

HOVEDOPPGAVE

Kandidatenes navn: Jon Ole Nødtvedt og Man Hoang Nguyen
Fag: Datateknikk
Oppgavens tittel (norsk):
Oppgavens tittel (engelsk): Mobility and context-awareness in workflow systems
Oppgavens tekst:

The main objective of this project is to develop a workflow prototype that incorporates context information caused by mobility. This information can be used to control transitions between activities or provide useful functionality to the user. The workflow client should be context-aware.

Oppgaven gitt: 19. januar 2004
Besvarelsen leveres innen: 14. juni 2004
Besvarelsen levert: 14. juni 2004
Utført ved: Institutt for Datateknikk og Informasjonsvitenskap
Veileder: Carl-Fredrik Sørensen

Trondheim, 14. juni 2004

Faglærer
Alf Inge Wang

Abstract

This project aims to describe how workflow systems can integrate and make use of context information from context rich environments, to enhance the execution of workflow processes. Context information can for example be used to control transitions between activities, activity enactment and process/activity coordination. A dynamic contextual environment also requires that a workflow system is capable of responding to contextual events. A set of requirements for a context-aware workflow system, based on existing workflow standards, theory behind context-aware computing and activity theory, will be presented and elaborated. Prototypes, which illustrate how these requirements can be implemented in a standard based workflow system, are also provided. Based on the solutions presented in the prototypes, a new interface for a workflow enactment service is presented. This new interface serves as the link between the contextual environment and the workflow system. We also present a solution for handling context related exception states. The definition of basic terms in workflow systems are expanded to better support context-aware behaviour. Ideas and solutions for more complex requirements not met in our prototypes are also discussed, such as situated activity coordination.

Preface

This report is the result of a master project in workflow and context-aware systems. The authors behind this report are Jon Ole Nødtvedt and Man Hoang Nguyen from the Institute for Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU) in Trondheim. The master project ran from January to June 2004.

We would like to thank our supervisor, PhD Fellow Carl-Fredrik Sørensen of the Software Engineering group at IDI-NTNU, for valuable insights and comments during our thesis work. We would also like to thank Torkel Fyrvik at MARINTEK, Asgeir Drøivoldsmo at the Institute for Energy Technology in Halden, and Knut-Olav Fjell at Statoil for providing scenarios and ideas to this project. In addition, we would like to thank Gerd Kortuem at the Computing Department of the Lancaster University for making their research results available to us.

Trondheim, June 14, 2004

Jon Ole Nødtvedt

Man Hoang Nguyen

Contents

1. Introduction	1
1.1 Background	1
1.2 Motivation	1
1.3 Problem description.....	2
1.4 Research method	3
1.5 Structure of the report	5
Part I: State-of-the-art	7
2. Workflow	8
2.1 Basic definitions	9
2.2 Workflow Reference Model	11
2.3 Types of workflow.....	16
2.3.1 Summary	17
2.4 Workflow process modelling.....	17
2.4.1 Petri-net	18
2.4.2 State Chart Diagram.....	22
2.4.3 Activity diagram	24
2.4.4 Summary	24
2.5 Exception handling	25
2.5.1 Summary	28
3. Context-awareness	29
3.1 Introduction to context-awareness	29
3.2 Context sensing.....	30
3.3 Model context information in computer systems	31
3.3.1 Context classifications	31
3.3.2 Entity-relationship model	31
3.3.3 Object oriented model	32
3.3.4 World model	33
3.3.5 ContextMap model	34
3.3.6 Event representation	35
3.3.7 Summary	36
3.4 Sentient objects.....	36
3.4.1 Summary	37
4. Situated actions and planning	39
4.1 Summary	41
5. Mobile workgroups with computer based support of their work	42
5.1 HandyMan	42
5.2 Summary	44
Part II: Our contribution.....	45
6. Application scenarios	46
6.1 Scenario 1: Maintenance performing on an oil platform.....	46
6.2 Scenario 2: Intelligent chemicals containers	52
6.3 Discussion	54
6.3.1 Context-awareness.....	54
6.3.2 Situated planning.....	55

6.3.3	Coordination of activities	58
6.3.4	Summary	58
7.	Requirements for workflow systems integrating context information	59
7.1	Basic workflow system requirements	59
7.2	Context information representation and retrieval.....	60
7.3	Context-aware functionality in a workflow system	62
7.3.1	Workflow enactment service context-awareness	62
7.3.2	Workflow client based context awareness.....	64
7.4	Summary of functional requirements.....	66
7.5	Mobility requirements	67
7.6	Non-functional requirements and design considerations.....	67
8.	Prototypes.....	68
8.1	Design overview	68
8.2	Context information used in workflow transitions	69
8.2.1	Prototype packages.....	70
8.2.2	Workflow enactment service.....	71
8.2.3	Inter-process communication	72
8.3	Workflow actions based on context changes	75
8.4	Context exception states handling in workflow systems.....	77
8.4.1	Handling context related exceptions.....	78
8.5	Process path revalidation	81
8.6	Invariant scenario	83
8.7	Client based context-awareness	84
8.7.1	Inference engine.....	84
8.7.2	Workflow client with local process enactment and rule based building of the situated process	85
8.7.3	Workflow client with activity contextual post conditions.....	87
	Part III: Discussion and conclusion	89
9.	Discussion	90
9.1	Discussion of prototypes	90
9.1.1	Context information used in the processing of workflow transitions ..	90
9.1.2	Workflow actions based on context changes	90
9.1.3	Handling context related exceptions.....	91
9.1.4	Process path selection and invariants	92
9.1.5	Client based context-awareness	92
9.2	Evaluation of research method.....	94
10.	Future work.....	95
11.	Conclusion.....	97
	Glossary	101
	References.....	105
	Appendix A: Vertical Prototyping.....	113
	Appendix B: UML State Chart Diagram	115
	Appendix C: UML Activity diagram.....	116
	Appendix D: Test report	117
	Appendix E: Context information used in workflow transitions process definition...	122
	Appendix F: Class diagrams for all packages in the initial prototype	125
	Appendix G: Workflow actions based on context changes process definition	130

Appendix H: Revised process for exception condition	133
Appendix I: XML schema for exception handling rules	136
Appendix J: XML exception handling document for context state exception condition	139
Appendix K: Process path revalidation process specification	140
Appendix L: Process path revalidation exception handler rules.....	144
Appendix M: Process description for invariant scenario	146
Appendix N: Exception handler rules for invariant scenario.....	151
Appendix O: Workflow client with local process enactment and rule based building of the situated process – process description	153
Appendix P: Workflow client with local process enactment and rule based building of the situated process – client knowledge base.....	159
Appendix Q: Workflow client with activity contextual post conditions process description.....	162

List of figures

Figure 1: Report overview	6
Figure 2: An example of an order processing workflow	8
Figure 3: Basic terms and their relationship	9
Figure 4: Workflow Reference Model – components and interfaces.....	12
Figure 5: Workflow process definition meta-data model	14
Figure 6: Workflow primitives	19
Figure 7: The structure of a task	20
Figure 8: The structure of a procedure.....	21
Figure 9: An UML State Chart Diagram for an order processing workflow	23
Figure 10: An UML activity diagram for an order processing workflow	24
Figure 11: Overall FAR architecture.....	27
Figure 12: Entity relationships between key entities	32
Figure 13: Classification of context for an entity	32
Figure 14: Class hierarchy for the context information model	33
Figure 15: An example of a ContextMap.....	35
Figure 16: Simple sentient object model	37
Figure 17: Screenshot from HandyMan	43
Figure 18: Maintenance related business processes	47
Figure 19: “Verify condition” workflow	48
Figure 20: “Plan maintenance job” workflow	49
Figure 21: “Accomplish and Report” workflow	50
Figure 22: “Prioritise and Co-ordinate” workflow	51
Figure 23: Left: physical view of an intelligent chemicals container. Right: Intelligent Container Test Bed	52
Figure 24: Example arrangement illustrating different hazards: (a) no hazard, (b) critical mass exceeded, (c) reactive chemicals in proximity, and (d) container stored in a disapproved area too long. The exclamation mark indicates which containers are involved in a hazardous condition.	53
Figure 25: “Accomplish and Report” workflow, revised figure.....	57
Figure 26: Use case diagram for a basic workflow system	60
Figure 27: Use case diagram for context framework and context-aware application	62
Figure 28: Use case diagram for workflow enactment service context-awareness ..	64
Figure 29: Use case diagram for workflow client based context-awareness	65
Figure 30: Workflow process illustration	70
Figure 31: Component view of the complete prototype.....	70
Figure 32: Class diagram of the most important classes in the workflow enactment service	72
Figure 33: Workflow client registration	73
Figure 34: Workflow client activity processing	74
Figure 35: Sequence diagram for context source polling from a workflow enactment service	75
Figure 36: Workflow processes illustration for context source subscription	76
Figure 37: Context source subscription with process initialisation	77
Figure 38: Exception scenario.....	78
Figure 39: Exception handling revised process illustration	79
Figure 40: Class diagram for the exception handler package.....	80
Figure 41: Sequence diagram for the exception handlings scenario	81
Figure 42: Process path revalidation process illustration.....	82
Figure 43: Invariant scenario process illustration.....	83
Figure 44: Processes for client based context-awareness prototype.....	86
Figure 45: Class diagram for the updated workflow client	86

Figure 46: Sequence diagram for activity post condition	88
Figure 47: WfMC workflow reference model with new interface.	97
Figure 48: Revised basic terms and relationships	98

List of tables

Table 1: An overview of prototyping techniques	114
Table 2: Basic state chart diagram symbols and notations.....	115
Table 3: Basic activity diagram symbols and notations.....	116
Table 4: Software test types.....	117
Table 5: System test for the prototype in Chapter 8.2.....	118
Table 6: System test for the prototype in Chapter 8.3.....	118
Table 7: System test for the prototype in Chapter 8.4.....	119
Table 8: System test for the prototype in Chapter 8.5.....	119
Table 9: System test for the prototype in Chapter 8.6.....	120
Table 10: System test for the prototype in Chapter 8.7.2.....	120
Table 11: System test for the prototype in Chapter 8.7.3.....	121

1. Introduction

This chapter will give the reader an introduction to the project report and present the project background, motivation, problem description and research method for our work.

1.1 Background

The project description for this thesis was given by the MOWAHS¹ project [81]. The MOWAHS project is a basic research project carried out jointly by the Software Engineering group and the database technology group at IDI. The project is supported by the Norwegian Research Council. MOWAHS consists of two parts. One part of the project looks at how mobile process support can be achieved for heterogeneous devices. The other part looks at how support for cooperative workspaces/transactions can be achieved. These topics are interesting because of the growth of usage of mobile devices with Internet connection. Current tools for mobility support are immature, and herein lay the research challenge. The MOWAHS project has three research goals:

- Helping to understand the work processes in virtual organisations and how they can be improved.
- Provide a flexible work environment where work processes can be executed and shared together with artefacts belonging to those work processes.
- Distributing the results.

