaining transitions in the evolution, thus leading to workflow patterns. Each pattern can be selected in a workflow model editor so as to automatically generate the specifications for both the workflow model (in this way, it is no longer needed to redraw the definition of the pattern in terms of places and transitions) and the user interface model (in this way, it is no longer needed to specify again the UI supporting the workflow pattern). Of course, these specifications can be edited before producing the final system.

Even more, we describe each pattern with an *identifier*, an *alias*, a *synopsis*, its *strengths*, its *opportunities*, its *weakness*, a possible *solution*, and an *example*.

Appendix E

Appendix F Workflow nets

In [vand98] Petri net theory is applied to process modeling and workflow nets are introduced (WF-net). A WF-net is a Petri net which has a unique source place (i) and a unique sink place (o). This corresponds to the fact that any case handled by the process description is created if it enters the WFMS and is deleted once it is completely handled by the WFMS. In such a net, a task is modeled by a transition and intermediate states are modeled by places. A token in the source place i corresponds to a case which needs to be handled, a token in the sink place o corresponds to a case that has been handled. The process state is defined by the marking. In addition, a WF-net requires all nodes (i.e. transitions and places) to be on some path from i to o . This ensures that every task (transition) or condition (place) contributes to the processing of cases.

Definition (WF-net) A Petri net $PN = (P, T, F)$ is a WF-net (WorkFlow net) if and only if:

- (i) PN has two special places: i and o . Place i is a source place: $\bullet i = \emptyset$. Place o is a sink place: $o \bullet = \emptyset$.
- (ii) If we add a transition t^* to PN which connects place o with i (i.e. $\bullet t^* = \{o\}$ and $t^* \bullet = \{i\}$), then the resulting Petri net is strongly connected.

Places in the set P correspond to conditions; transitions in the set T correspond to tasks. Note that the requirements stated in Definition are *minimal* requirements.

Even if these requirements are satisfied it is still possible to define a workflow process definition with potential deadlocks and/or livelocks.

Tokens in a WF-net represent the *workflow state* of a single case. The workflow state contains partial information about the state of a case. In addition the case has *workflow attributes* and *application data*.

Routing constructs

Tasks may be optional, i.e. there may be tasks which only need to be carried out for a number of cases. The order in which tasks are performed may also vary from case to case. By routing a case along a number of tasks, it is possible to determine which tasks need to be carried out and in what order. There are four basic constructions for routing.

a) Sequential routing

There is a sequential performance of tasks when these have to be carried out one after another. If two tasks need to be carried out sequentially, there is usually a clear interdependence between them. For example, the result of the first is required in order to perform the second. In a Petri net, this form of routing is modeled by linking the two tasks using a place. Figure F-1 shows an example of sequential routing.

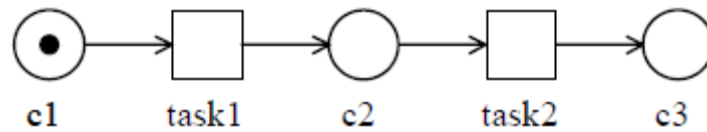


Figure F-1. Sequential routing [vand02].

b) Parallel routing

If more than one task can be carried out at the same time or in any order, then we refer to parallel routing. If we confine ourselves to the situation with two tasks, *task1* and *task2*, then there are three possibilities. Both tasks can be performed simultaneously; *task1* can be carried out first, then *task2*; or *task2* can be first, followed by *task1*. Figure F-2 illustrates how we can model this situation using a Petri net.

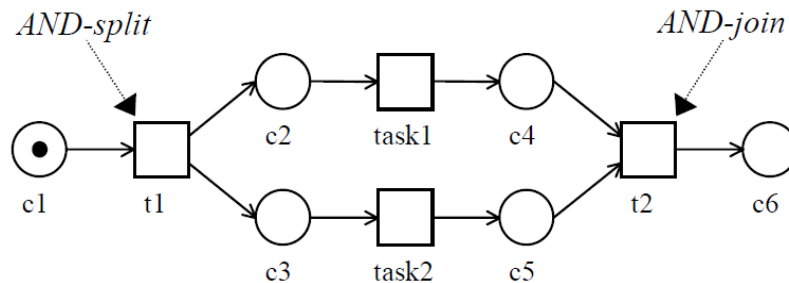


Figure F-2. Parallel routing [vand02].

c) Selective routing

It is used to allow for a routing which may vary between cases. In this way, the routing of a case may depend on the workflow attributes of a case, the behavior of the environment, or the workload of the organization. To model a choice between two or more alternatives, two building blocks are used: (1) the OR-split and (2) the OR-join (in both cases an exclusive OR). An OR-split can be modeled by a place with multiple outgoing arcs; an OR-join is modeled by a place with multiple incoming arcs. Based upon the case attributes, transition *t1* in Figure F-3 produces a token for either *c2* or *c3* (but not for both).