In our thesis work, the focus will be on work processes in the context of workflow systems. Further, we want to look at how context information can be integrated into workflow systems to enhance the execution of workflow processes. The context information can among other things be used to control transitions between activities, activity enactment and provide useful functionality to the user. The focus of our work is related to the second research goal for the MOWAHS project. We try to use the dynamic information from a context rich environment, to provide more dynamic workflow processing and thereby provide a more flexible work environment for the users.

1.2 Motivation

The motivations for a context-aware workflow system are two-fold. The first motivation comes from the requirements for support of mobility in computer systems. As mobile devices have become available, companies have started seeing the value in using computerised tools in mobile business processes. The reason for this is that such

¹ Mobile Work Across Heterogeneous Systems

tools may increase efficiency and profitability by providing needed information and services in the field, which would otherwise not be available. When computerised tools support mobility, they should also be able to dynamically handle changing contextual surroundings. Acquisition and interpretation of context information from the surrounding environment may be necessary for the correct execution of the computer system and/or to provide valuable context specific information and services to the overall system and the user.

The second motivation comes from the value workflow systems provide in industry today. Workflow systems provide a way to plan and execute business critical processes as efficiently as possible. The adoption of mobile computerised equipment in companies also provides the possibility for extending workflow systems with support for mobile work processes and activities. Mobile work processes had previously no workflow support. As stated earlier, context-awareness becomes an important issue when using computerised tools in a mobile setting. This is also the case, when we are dealing with computerised workflow support of processes and activities performed in the field. The support of such processes and activities requires a dynamic behaviour not previously found in workflow systems. As will be explained later in this report, workflow systems are generally very statically defined, with support for only minimal amounts of dynamic behaviour. It is therefore important to study how workflow systems can integrate and use context information in their support of process and activity enactment.

1.3 Problem description

The initial task description for this thesis was:

“The main objective of this project is to develop a workflow prototype that incorporates context information caused by mobility. This information can be used to control transitions between activities or provide useful functionality to the user. The workflow client should be context-aware”.

As the task description states, our focus will be on the development of workflow prototypes, which will be context-aware. Please note that issues regarding mobility specifically will not be the focus for this report. Instead, mobility will be considered as a source for changes in context information. Some issues regarding mobility will nevertheless be discussed briefly.

The report will provide a pre-study of workflow technology and context-aware computing. Further, based on the pre-study and presented scenarios, requirements for context-aware workflow systems will be presented. Suggested solutions to these requirements will be presented in several prototypes. The key to our contribution lies within the functionality that is implemented in the prototypes and how they relate to the existing workflow standards.

The problem description clearly relates our work to existing research performed in the field of mobile, context-aware computer systems. Since workflow systems are a

concrete example of groupware² systems, which satisfy the need for managing task interdependencies, and we are focusing on the design of such systems, the relation to existing work in the groupware field is also clear. Our work does not contribute to the understanding of cooperative work. Instead we base our understanding of cooperative work on existing work and our own assumptions, thus this report is not directly related to the research field of CSCW³.

1.4 Research method

Research in software engineering can rely on a wide range of methods. Most of these research methods have the ultimate goal of answering empirical questions through controlled experiments, but different questions call for different research methods, because the nature of a research questions often constraint the methods that can be used to answer them. According to [64], there are four general categories of research approaches, each of which can be stated as follows:

- **Scientific method:** Scientists develop a theory to explain a phenomenon; they propose a hypothesis and then test alternative variations of the hypothesis. As they do so, they collect data to verify or refute the claims of the hypothesis.
- **Engineering method:** Engineers develop and test a solution to a hypothesis. Based upon the results of the test, they improve the solution until it requires no further improvement.
- **Empirical method:** A statistical method is proposed as a means to validate a given hypothesis. Unlike the scientific method, there may not be a formal model or theory describing the hypothesis. Data is collected to verify the hypothesis.
- **Analytical method:** A formal theory is developed, and results derived from that theory can be compared with empirical observations.

The main objective for our thesis is to develop prototypes that illustrate the concepts behind context-aware workflow systems. These concepts refer to how context information arising from for example mobility can be integrated into workflow systems and be used to improve the execution of workflow processes. To achieve this objective, our research work has been conducted following a combination of the scientific and engineering method. It is important to mention that the prototypes we have proposed in this report are not meant to be final products which do not require any further improvement, as stated in the engineering method.

The task description forms the basis for our work. However, to elaborate further on this description, we have defined a set of research questions to be answered in our work:

- How to integrate context information that is usable for process reasoning and enactment in workflow systems?

² Groupware definition: “Computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment” [19].

³ CSCW definition: “CSCW should be conceived as an endeavour to understand the nature and requirement of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements” [49].

- How to specify rules and components necessary for a workflow system to be reactive to context changes before and during workflow process and activity execution?
- How can a workflow system handle contextual exception states?
- How can context information be used to enable more dynamic process execution with regards to selection of process paths?
- How can situated actions and situated process reasoning be supported in a workflow system?
- How is situated process and activity enactment accounted for at a later time?
- How to select and use relevant context sources related to current activity or process plan?
- How can a workflow system function as a coordinator for cooperative situated activities?
- How to create intelligible and accountable context-aware workflow process systems? By intelligible we mean that a workflow system should be able to show users what it knows about context information and what it is doing about it. By accountable we mean that the system should enforce user accountability, when it tries to mediate user actions that impact others.

Based on existing standards and research in the fields of context-aware systems and workflow systems, we will try to find the interfaces and components necessary to allow for context-aware workflow systems. Our solutions will be illustrated through vertical prototypes⁴.

In addition, our research method includes the following activities:

- Literature survey: This survey was conducted to identify related literature to workflow technology, support for context aware systems, situated-actions and situated planning. Using the Internet and the libraries located at Gløshaugen, Trondheim, we were able to collect and structure relevant information such as articles, books and notes, which are within our subject. For the Internet we used research sites such as Association for Computing Machinery (ACM) [69], Elsevier Computer Science [73], Cite Seer [71] and Science Direct [84] for seeking after relevant documents. The literatures study has given us useful knowledge and information, which helped us to understand the theoretical part of this thesis.
- Review of different sources: This review aimed at identifying the requirements of workflow system, design and architecture considerations, and understanding existing standards.
- Collection of data: We have been in touch with relevant companies where our purpose was to collect information about workflow processes that are being used in different industries. The meetings were conducted at the Statoil offices at Rotvoll Forskningspark, at the offices of the Institute for Energy Technology (IFE) in Halden and at NTNU where we met a representant from MARINTEK. At Statoil we found that they were focused on mobile technology that could be used in oil platforms. For us, it would be a great

⁴ Vertical Prototyping: A technique for doing prototyping. See Appendix A for a description of this term.

opportunity to get useful input from Statoil, but they did unfortunately not have time and resources to cooperate with us. The meeting with IFE gave us the opportunity to meet scientists working in the field of augmented reality⁵. They found the ideas behind our research to be very interesting, but because of the current state of their own projects, they did not have time to cooperate with us. From Marintek we were given a case study, which contains workflow process information of how to perform maintenance on an oil platform. This case study has been used as a basis in the development of one scenario⁶. In addition, we created another scenario based on the research results we got from the people at Lancaster University. We also participated in a seminar⁷, which took place at Ingeniørenes Hus in Oslo. At this seminar, we got the chance to get updated on the current usage of mobile technology in the Norwegian industry. Several companies and technology providers presented respectively their needs and products within mobile computing at this seminar. This has helped us to understand the current state of mobile computing in the industry and which technology that is available in the market. Some of what we learnt at this seminar has been used to specify how mobile workgroups use mobile computing equipment.

The activities have been conducted in parallel to optimise resource usage and to reduce the risks connected to the research nature of this project. In addition, performing activities in parallel can improve cross-fertilisation between concurrently running parts of the project. This approach has proven to be very useful for us.

1.5 Structure of the report

The structure of this report reflects the order in which these issues have been dealt with throughout the research process. This report is organised as follows:

- Chapter 1: Gives the reader an introduction to the project report.
- Chapter 2: Presents the technology and ideas behind workflow systems.
- Chapter 3: Gives the reader an introduction to context-awareness in computer systems.
- Chapter 4: Provides the theory behind situated actions and planning.
- Chapter 5: Studies mobile workgroups with computerised work support systems.
- Chapter 6: Presents application scenarios for context-aware workflow systems.
- Chapter 7: Presents requirements for a context-aware workflow system
- Chapter 8: Presents the prototypes developed in response to the requirements.
- Chapter 9: Discusses our findings from Chapter 7 and 8.
- Chapter 10: Presents what further work should be done, based on our contribution and our discussion.
- Chapter 11: Draws conclusions based on all parts of the report.

⁵ “An augmented reality system generates a composite view for the user. It is a combination of the real scene viewed by the user and a virtual reality (VR) scene generated by the computer that augments the scene with additional information”. [65]

⁶ “Scenarios are short stories, descriptions about use of technology contextualised in a meaningful setting. ... [Scenarios] can ... be used in describing completely novel practices made possible by new technology. [29]

⁷ The subject of the seminar was “Mobile IKT-løsninger i industrien – status og fremtidsmuligheter”.

Figure 1 shows a graphical view of the chapters and how they are related in this report. Chapter 1 contains our introduction, while Part I consists of our presentation of workflow in Chapter 2, context-awareness in Chapter 3, situated actions and planning in Chapter 4 and finally mobile workgroups in Chapter 5. Part II is made up of the application scenarios and our contribution. While the final part, discusses and draws conclusions based on all our previous parts.