Appendix F

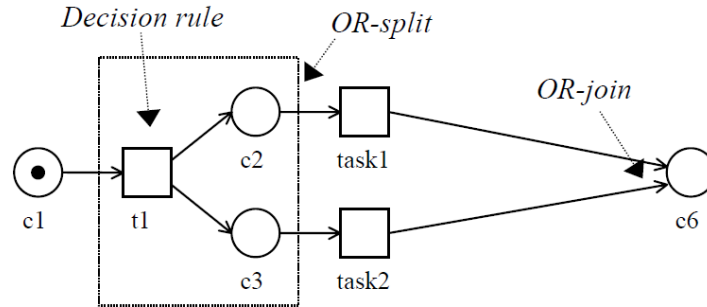


Figure F-3. Selective routing (1) [vand02].

In Figure F-3, the number of tokens produced in each of the output places of $t1$ is variable (0 or 1). A choice is made based upon the value (case attributes) of the token in $c1$ and the decision rule in $t1$. However, we can also produce this choice by using two transitions containing the appropriate preconditions. Recall that a precondition is based on the colors of the tokens to be consumed and acts like a transition guard. Figure F-4 shows how this is possible.

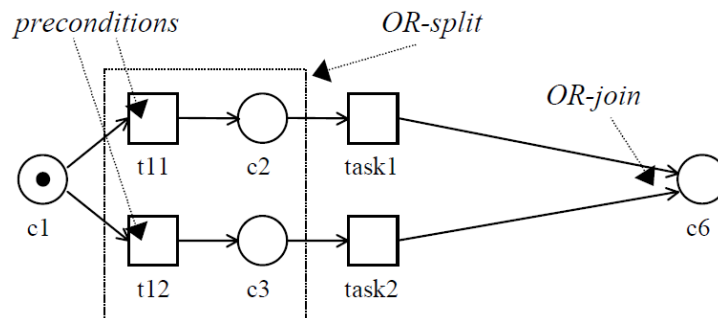


Figure F-4. Selective routing (2) [vand02].

d) Iterative routing

Ideally, a task will be performed only once per case. In certain situations, however, it is necessary to apply *iterative routing*, for example, when a certain task needs to be repeated until the results of a subsequent test prove positive. Figure F-5 shows how we can model iterative routing.

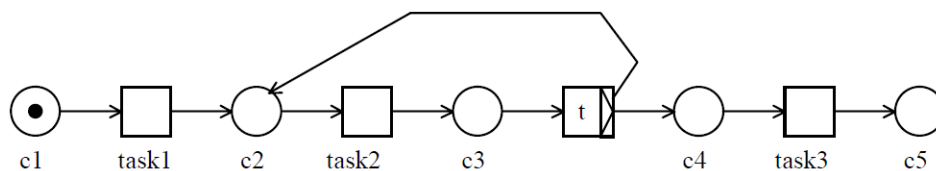


Figure F-5. Iterative routing [vand02].