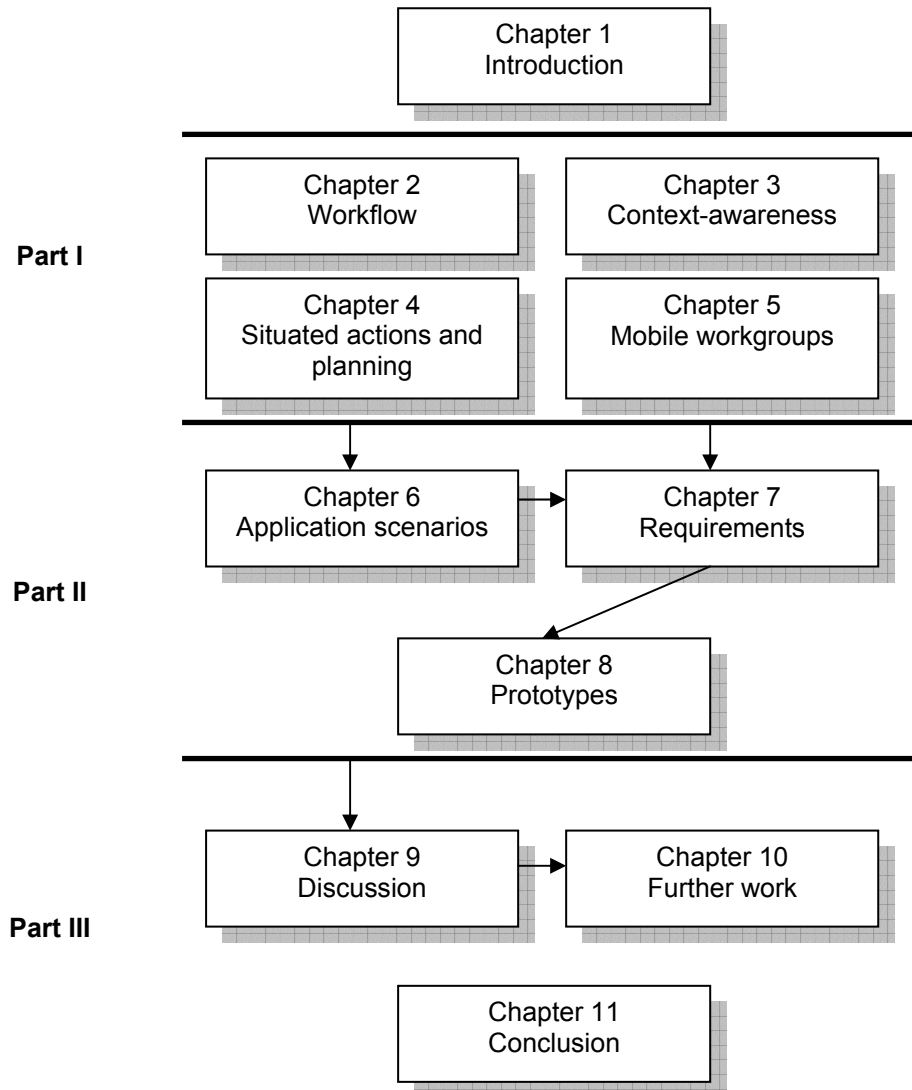


Figure 1: Report overview

Part I: State-of-the-art

The objective of this part is to provide a state-of-the-art study of workflow technologies, context-aware systems and other related subjects. Part I begins with an introduction to workflow. This includes workflow terminology, the workflow reference model, types of workflow, exception handling and workflow modelling. Further, an introduction to context-awareness, context sensing, context modelling and sentient objects will be described in the chapter for context-awareness. The chapter for situated actions and planning provides an introduction to these topics, which will be used as a basis for further study into how situated actions and planning influences workflow systems. The last chapter in this part describes some mobile workgroups with computer based support of their work.

2. Workflow

A workflow can be described as a collection of tasks that are completed by multiple resources. A workflow engine controls the execution of a defined business process. The sequence of tasks constituting a workflow could execute over a period from a few milliseconds to run for months. Typically, business processes last from a few minutes to several days. Workflow can thus be regarded as long-lived transactions.

The concept of workflow has been around for some time. Technologies like Java [89], XML [96], and the Web have made it easier to define and exchange information across applications. The growth of the Web has also had a major impact on workflow, establishing a ubiquitous⁸ platform to interact and participate in workflows. This empowers businesses using workflow and makes workflow-based tools especially useful. As businesses utilise the Web more dynamically for e-commerce [66] and for interfacing with customers, partners, suppliers and employees, the use of workflow technology becomes imperative. Using workflow encourages a business to capture and define the processes it uses. As an example, Figure 2 show how a part of an order processing workflow [72] can be automated.

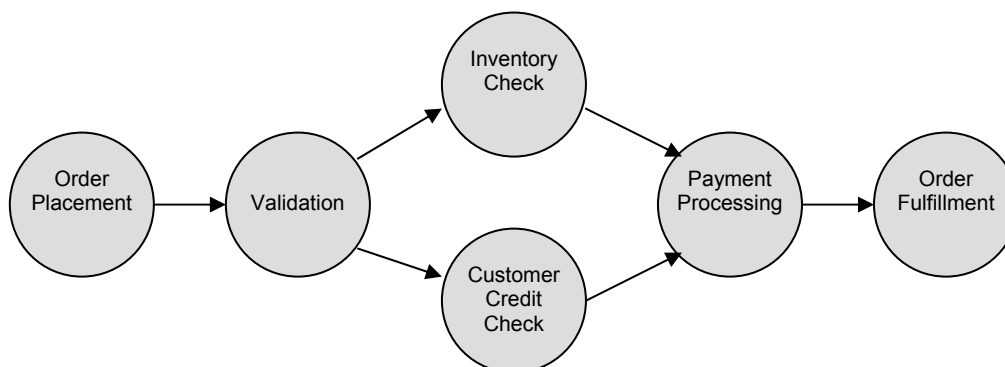


Figure 2: An example of an order processing workflow

Each node in this workflow represents an activity. Activities can depend on other activities, like **Validation** requires the completion of **Order Placement**, before it can start. Some activities can proceed in parallel, like **Inventory Check** and **Customer Credit Check**. Activities may be automated, or they may require manual processing. In general, a workflow could have a combination of automated and non-automated activities. Later in the report, we will present a more generic workflow process, which is implemented in our prototype. This prototype uses the Workflow Management Coalition's Interface 1 specification, which we will describe greater detail in this section.

⁸ "Ubiquitous" is a term meaning the seamless integration of computing into the fabric of everyday life.

2.1 Basic definitions

In this section, the basic terms of workflow terminology will be defined. The relationships among them are illustrated in Figure 3 [61] according to the Workflow Management Coalition (WfMC) [101]. WfMC is a non-profit international standardization organization founded in August 1993. WfMC Members include workflow vendors, users, consultants and people from the academic community. WfMC was founded to encourage the use of workflows. Main goals include defining workflow terminology and standardising specification used for interconnecting workflow products.

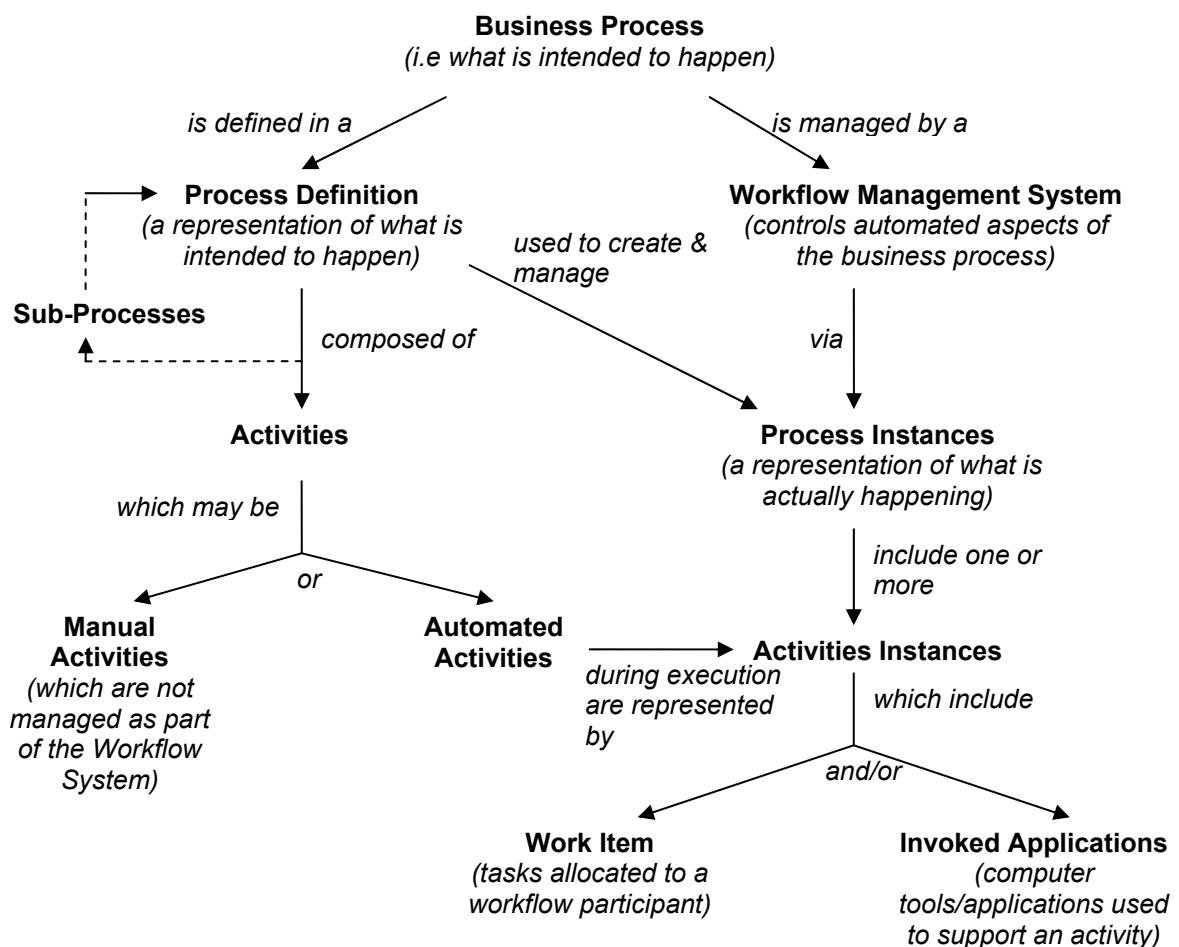


Figure 3: Basic terms and their relationship⁹

Workflow

WfMC has defined *workflow* in [61] as:

The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.

⁹ Copyright @ 2002 The Workflow Management Coalition

A workflow consists of a process that is automated. A work item or data set is created, processed and changed in stages at a number of processing points to meet business goals. Most workflow engines can handle complex series of processes. Conditions, that can be expressed mathematically or logically, can be managed by a workflow system.

A workflow process is normally based on several logical steps, each of which is known as an activity. An activity can involve manual interaction with a user or workflow participant, or the activity might be executed using machine resources. Automating the actual work may increase efficiency, and provide managers with the facilities to create the virtual organization, and to participate effectively in e-commerce.

The concept of a virtual organization can be viewed as a set of participants with various relationships that wish to share resources to perform some task. For example, if several parties co-operate together towards a particular project, in for example a space shuttle development project, a virtual organization is formed and each party will be considered as the member of the virtual organization.

Workflow Management System

A workflow is created and managed in a workflow management system. WfMC has defined *workflow management system* in [61] as:

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.

When making a workflow specification, a workflow planner would look at a process to be defined and divide it into individual sub-processes. The sub-processes are then divided into activities. This workflow specification forms the basis for the execution of the workflow process and once made, several instances of the workflow process can be executed simultaneously.

The main goal of a workflow management system is to manage the flow of activities through the workflow system. Users and their roles are managed through the workflow management system. The participation of users are managed by setting deadlines, activity synchronisation and by passing activity data from one participant to another and ensuring that they fulfil their contribution as expected.

Business Process

A *business process* is a set of one or more linked procedures or activities, which collectively realise a business objective or policy goal. Normally it is made up within the context of an organization structure defining functional roles and relationships [61].