Appendix F

Appendix G Abstract user interface events

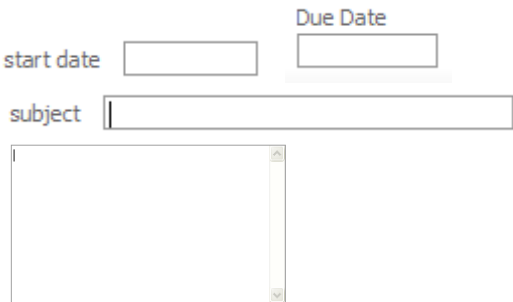
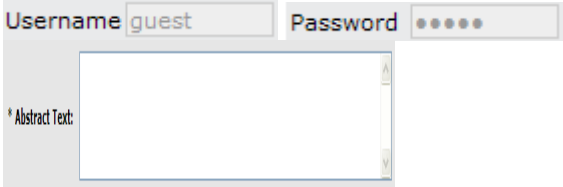
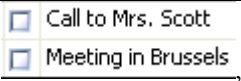
Mappings between AUI events and their concretization in terms of CUI events used in ECA rules, a complete list of examples is presented in this appendix. ECA rules are used indifferently at both the AUI and the CUI levels. This table shows that for a same abstract event in a dialog model for AUI, several different mappings can be ensured with concrete events in a dialog model for CUI depending on the context of use, particularly the various interaction modalities, in this case: graphical and vocal ([Stan08] for more details on vocal examples). Although that exist a lot of CUI events, we focus on those that are frequently used. Also, notice that certain facets or combinations of them are not applicable as they do not make any sense or are contradicted, they are:

■ Activate an AIC with navigation facet. In this case is irrelevant to exemplify.

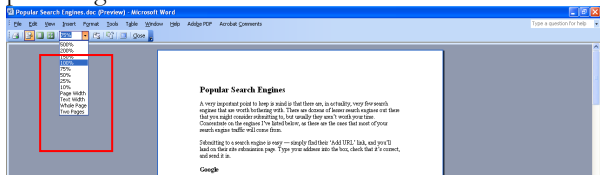
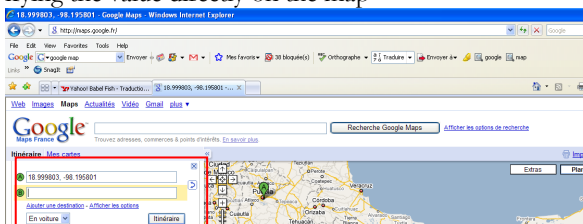
Activate an AIC with control facet & activate an AIC with navigation + control facet. Theoretically possible but violate usability guidelines “every control should produce a feedback”.

Appendix G

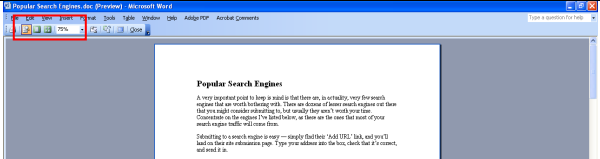
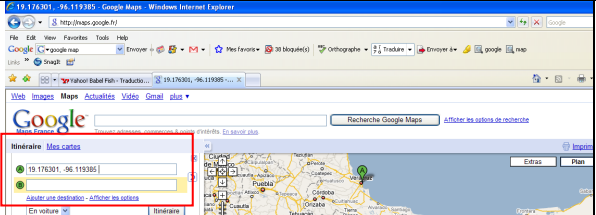
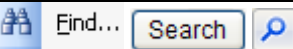
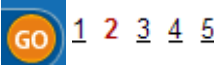
Example of CUI events

AUI event	Example of CUI events		
	Graphical		Vocal
	Graphical UI	Web	
Activate an AIC with input facet	To fill out a form, type or modify text 	To fill out a web form, type or modify text 	vocalInput or record
	to select a task 	to select an option 	

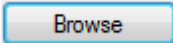
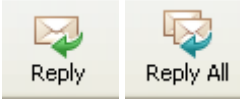
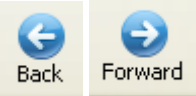
Appendix G

Activate an AIC with output facet	Calendar shows the information of the date (day, month, year)  to advertise about new messages or e-mails  to display content (information or data or images) as a letter 	To display content as a message, a list of contacts, an image	Results 1 - 10 of about 8,090 for usixml. <table><thead><tr><th>Name</th><th>Email</th></tr></thead><tbody><tr><td>mlozano</td><td>mlozano@dsi.uclm.es</td></tr><tr><td>negrissima</td><td>negrissima@hotmail.com</td></tr></tbody></table> 	Name	Email	mlozano	mlozano@dsi.uclm.es	negrissima	negrissima@hotmail.com	vocalPrompt or audio or vocal-Feedback or vocalMenu with vocalMenuItems
Name	Email									
mlozano	mlozano@dsi.uclm.es									
negrissima	negrissima@hotmail.com									
Activate an AIC with input + output facet	The combo box for selecting the percentage for visualizing a document is input when the user select its value but it is output when the user clicks on the magnifier icon and the percentage shows the new selected value 	Text field to edit the location of an itinerary is input when the user type the value but is out put after modifying the value directly on the map		vocalPrompt or (vocalInput and grammar and connect)						