Process Definition

A *process definition* is a representation of a business process in a computerised form. The representation supports automated manipulation, such as modelling, or enactment

by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc. [61].

Activity

An *activity* is a description of a piece of work that forms one logical step within a process. An activity may be manual, which is not supported by computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resource(s) to support process execution: where a human resource is required, an activity is allocated to a workflow participant [61].

Automated activity

An *automated activity* is an activity which is capable of computer automation using a workflow management system to manage the activity during execution of the business process, which it forms a part of [61].

Manual activity

A *manual activity* is an activity within a business process, which is not capable of automation and hence lies outside the scope of a workflow management system. Such activities may be included within a process definition, for example to support the modelling of the process, but do not form part of a resulting workflow [61].

Process Instance

A *process instance* is the representation of a single enactment of a process including its associated data. It therefore represents an instance of a process definition that includes manual and automated aspects [61].

Activity Instance

An *activity instance* is the representation of an activity within a (single) enactment of a process, for instance within a process instance [61].

Work Item

A *work item* is a representation of the work to be processed (by a workflow participant) in the context of an activity within a process instance [61].

Invoked Application

An *invoked application* is a workflow application that is invoked by the workflow management system to automate an activity, fully or in part, or to support a workflow participant in processing of a work item [61].

2.2 Workflow Reference Model

The workflow reference model [62] has been developed by the WfMC to identify the interfaces within the generic workflow product structure. Different products in the market will have different levels of conformance to these models due to different positioning in the market, especially concerning interoperability. Figure 4 [62]

illustrates the workflow reference model that defines a reference architecture consisting of components and related interfaces among them.

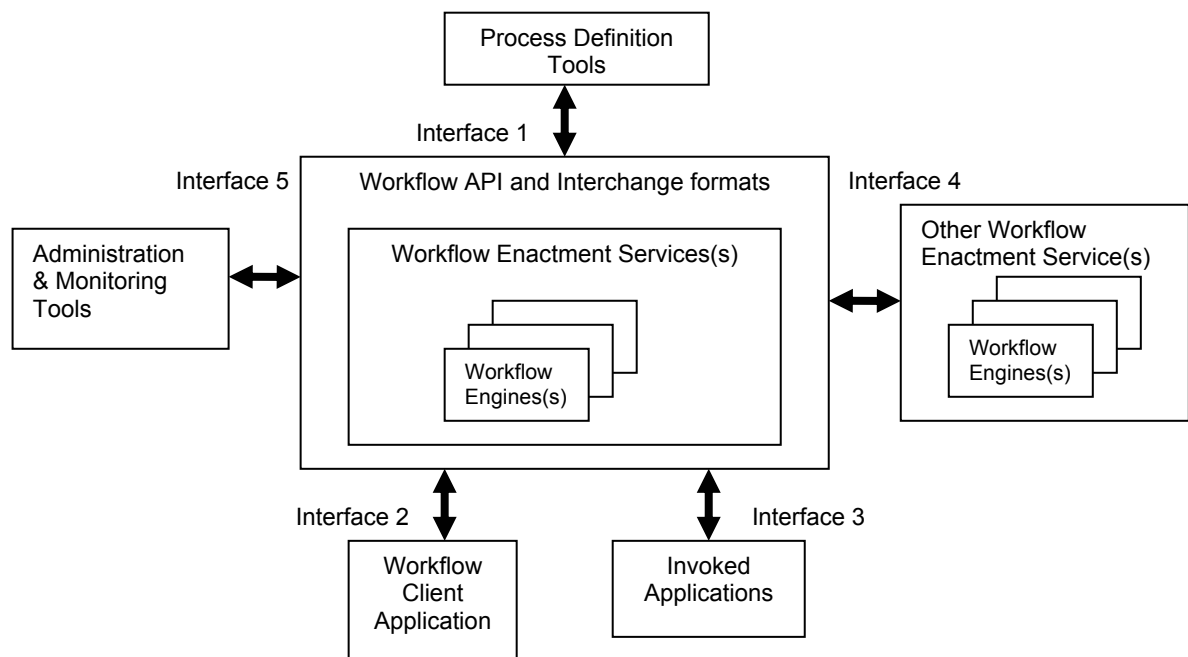


Figure 4: Workflow Reference Model – components and interfaces¹⁰

Below is a description of these components and interfaces in Figure 4:

Workflow Enactment Service

A workflow enactment service is a software service that consists of one or more workflow engines to manage and execute particular workflow instances [62]. As we see in the reference model (Figure 4), the workflow enactment service is separated from the other functions, which must be performed by a workflow management system through the use of interfaces. Applications may interface to this service via the workflow application programming interface (WAPI) [60].

The workflow management service may be centralised or functionally distributed. In a distributed workflow enactment service, several workflow engines are involved in the enactment of one process. Each workflow engine interacts only with users and application tools managed by it. A homogenous workflow enactment service comprises one or more compatible workflow engines, which provide the run-time execution environment for workflow processes with a defined set of process definition attributes. A heterogeneous workflow enactment service comprises two or more homogeneous services, which follow common standards for interoperability at a defined conformance level.

¹⁰ Copyright @ 2002 The Workflow Management Coalition

Workflow Engine

A workflow engine is a software service that provides the run-time execution environment for a process instance [62]. A workflow engine may be responsible for the whole run-time process execution, but also for only a part of it.

Workflow Application Programming Interface (WAPI) & Interchange Formats

WAPI [63] is an abbreviation for Workflow API's and Interchange Formats, published by the WfMC, and incorporating specifications to enable interoperability between different components of workflow management systems and applications. WAPI consists of a set of API calls and interchange functions supported by a workflow enactment service at its boundary for interaction with other resources and applications. Most of the WAPI are APIs with defined parameter and result sets.

We will now describe the five interfaces that are presented in the reference model (Figure 4):

Interface 1: Process Definition Tools

This interface provides software tools that are used by process designers to create a representation of a business process, including all process related data, which can be interpreted by a workflow enactment service later.

The specification for this interface [63] defines a common interchange format which allows different tools to share process definitions and exchange information. The interface covers standard definitions and the interchange of such information as:

- Process start and termination conditions.
- Identification of activities within a process.
- Identification of data types and access paths.
- Definition of transition conditions and flow rules.
- Information for resource allocation decisions.

To illustrate the sharing of process definitions, a workflow process meta-data model has been created [63] in Figure 5:

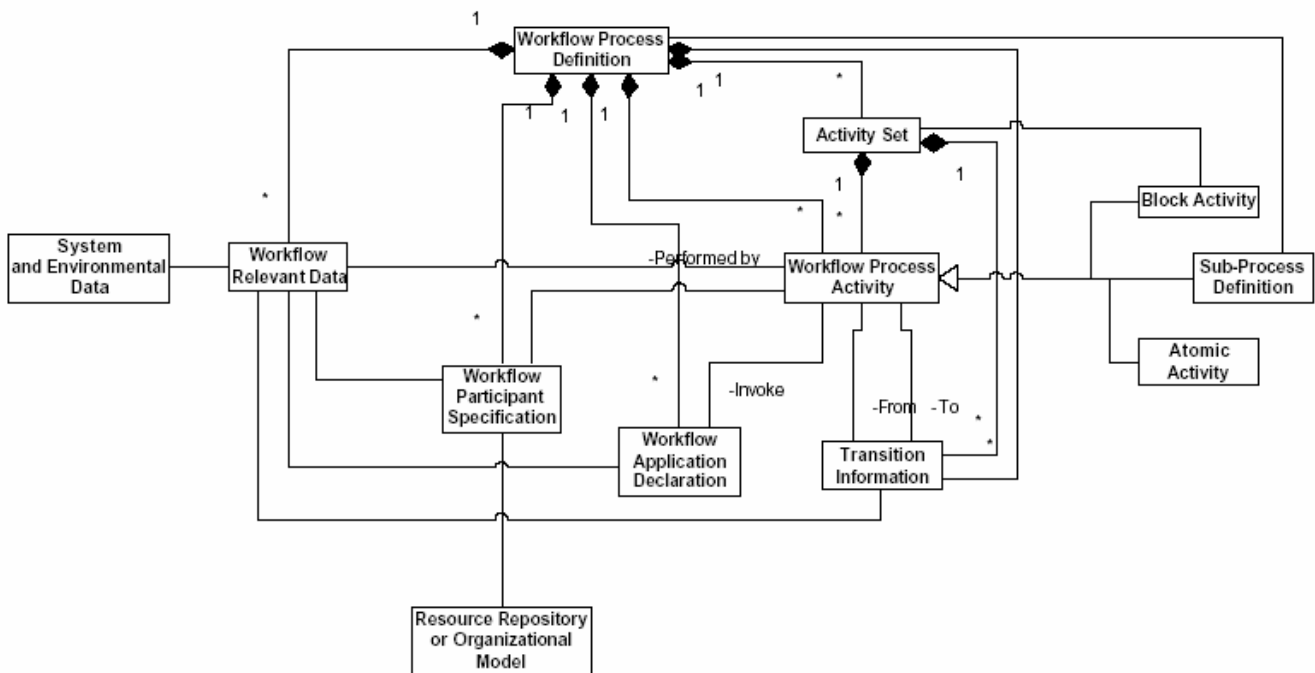


Figure 5: Workflow process definition meta-data model¹¹

The meta-data model describes commonly used entities in workflow process definitions. Attributes are used to describe the characteristics of these entities. This model is described in the XML Process definition language (XPDL) [63]. The top-level entities of the model are defined in Figure 5 [63]. The different entities are explained below:

- Workflow process definition:
 - Provides contextual information which applies to all other entities. It is also a container for the process itself.
- Workflow process activity:
 - Defines a logical, self-contained unit of work. The activity may also be a container for a separately specified sub-flow.
- Transition information:
 - Binds activities together. Each transition has three elementary properties; the from-activity, the to-activity and the condition under which the transition is made.
- Workflow participant declaration:
 - Describes the resources which function as performers in the process. The performer does not have to be a human, but it can refer to several people with similar responsibilities or a computer resource.

¹¹ Copyright @ 2002 The Workflow Management Coalition

- Resource repository:
 - Is associated with the workflow participant declaration. The workflow participant declaration may refer to a resource repository or an Organizational Model in the case of human participants.
- Workflow relevant data:
 - Refers to data created within a workflow process and used during execution of the process. The data is available to activities and applications executed while performing the process. The data may also be used to pass information from an activity to another and it may be used while evaluating transitions. The same is valid for system and environmental data. The difference is that system and environmental data is created and maintained by the workflow management system or the local system environment, while workflow relevant data is defined as part of the workflow process definition.

When implementing this interface, it brings two important advantages. Firstly, it provides independence of modelling tools and workflow run-time products. Secondly, it provides the potential to export a process definition to several different workflow products that could cooperate to provide a distributed run-time enactment service.

Our prototypes are also based on the specification for this interface. Interfaces 2, 3, 4 and 5 are not so relevant for our work regarding the implementation and design considerations. However, we will give a short description of these interfaces below to get a complete overview regarding the reference model.

Interface 2: Workflow Client Application

This interface defines standards for workflow engines to maintain work items that a workflow client presents to the user. The workflow client presents the user with work items and may also invoke appropriate applications to present the user with the task and data associated with it. Once the user has completed the task, the work item is returned to the workflow enactment service. Workflow clients may be provided as part of a complete workflow management system or a third party product or application.

Interface 3: Invoked Application

This interface defines a standard interface allowing a workflow engine to invoke a wide variety of applications. This is useful in order for workflow management systems to support complex business processes. Many workflow management systems have to deal with limited types of applications such as word processors or spreadsheets. For other types of applications, the required operations may be executed using standard interchange mechanisms such as the HTTP protocol [97]. Some workflow products use so-called “Tool Agents” that can handle the application control and information exchange. These tool agents represent a specific invocation technology, for instance: Microsoft .NET [79] or communication protocols like SOAP [85], IIOP [82], DCOM [80] or CORBA [83].

Interface 4: Workflow Interoperability

This interface defines a standard interface, which will allow workflow management systems developed by different vendors to pass work items between each other across a network. This network can be a LAN [98], WAN [99] or Internet/Intranet. In its glossary, the WfMC has defined workflow interoperability in [61] as:

The ability of two or more workflow engines to communicate and interoperate in order to coordinate and execute workflow process instances across those engines.

Different levels of interoperability are defined by the Workflow Management Coalition in which the interoperability can be achieved. These levels range from supporting simple passing of activities between different systems, to supporting complete sharing and transferring of process definitions and adoption of a common look and feel. A number of APIs have been defined to support this concept.

Interface 5: Administration & Monitoring Tools

This interface defines a standard which will allow the activity status monitoring application of one vendor to work with workflow enactment service. This will allow complete view of the status of work throughout the organisation and extended enterprise without regard to which workflow system is currently controlling the work. It will also allow users to choose better monitoring tools to work with their preferred workflow engine.

2.3 Types of workflow

According to [3], workflow systems can be segmented into the several types: Production, Administrative, Collaborative and Ad-hoc. These types are separated with respect to how they are used and what features they have.

- **Production workflow systems**
Try to achieve the highest throughput possible. The human interaction with the system is minimised and as many as possible activities are automated. The tasks are usually very repetitive.
- **Administrative workflow systems**
Focus on the definition of the process. The definition process is made as easy as possible. Many process definitions may run concurrently, sacrificing throughput, but achieving flexibility.
- **Collaborative workflow systems**
Concentrate on supporting groups working together. Process definitions can be changed often and they have a loose structure.
- **Ad hoc workflow systems**
Feature easy process definition and flexibility. This is done so that users can adapt easily to changing circumstances. Users own their own process,

which separates this type of workflows from process workflows where the organisations own the processes.

2.3.1 Summary

The presented WfMC standard and the types of workflow systems in the previous sections are statically defined, because of their usage of pre-planned definition of processes. A dynamic workflow system is essential because it will enable the workflow system to better adapt to a dynamic context rich environment. The ad hoc workflow system type is best suited for such dynamic behaviour. However, even an ad hoc workflow system does not have enough dynamic behaviour we want to satisfy the requirements in a very dynamic working environment.

2.4 Workflow process modelling

Some existing workflow management tools like Staffware [87], FlowMark [74], and TeamWare [90] concentrate on project execution and provide little or no support for process modelling and project planning. In particular, a process plan change during enactment requires a complete restart of the process in most workflow management tools. However, there are a number of other approaches in the area of process modelling and enactment research [70]. Some examples of these approaches are: Endeavors [10], Serendipity [24], EPOS¹² [34] [38], SPADE¹³ [4], Petri nets [1] [2], State Chart diagrams [68] and Activity diagrams [67]. In short, these approaches can be described as:

- Endeavors is a support system for distributed execution of (workflow) processes.
- Serendipity is a process modelling and enactment environment that supports collaborative modelling as well as execution of software processes.
- EPOS is a Software Engineering Environment with emphasis on Process Modeling, Software Configuration Management and support to cooperative work.
- The SPADE project aims at defining and developing a software engineering environment for software process modelling and enactment. Its process modelling language is based on a high-level Petri net formalism.
- Petri nets are used to model concurrent systems.
- A state chart diagram is a view of a state machine that models the changing behaviour of a state.
- Activity diagrams are a closely related modelling technique to state chart diagrams.

These approaches all have their own way to model workflow processes with their respective properties and attributes. By studying workflow modelling techniques, we will get a better understanding of workflow processes modelling. We can then make

¹² EPOS stands for Expert System for Program and System Development

¹³ SPADE stands for Software Process Analysis Design and Enactment

an informed decision as to which modelling technique is best suited to representing context-aware workflow processes.

In the following subsections, we will present three of the approaches mentioned above with some examples.

2.4.1 Petri-net

Petri-net¹⁴ is a conceptual framework that has been used to model and analyse a variety of systems ranging from operating systems to logistics systems. So far, we have seen that Workflow Management Systems (WfMSs) are software tools that support and control specified tasks in an environmental setting. However, there is no clear definition of these systems. Moreover, there is no general conceptual model for workflow management systems, like the relational data model for most database management systems. For this reason, a conceptual standard is needed for modelling workflow systems.

The formalisation of Petri-net can also be used to model and analyse workflow systems. In [1], the authors have suggested a Petri-net based approach to model workflow and workflow management systems. This example is concerned with workflows in offices and their management. As a result, they use Petri-net based analysis techniques to analyse the workflow in an office. They have also developed a prototype of a workflow management system based on the formalisation of Petri-net.

In [2], three good reasons for using a Petri-net-based WfMC are suggested by the author:

- Reason 1: Formal semantics despite the graphical nature
 - This means that business logic can be represented by a formal, but also graphical, language. To illustrate this, Figure 6 shows how the workflow primitives identified by the WfMC [62] are mapped into Petri nets. Tasks are mapped as into transitions and relations are modelled by places.

¹⁴ In this report, we assume that the reader knows the basic concepts of Petri nets. Otherwise, the reader may refer to [37] [43] [44]

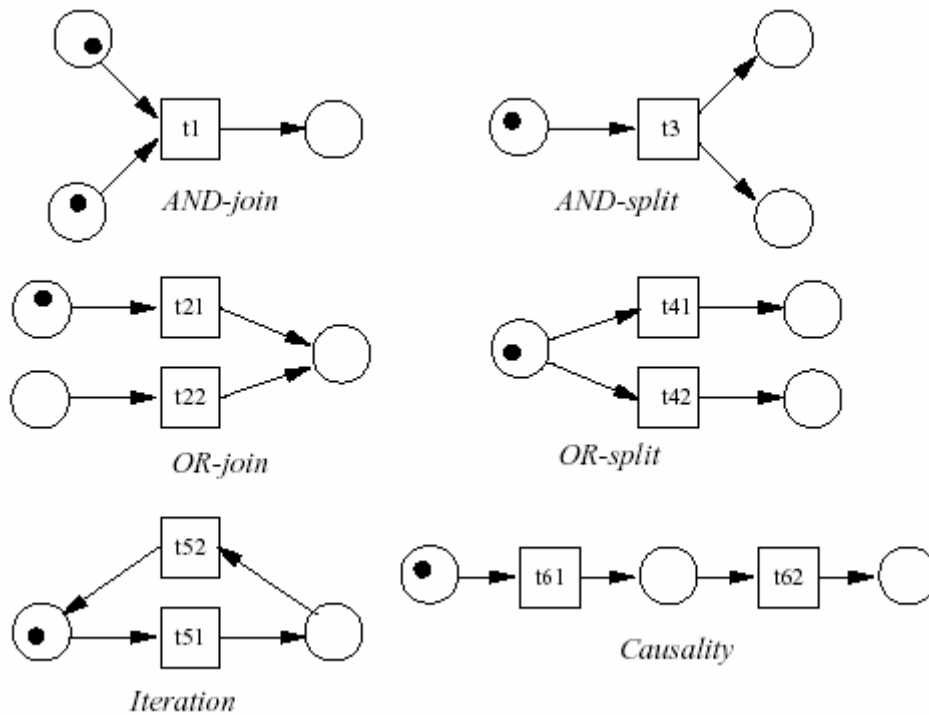


Figure 6: Workflow primitives

- Reason 2: State-based instead of event-based
 - This allows a clear distinction between the enabling of a task and the execution of a task. It is important to have this distinction because the enabling of a task does not imply that the task will be executed.
- Reason 3: Abundance of analysis techniques
 - Petri-nets provide several analysis techniques that can be used to prove properties (for instance safety/ invariance properties, etc.) and to calculate performance measures (response/ waiting times, etc.). This is useful to evaluate alternative workflows.

The graphical representation of a Petri-net consists of *places*, *transitions*, and *arcs* that connect them. *Input arcs* connect places with transitions, while *output arcs* start at a transition and end at a place. A place can contain *tokens* and transitions are components. Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled. Figure 7 shows an example of the structure of a task represented in Petri-net. This example is extended from Figure 2 that shows an order processing workflow. The task *Validation* in that workflow process is illustrated in Figure 7.