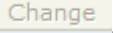
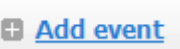
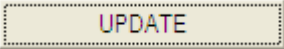
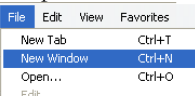
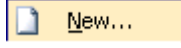
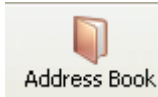
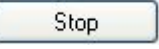
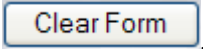
Appendix G

				
Activate an AIC with output + navigation facet	A clock is always present, if previously an alarm was activated, at the time specified a new UI will be displayed 	<pre><output actionType=show actionItem=time name=TimerAlarm> <navigation action- Type=navigate actionItem=alarmAUI></pre>		
Activate an AIC with input + navigation facet	To find or search something 	To go from one AC to other AC (not necessarily the next one)  , to add a comment showing a new AC  , to go or to back from one AC to next or previous one, and show		

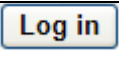
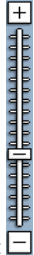
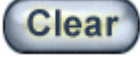
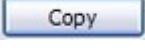
Appendix G

	to browse a directory 	<input actionType=Browse actionItem=directory name=btnBrowse> <navigation action- Type=navigate actionItem=directoryAUI>	 it Previous Next	
	to add a file or photo  , to add a comment showing a new AC 		to search or find something on the Web 	
	to open a file  , to click on a button to type 		to reply an e-mail 	
	to go or to back from one AC to next or previous one, and show it 		to select a view of a web pages 	
	to select a page on an agenda or a sheet in a document  			
Activate	a progress bar		Calendar showing detailed information of the selected	

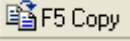
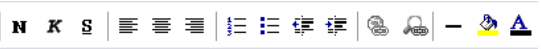
Appendix G

an AIC with output + control facet	 , system shows the page number that is repaginated  ,	 Calendar , a progress bar 	
Activate an AIC with input + control facet	To save changes  , to save modifications  Modify , to update information  , to add information 	To update information  , to add information or files or images 	vocalInput and grammar and submit
	to paste an element  (keyboard shortcut Ctrl+v)	to sort by preferences 	
	to sort by alphabetical order 	to open e-mail  Mail , to open a window 	
	to open a new AC  , to open an address book  Address Book	to play a video file on youtube 	
	to stop an action  , to pause an action  , to cancel an element or action	to cancel an action or element  , to close or exit from a window  , to clear  ,	

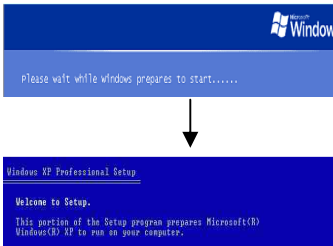
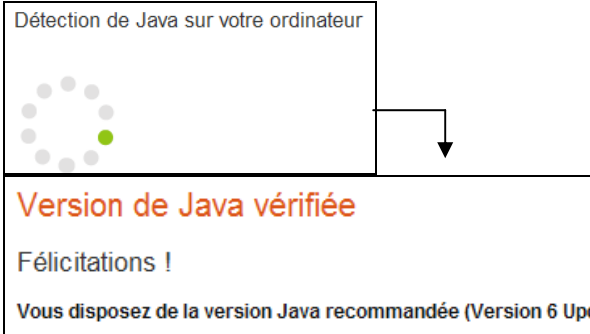
Appendix G

to exit , to exit , click on a push button or icon to close an AC 	to delete something  , to log out , to stop an action 
to move 	click on a push button to register 
to tag an element , to mark 	zoom in or zoom out 
to print 	to print 
to cut an element  Cut (keyboard shortcut Ctrl+x), to clear , to delete 	to get e-mail 
to create a new file 	to copy text, image or file 
to delegate a task 	to refresh an AC 

Appendix G

to save  	to activate or deactivate something  Activate PayPal on this page. Activate Service ☒ 
to activate or deactivate something  	to forward an e-mail 
minimize or maximize or restore a container    , zoom in or zoom out  	to send an e-mail 
to copy an element  (keyboard shortcut Ctrl+c)	to move an element 
to modify text or image  	to edit text 
to view content 	to paste an element  (keyboard shortcut Ctrl+v)

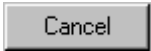
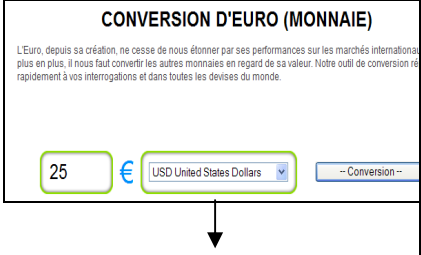
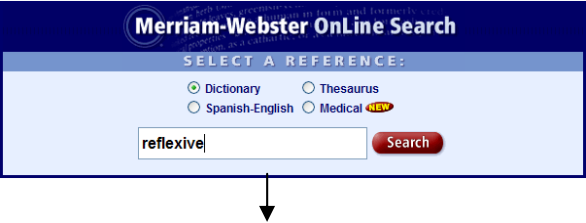
Appendix G

Activate an AIC with output + navigation + control facet	The message preparing an installation is an output that will navigate to the next element when finished with its control 	<pre><output action- Type=Show actionI- tem=message name=outputText> <navigation action- Type=navigate actionI- tem=welcomeSetupAUI> <control action- Type=start actionI- tem=windowsSystem></pre>	When installing plug-ins on a web browser there is an output message indicating that a process is working on your computer after this work there is a navigation sometimes to another page sometimes is just a new element that shows announcing the result of the installation 	