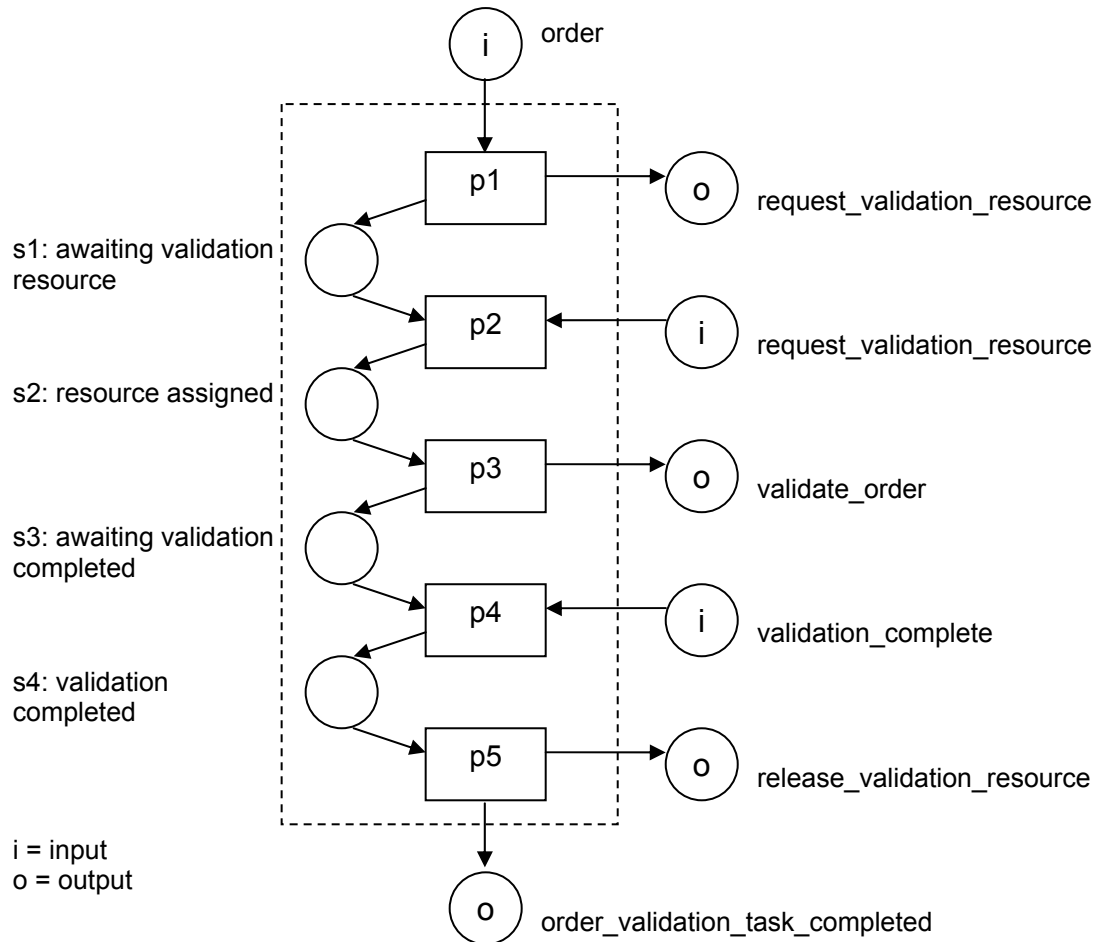


Figure 7: The structure of a task

The task in Figure 7 can be decomposed into five transitions p1, p2, p3, p4 and p5 and four places s1, s2, s3 and s4. Tasks are atomic, which mean they cannot be decomposed into other tasks. Transition p1 is triggered by a *job token* arriving via input connector **order**, which in turn sends a request to the resource manager via output connector **request_validation_resource**. The job token is then consumed by p1 and put on place s1. Each following place marks a new stage in the life-cycle of the task. Finally, at the end, transition p5 sends a token to the resource manager to indicate that the validated resource has been released. Transition p5 also returns the job token to the procedure that started the task, through output connector **order_validation_task_completed**.

A workflow procedure is represented in the Petri net illustration in Figure 8. This is the same process as illustrated in Figure 2. It is easy to see that tasks and procedures have similar connectors. A procedure is composed of tasks, control activities and subprocedures. In the illustration control activities are marked with a “c” and tasks are marked with a “t”. Subprocedures are usually marked with a “p”. Control activities are mainly used to route jobs inside the procedure.

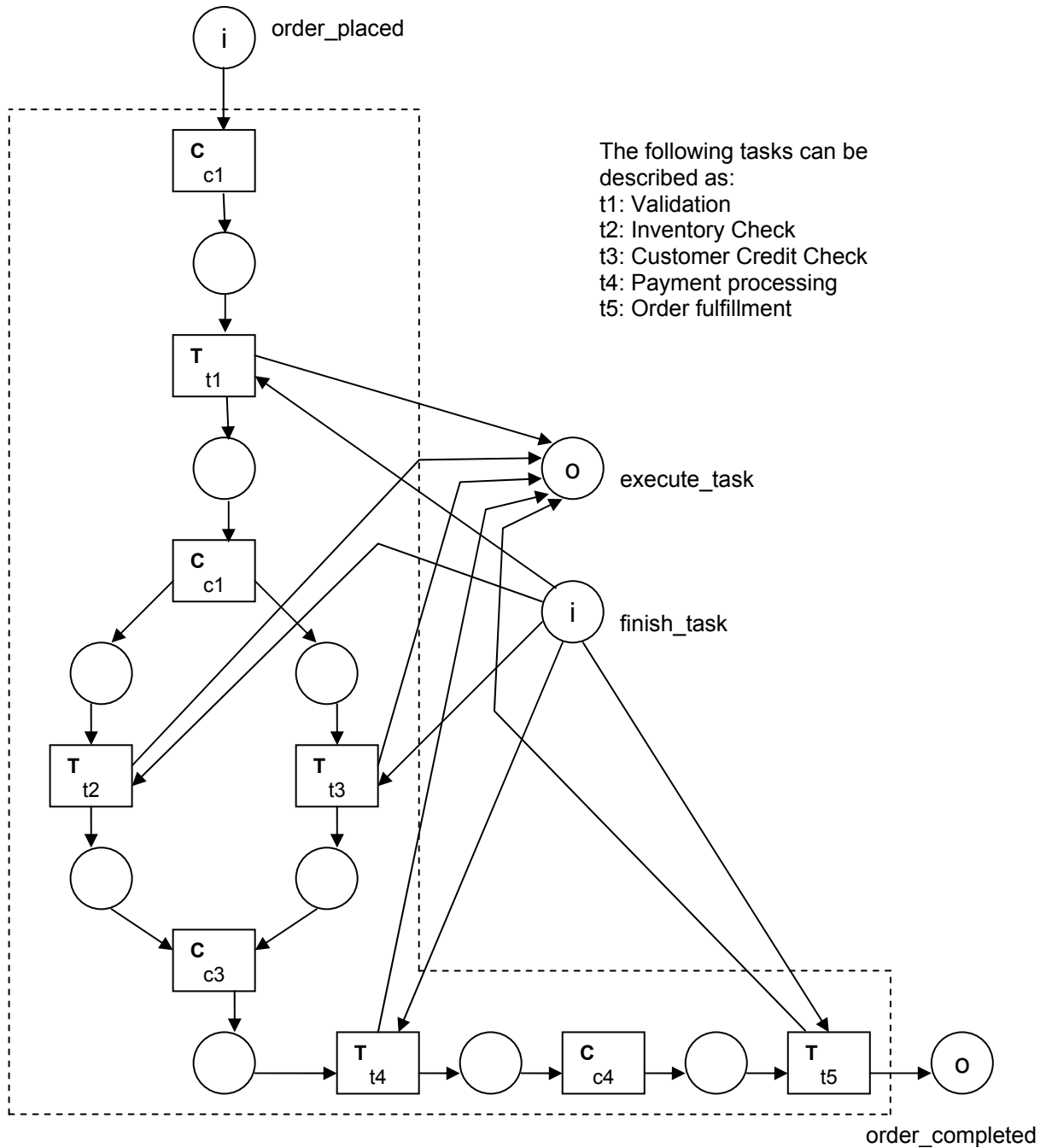


Figure 8: The structure of a procedure

The connectors for input (i) and output (o) are limited to `execute_task` and `finish_task` for this example. This decision was made because we want to model the procedure as simply as possible. With many connectors, as in the example in Figure 7, the input and output arcs that are connected to each task (t) may become difficult to follow. This is something one can take into consideration for the graphical view of process modelling, when processes become more complex.

2.4.2 State Chart Diagram

A state chart diagram¹⁵ [68] is a view of a state machine that models the changing state of a process. State chart diagrams show the various states that an object goes through, as well as the events that cause a transition from one state to another. State chart diagrams are especially useful in modelling reactive objects whose states are triggered by specific events.

State chart diagram model elements

The common model elements that state chart diagrams contain are:

- States
- Start and end states
- Transitions
- Entry, do, and exit actions

A state represents a condition during the life of an object during which it satisfies some condition or waits for some event. Start- and end-states represent the beginning or ending of a process. A state transition is a relationship between two states that indicates when an object can move the focus of control to another state once certain conditions are met. In a state chart diagram, a transition to the self element is similar to a state transition. However, it does not move the focus of control. A state transition contains the same source and target state.

Actions in a state chart diagram

Each state on a state chart diagram can contain multiple internal actions. An action is best described as a task that takes place within a state. There are four possible actions within a state:

- On entry
- On exit
- Do
- On event

Figure 9 shows an example of a state chart diagram for an order processing workflow. The example is the same process that was illustrated in Figure 2.

¹⁵ See Appendix B for a description of notations used in Unified Modeling Language (UML) state chart diagram

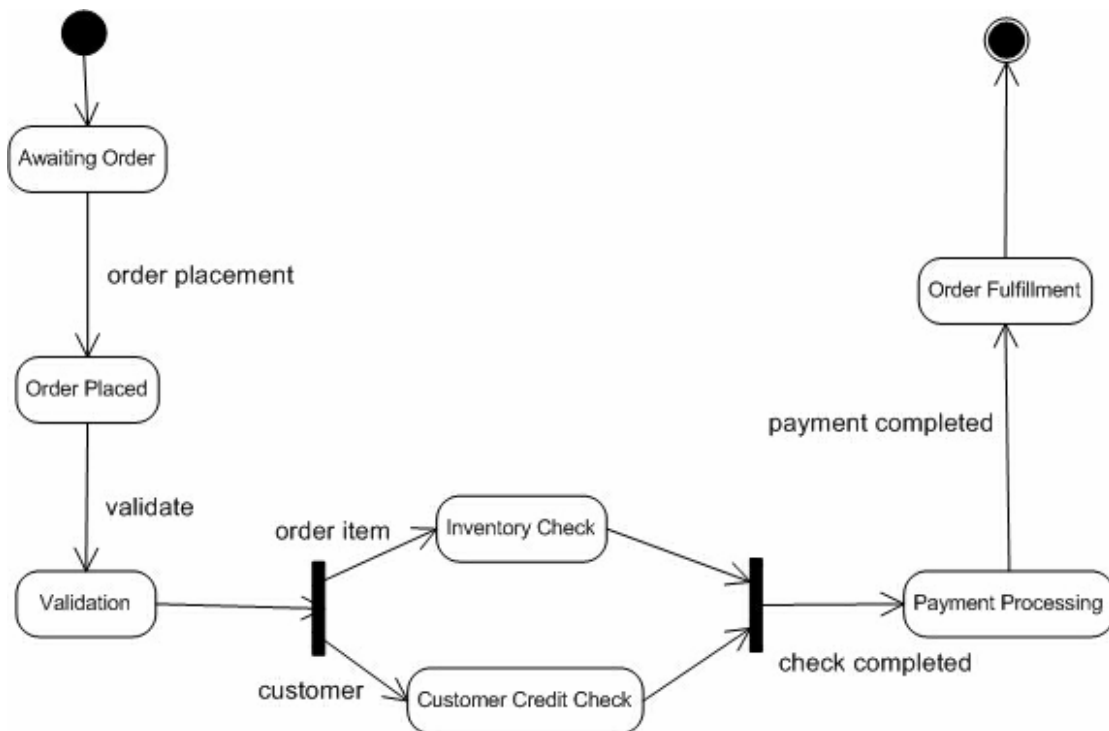


Figure 9: An UML State Chart Diagram for an order processing workflow

The rounded rectangles represent states: as seen in Figure 9. These are **Awaiting Order**, **Order Placed**, **Validation**, **Inventory Check**, **Customer Credit Check**, **Payment Processing**, and **Order Fulfillment** states. An object starts in an initial state, represented by the closed circle, and end up in a final state, represented by the bordered circle.

The arrows in Figure 9 represent transitions, which are progressions from one state to another. The notation for the labels on transitions is in the format **event [guard]/[method list]**. It is mandatory to indicate the event which causes the transition, for example **validate** or **order item**. Guards, which are optional, are conditions that must be true for the transition to be triggered. Guards can be described in any manner, including both free form text and formal language. The invocation of methods can optionally be indicated on transitions. The order in the listing implies the order in which they are invoked.

The notation used within states is the same as that used on transitions, the only difference being that the method list is mandatory and the event is optional. Had there been no event indicated, those methods would have been invoked continuously (in a loop), whenever the object is in that state. Methods to be invoked, when the object enters the state, can be indicated by the keyword **entry**. Methods to be invoked as the object exits the state can be indicated by the keyword **exit**. The capability to indicate method invocations, when you enter and exit a state is useful because it enables you to avoid documenting the same method several times on each of the transitions that enter or exit the state, respectively.

Transitions are the result of the invocation of a method that causes a change in state. However, we can also have recursive transitions, also called self transitions that start and end in the same state.

2.4.3 Activity diagram

Activity diagrams¹⁶ [67] are a closely related modelling technique to state chart diagrams. State chart diagrams model how an object changes state in response to external stimuli. The activity diagrams model how an object changes state in response to internal events. This means that all states are action states. This technique is commonly used to model workflow processes. Figure 10 shows an example of a state chart diagram for an order processing workflow.

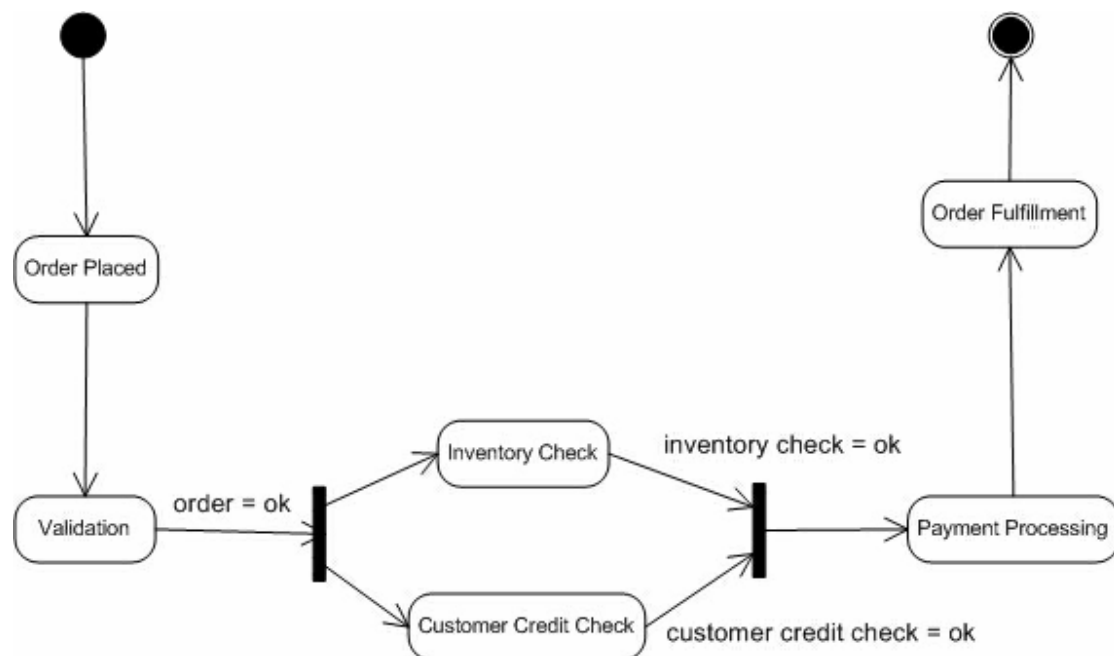


Figure 10: An UML activity diagram for an order processing workflow

It is obvious that the activity diagram uses many of the same notations as the state chart diagram. This has to do with the fact that an activity diagram is a special case of a state chart diagram. As seen in Figure 10, transitions or control flow as they are called in activity diagrams, are labelled with conditions to be satisfied.

2.4.4 Summary

A difference between the Petri-net modelling of workflow processes and the workflow process modelling used by WfMC [63], which is an activity diagram modelling technique [67], is that each activity is decomposed into several transitions

¹⁶ See Appendix C for a description of notations used in Unified Modeling Language (UML) activity diagram

and places for the Petri-net approach. This allows Petri-net models the advantage of specifying the flow of an activity at a much more detailed level than the activity approach from WfMC. We did not find detailed activity modelling essential for our work, so we decided to go for activity diagram modelling. This technique is good for illustrating the flow between action states of an object, which is exactly what we want to do. The flow of an activity can also be drawn as a sub flow using the activity diagram technique. Since we need to represent processes driven by external events, we will also make use of the state chart diagram modelling technique.

2.5 Exception handling

Exception situations arise in workflow systems as in any other computer system. It is therefore necessary to study how exceptions in a workflow system can be handled, without requiring human intervention.

The WfMC definition of Interface 1 [63] defines exception transitions on the same level as normal transitions. These transitions functions as normal transitions, but are called exceptions, since they deviate from the expected workflow process enactment. Transitions defined as exceptions are handled by the workflow enactment engine, like any other transition.

There are also other ways of handling exceptions in workflow systems. There are two types of exceptions in workflow systems in addition to what has been defined on the workflow process definition level: expected exceptions [16] and unexpected exceptions. These exceptions are handled by the exception handler of the workflow system. Expected exceptions are exception situations known in advance by the workflow planner. Unexpected exceptions are not known and usually require intervention by humans. Expected exceptions can be handled by the exception handler using the semantics of the workflow system. The exception handler usually uses some form of reactive processing to handle expected exceptions. Expected exceptions are unpredictable, asynchronous and may require special treatment. This makes these exceptions hard to represent in workflow process definitions.

Workflow systems are gradually starting to support expected exceptions. Commercially available workflow systems usually support a limited number of such expected exceptions. In [14], a new approach to exception handling of expected exceptions is presented. This approach includes a new language for expressing expected exceptions, called Chimera-Exc [14], and a way of integrating the exception handler with the workflow system, called FORO Active Rule component (FAR) [14]. FORO is a workflow management system.

The exception handling mechanism defined in [14] captures exceptions and reacts to them. The approach suggested has much in common with trigger management strategy used in active databases [12] [13]. Active rules defined for active databases are also representative for expected exceptions. The common characteristics are:

- Event part, which defines the symptoms of the exception.

- Condition part, which defines a check whether the event really constitutes an exception.
- Action part, which describes what corrective action(s) to be taken.

Each rule is executed in a separate transactional context.

The exceptions need to access the state of workflow processes. This state is shared between the workflow enactment engine and the exception handler. The authors of [14] therefore introduce a schema definition language for managing state information about workflows. The schema is a simple, object-oriented schema, consisting of object classes. The classes defined in Chimera-Exc are:

- Workflow management classes, which store meta information about workflows and their enactment.
- Exception management classes, which store meta information about exceptions and their management
- Workflow specific classes, which store values of variables defined within workflow schemas.

Events, conditions, actions and priorities are central in Chimera-Exc. Below follows a closer description of these parts:

- Events
 - Each event can monitor several events.
 - Belong to one of 4 classes:
 - Data manipulation events, which enable the monitoring of operation that change the content of the database used.
 - External events, which are raised by external applications interacting with the exception handler
 - Temporal events, which can be instant or periodic events.
 - Workflow events, which enable monitoring of workflow processes and activities.
- Conditions
 - Verify that rule triggering really constitutes an actual exception.
 - True conditions are conditions, which needs to be handled, while false conditions are false alarms.
- Actions
 - Define one or more primitives to be executed in order.
 - Primitives are divided into two main categories:
 - Data modification primitives, which can be create, modify or delete primitives.
 - Workflow management primitives, which are used to initiate actions within the workflow enactment service.
- Priorities
 - Define the order of execution within a triggered rule set.

After rules have been created, they are stored within in a rule repository. A rule only becomes triggered, when an event belonging to that rule occurs. A **Scheduler** process is responsible for starting the execution of the rules. This process responds to events or is activated periodically. Before the actual execution takes place, the scheduler

process has to determine which rules have been triggered. These are then placed in a ready queue before execution. The Scheduler also handles ordering according to priority.

Exceptions defined in Chimera-Exc [14] are handled by FAR. The overall architecture of FAR is shown in Figure 11 [14].

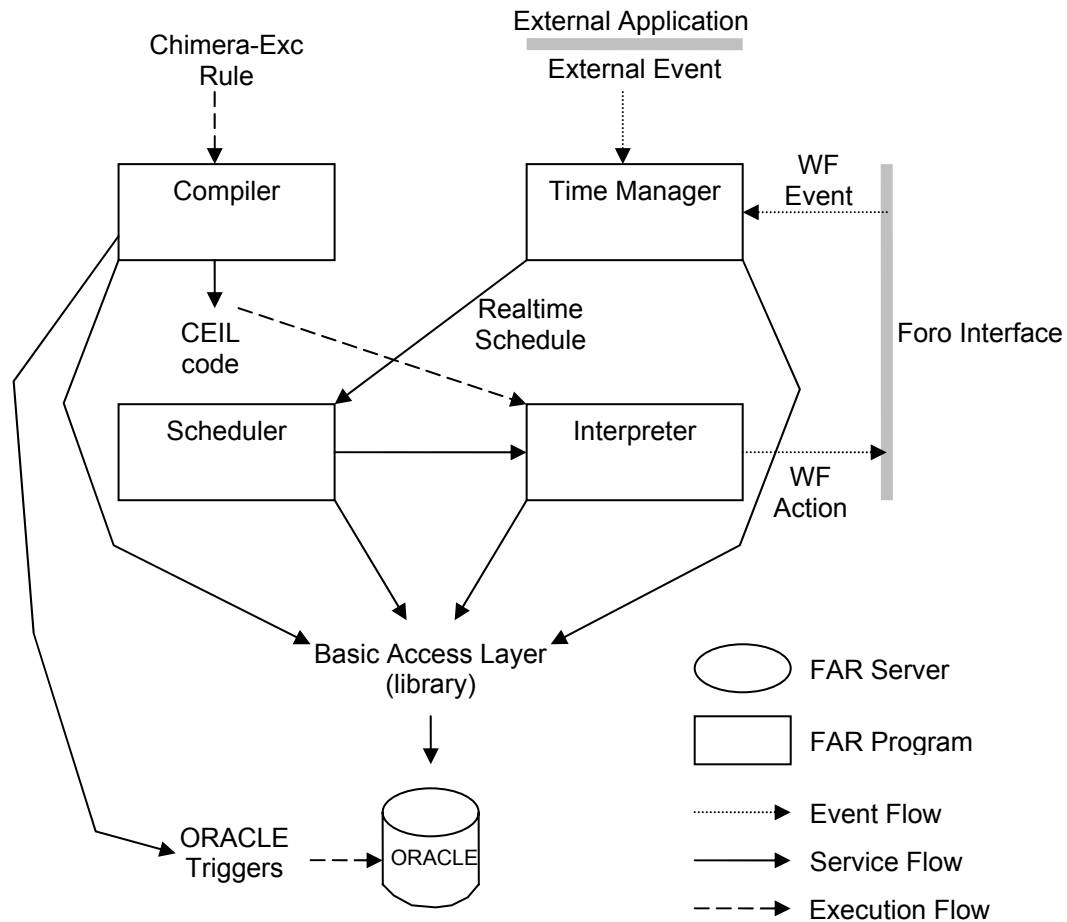


Figure 11: Overall FAR architecture

The main components are the **Compiler**, **Time Manager**, **Scheduler** and **Interpreter**. Below follows a short description of each of these components.