Appendix G

Activate an AIC with input + output + navigation facet	Selecting “search” option, there is a navigation to other AC showing an option Open Explore Search... Look in: Start Menu When was it modified? What size is it? More advanced options	<input action-Type=search actionItem=option name=menuItemSearch> <navigation action-Type=navigate actionItem=searchAUI> <output action-Type=show actionItem=defaultSearchParam eters>	When selecting a different language, there is a navigation to other AC showing its new content Werkzeuge • Liste der diese Seite • Änderungen an verlinkten Seiten • Hochladen • Spezialseiten • Druckversion • Feedback • Partnerseiten • Seite zitieren • Arabisch • Aserbaidschanisch • Chinesisch • Englisch • Französisch • Deutsch • Griechisch • Hebräisch • Hindi • Indonesisch • Italienisch • Japanisch • Koreanisch • Latein • Lateinisch • Niederländisch • Norwegisch • Persisch • Polnisch • Portugiesisch • Rumänisch • Russisch • Serbisch • Spanisch • Swahili • Thai • Türkisch • Ukrainisch • Ungarisch • Vietnamesisch • Walisisch • Weißrussisch • Yoruba Geschichte Erste Wikis entstanden als Wissensverwaltungswerkzeuge von Software-Architekten im Rahmen der <i>Enthusiasten-Theorie</i>. Das erste Wiki, <i>WikiWikiWeb</i> genannt, amerikanischer Softwarestar Ward Cunningham ab 1994 entwickelt und 1995 übers Internet verfügbar gemacht. Das Wiki basierte auf Ideen aus <i>HyperCard</i>-Systemen. 1996 Jahren entwickelt wurde <i>WikiWiki</i>. Der Name <i>WikiWiki</i> kommt, da er bei der Ankunft am Flughafen auf Hawaii die Bezeichnung <i>Wiki Wiki</i> für den dortigen <i>Shuttlebus</i> kennegeheißt. Dabei übernahm er die Verdoppelung, die im Hawaiianisch für eine <i>Stärkung</i> (<i>Leute schnell</i>) steht. Cunningham betrachtet Wiki weiterhin als eine Abkürzung für den eigentlichen Namen <i>WikiWikiWeb</i>. Das Konzept von Wikis ahmt den, was sich <i>Tim Berners-Lee</i>, der Erfinder des <i>World Wide Web</i>, um 1990 ursprünglich unter selbigen vorstellte. Die Informationen sollten am privaten Rechner verfügbar und sofort bearbeitbar sein. In historischer Perspektive beschreibt er das in seinem Buch <i>Viewing The Web</i> (<i>deutsche Lebensbegegnung</i>, <i>Der Web-Support</i>)^[1] Wikis sind typische Beispiele für das so genannte <i>Web 2.0</i>, welches neue Aspekte in <i>Interaktion</i> und <i>Sozialkompetenz</i> einfließt. Konzeption Wesentlich bei der meisten Wiki-Software, darunter das in der <i>Wikipedia</i> selbst eingesetzte <i>MediaWiki</i> und das oft in Unternehmens-Intranets Werkzeuge • Liste der diese Seite • Änderungen an verlinkten Seiten • Hochladen • Spezialseiten • Druckversion • Feedback • Partnerseiten • Seite zitieren • Arabisch • Aserbaidschanisch • Chinesisch • Englisch • Französisch • Deutsch • Griechisch • Hebräisch • Hindi • Indonesisch • Italienisch • Japanisch • Koreanisch • Latein • Lateinisch • Niederländisch • Norwegisch • Persisch • Polnisch • Portugiesisch • Rumänisch • Russisch • Serbisch • Spanisch • Swahili • Thai • Türkisch • Ukrainisch • Ungarisch • Vietnamesisch • Walisisch • Weißrussisch • Yoruba Geschichte Erste Wikis entstanden als Wissensverwaltungswerkzeuge von Software-Architekten im Rahmen der <i>Enthusiasten-Theorie</i>. Das erste Wiki, <i>WikiWikiWeb</i> genannt, amerikanischer Softwarestar Ward Cunningham ab 1994 entwickelt und 1995 übers Internet verfügbar gemacht. Das Wiki basierte auf Ideen aus <i>HyperCard</i>-Systemen. 1996 Jahren entwickelt wurde <i>WikiWiki</i>. Der Name <i>WikiWiki</i> kommt, da er bei der Ankunft am Flughafen auf Hawaii die Bezeichnung <i>Wiki Wiki</i> für den dortigen <i>Shuttlebus</i> kennegeheißt. Dabei übernahm er die Verdoppelung, die im Hawaiianisch für eine <i>Stärkung</i> (<i>Leute schnell</i>) steht. Cunningham betrachtet Wiki weiterhin als eine Abkürzung für den eigentlichen Namen <i>WikiWikiWeb</i>. Das Konzept von Wikis ahmt den, was sich <i>Tim Berners-Lee</i>, der Erfinder des <i>World Wide Web</i>, um 1990 ursprünglich unter selbigen vorstellte. Die Informationen sollten am privaten Rechner verfügbar und sofort bearbeitbar sein. In historischer Perspektive beschreibt er das in seinem Buch <i>Viewing The Web</i> (<i>deutsche Lebensbegegnung</i>, <i>Der Web-Support</i>)^[1] Wikis sind typische Beispiele für das so genannte <i>Web 2.0</i>, welches neue Aspekte in <i>Interaktion</i> und <i>Sozialkompetenz</i> einfließt. Konzeption Wesentlich bei der meisten Wiki-Software, darunter das in der <i>Wikipedia</i> selbst eingesetzte <i>MediaWiki</i> und das oft in Unternehmens-Intranets Werkzeuge • Liste der diese Seite • Änderungen an verlinkten Seiten • Hochladen • Spezialseiten • Druckversion • Feedback • Partnerseiten • Seite zitieren • Arabisch • Aserbaidschanisch • Chinesisch • Englisch • Französisch • Deutsch • Griechisch • Hebräisch • Hindi • Indonesisch • Italienisch • Japanisch • Koreanisch • Latein • Lateinisch • Niederländisch • Norwegisch • Persisch • Polnisch • Portugiesisch • Rumänisch • Russisch • Serbisch • Spanisch • Swahili • Thai • Türkisch • Ukrainisch • Ungarisch • Vietnamesisch • Walisisch • Weißrussisch • Yoruba Geschichte Erste Wikis entstanden als Wissensverwaltungswerkzeuge von Software-Architekten im Rahmen der <i>Enthusiasten-Theorie</i>. Das erste Wiki, <i>WikiWikiWeb</i> genannt, amerikanischer Softwarestar Ward Cunningham ab 1994 entwickelt und 1995 übers Internet verfügbar gemacht. Das Wiki basierte auf Ideen aus <i>HyperCard</i>-Systemen. 1996 Jahren entwickelt wurde <i>WikiWiki</i>. Der Name <i>WikiWiki</i> kommt, da er bei der Ankunft am Flughafen auf Hawaii die Bezeichnung <i>Wiki Wiki</i> für den dortigen <i>Shuttlebus</i> kennegeheißt. Dabei übernahm er die Verdoppelung, die im Hawaiianisch für eine <i>Stärkung</i> (<i>Leute schnell</i>) steht. Cunningham betrachtet Wiki weiterhin als eine Abkürzung für den eigentlichen Namen <i>WikiWikiWeb</i>. Das Konzept von Wikis ahmt den, was sich <i>Tim Berners-Lee</i>, der Erfinder des <i>World Wide Web</i>, um 1990 ursprünglich unter selbigen vorstellte. Die Informationen sollten am privaten Rechner verfügbar und sofort bearbeitbar sein. In historischer Perspektive beschreibt er das in seinem Buch <i>Viewing The Web</i> (<i>deutsche Lebensbegegnung</i>, <i>Der Web-Support</i>)^[1] Wikis sind typische Beispiele für das so genannte <i>Web 2.0</i>, welches neue Aspekte in <i>Interaktion</i> und <i>Sozialkompetenz</i> einfließt. Konzeption Wesentlich bei der meisten Wiki-Software, darunter das in der <i>Wikipedia</i> selbst eingesetzte <i>MediaWiki</i> und das oft in Unternehmens-Intranets
Activate an AIC with input + output + control facet	Introducing a value to get a new value after an operation 100 Euro (EUROPE) Dollar des États-Unis (ÉTATS-UNIS) Go 141.44		Introducing data to obtain new data after a conversion En español Hay docenas de pocos motores de la búsqueda hacia fuera allí a los cuales usted puede ser que considere someter, pero no son generalmente digno de su tiempo. Rechercher ce texte sur le Web Traduire à nouveau (Inscrivez jusqu'à 150 mots) There are dozens of lesser search engines out there that you might consider submitting to, but usually they aren't worth your time. Anglais vers espagnol Traduire

Appendix G

Activate an AIC with input + navigation + control facet	to cancel an element or action and return/go to other AC  to click on a push button or icon to close an AC and return/go to other AC 	to cancel an action or element and return/go to other AC  to close or exit from a AC and return/go to other one  <pre><input action-Type=close actionI-tem=window name=btnclose> <navigation ac- tion-Type=navigate actionI-tem=previousAU I> <control action-Type=finish actionI-tem=application ></pre>	
Activate an AIC with input + output + navigation + control	Introducing a value to get a new value after an operation and present the result in a new AC 	<pre><input action-Type=insert actionI-tem=amount name=val1> <control action-Type=calculate actionI-tem=convertCurr ency> <navigation ac-</pre> Introducing data to obtain new data after a search and present the result in a new AC 	

Appendix G

	Universal Currency Converter® Results Using live mid-market rates. More currencies Live rates at 2008.09.15 13:09:18 UTC 25.00 EUR = 35.3302 USD Euro United States Dollars 1 EUR = 1.41321 USD 1 USD = 0.707810 EUR Trade Currencies. Click here!!	tion- Type=navigate actionI- tem=resultAUI> <output action- Type=show actionI- tem=valResult>	Merriam-Webster's Online Dictionary reflexive 4 entries found. ¹reflexive (adjective) ²reflexive (noun) reflexive pronoun self-reflexive Main Entry: reflexive 🔊 Pronunciation: rɪf-ˈlɛks-ɪv Function: adjective Etymology: Medieval Latin <i>reflexivus</i>, from Latin <i>reflexus</i> Date: 1640 1 a : directed or turned back on itself, also : overtly and usually ironically <i>reflecting</i> conventions of genre or form <a reflexive novel> b : marked by or capable of reflection : REFLECTIVE 2 : of, relating to, characterized by, or being a relation that exists between an entity and itself <the relation 'is equal to' is reflexive but the relation 'is the father of' is not> 3 : of, relating to, or constituting an action (as in 'he prejudiced himself') directed back on the agent or the grammatical subject 4 : characterized by habitual and unthinking behavior	

Table G-1. Some mappings between concrete and abstract events.