The **Compiler** is responsible for translating rules written in Chimera-Exc [14] to an internal representation. It also produces relational triggers to capture data events directly inside the database.

The **Time Manager** is responsible for management of time dependent events. This includes workflow and external events.

The **Scheduler** is responsible for ordering rules according to priority and submitting them to the **Interpreter** for execution. It is activated periodically or in response to real-time rule triggering.

The **Interpreter** is responsible for executing rules, with some degree of parallelism.

2.5.1 Summary

Chimera-Exc [14] and the FAR [14] architecture provide flexible mechanisms for handling exceptions, which arise through workflow enactment. By using this approach one avoid having to model each exception situation in the workflow process description. One can specify a wide range of exception situations with actions changing both the state and possibly the description of a workflow process.

3. Context-awareness

An introduction to context and awareness will be presented in this chapter. Further, context sensing, different techniques for context information modelling in computer systems, and sentient objects which represent an approach to context-awareness in distributed computer systems will also be presented.

3.1 Introduction to context-awareness

Several definitions for *context* have been put forward in the literature, serving different purposes. In Information Bases [55], context describes a group of conceptual entities from a particular standpoint. In Artificial Intelligence [36], context appears as a means of partitioning a knowledge base into manageable sets or as logical construct that facilitates reasoning activities.

Although context has already been subject of investigation in different fields, only recently has this notion been explored for ubiquitous computing. Most of the initial efforts for defining context in ubiquitous computing were specific for certain kinds of context - location and time being the most obvious examples. Schilit and Theimer [47] claimed in 1994 that the important aspects of context were the user location and identities of nearby people. Brown et al. [11] and Ryan et al. [45] gave their definition in terms of examples of context information instead of generalising the concept. Since the number of examples that can be given is limited, the application of this definition is also limited.

Schilit et al. [47] claim that the important aspects of context are where you are, who you are with, and what resources are nearby. They define context to be the changing environment. The environment is composed by the following views:

- Computing environment: e.g., available processors, devices accessible for user input and display, network capacity, connectivity and costs of computing.
- User environment: e.g., location, collection of nearby people and social situation.
- Physical environment: e.g., lighting and noise level.

Also this definition turned out to be too specific. It was necessary to give definitions without having to enumerate examples of context because the user experience changes from situation to situation. For those reasons, Dey and Abowd in [18] came up with a more generic definition of context, which is:

“Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to

the interaction between a user and an application, including the user and applications themselves”.

Context-awareness seeks to exploit human-computer interactions, by providing computing devices with knowledge of the users’ environment, i.e. with context.

Awareness of the context can potentially be used to diminish the amount of explicit input a user is required to give to a computing system. Contextual information about the current activity, what the user knows and what the user and system capabilities are, can greatly simplify the user scenario. Such manipulation of contextual information can also be used to reduce the teaching needed, for the user to accomplish an activity.

A definition of what it means for a computer system to be context-aware has been provided by [18]:

“A system is context-aware if it uses context to provide relevant information and/or services, where relevancy depends on the user’s task.”

This definition has use in the mobile computing field, where context is to a high degree the deciding factor of the level of service when providing information and services to mobile users.

3.2 Context sensing

A context-aware workflow system using mobile equipment in its workflow enactment requires sensor technology beyond what commonly available today. Sensors are usually hardwired to the system using the sensed information where all the processing of the sensed values takes place. In other words, the sensors function as “dumb sensors”. Examples of such sensors are thermometers and pressure sensors. This approach to context sensing is inappropriate when dealing with distributed and mobile context-aware systems. Mobile equipment must be able to acquire the context information from the environment without using wires.

The development of better sensing technology is a requirement for achieving the vision of ubiquitous computing, which is presented by Mark Weiser in [58]. Several strides have been made in context sensing technology. **Smart sensors** [46] are such an advancement. **Smart sensors** have built in memory, short-range wireless transceivers and a small battery. Using this technology, it is possible to develop mobile context-aware systems, which are capable of sensing their environmental surroundings. Another advancement of the **smart sensor** technology is **smart dust** [46]. This technology allows for self configuring sensors, which can be scattered throughout the environment. The sensors would then set up a wireless network through which sensed context information can be extracted. The goal is that these sensors should be so cheap that they can be discarded once their batteries run out.

3.3 Model context information in computer systems

There are two main challenges when building a context-aware computer system. Firstly, the system architectures have to be decided for how to map the context data into a computer readable representation. This means that raw context data such as sensed values from sensors have to be modelled in a computer supported representation. To accomplish this, it is important to decide which information should be included in a context-aware system. It is therefore necessary to adapt some form of context classification, which separates the different types of context information. This challenge is dealt from sections 3.3.1 to 3.3.7. We will first present several context classifications, and we will then move on to several examples of how context information is modelled.

The next main challenge when building a context-aware computer system is to actually acquire sensor output and map this into a context information model, which is then made available for the main context-aware application. Section 3.4 deals with examples to solutions of this challenge.

3.3.1 Context classifications

A context information classification is presented by Dey et al. in [33]. The context is classified by the entities the context addresses and categories of context information. The most significant entities defined are **places**, **people** and **things**. **Places** refer to geographical locations such as a room. **People** can be single individuals or groups of people. **Things** can both be physical objects and software components. The categories of context information are **identity**, **location**, **status** and **time**. **Identity** refers to a namespace unique identifier of an entity. **Location** is not simply limited to geographical location, but also aspects like orientation, elevation, co-location and proximity are part of location. **Status** refers to the relevant information that can be sensed about an entity. **Time** is used to characterise a situation. Together with other pieces of context information, time can be used to produce historical information.

Schmidt et al. present a different classification in [48]. In this classification we have two main categories of context information: human factors and physical environment, which each has three subcategories. Orthogonal to these categories, context history provides another dimension of context information.

3.3.2 Entity-relationship model

In [27], a solution for sensor fusion of context information from several sensors is presented. The architecture presented uses a bit more pragmatic context classification than the classifications presented earlier. The classification has three main categories:

1. Environment
2. The activity the user is currently performing
3. The users own physiological state.

The context model used in the architecture is based on the entity-relationship model [27], which is represented in a relational database. This solution for context modelling reflects the user-centred and application-oriented design philosophy the authors of [32] have adopted.

The application scenario presented in [27] is a small group of users who frequently use a conference room. Basic information about the users and the conference room is predefined, while presence and user's activity are dynamic based on context information. In Figure 12 [27], the entity relationships between the conference room and user, and user and activity are presented.

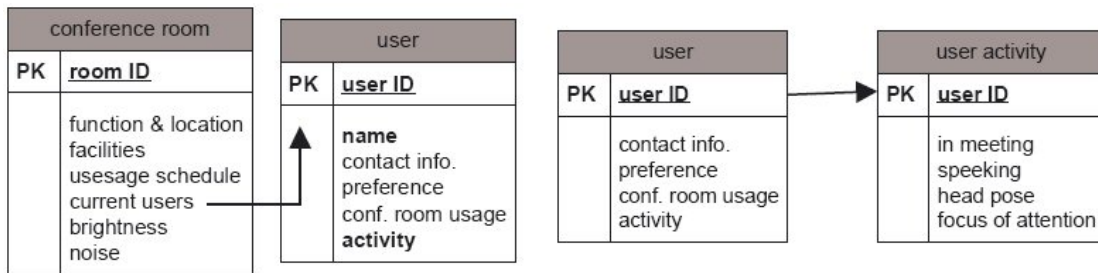


Figure 12: Entity relationships between key entities

3.3.3 Object oriented model

A system for network-centric context-aware support of mobile users in next generation networks is presented in [30]. This system is called Transparent Enterprise Access for Nomadic Users (TEANU). The system supports device migration between networks and context-aware services related to these migrations.

The solution for context modelling in [30] adopts the context classification by Dey et al. in [33]. Based on this classification a classification of context for an entity is developed for the context-aware service. This is illustrated in Figure 13 [30].

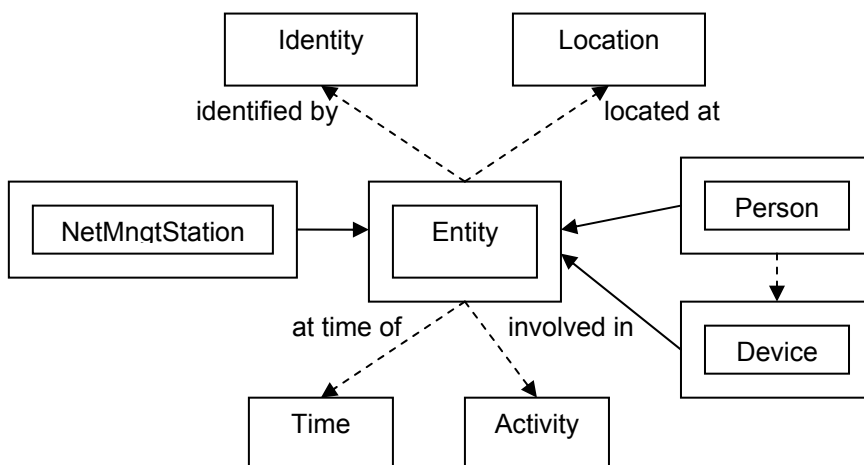


Figure 13: Classification of context for an entity

The TEANU system uses policy-based network Management (PBNM) [50] [75] as a means of flexible configuration of network elements. The use of policies levitates the network administrator of configuring every single device manually. Instead multiple network elements can be reconfigured by changing or creating new policies. The policy method is well suited for context, since context is usually complex, fluctuating and layered. The policy language is an English-like declarative language, which uses a rule-based format with “if-then” condition and action relationships. The policies are represented as a class hierarchy. The context representation is part of this class hierarchy as illustrated in Figure 14 [30].

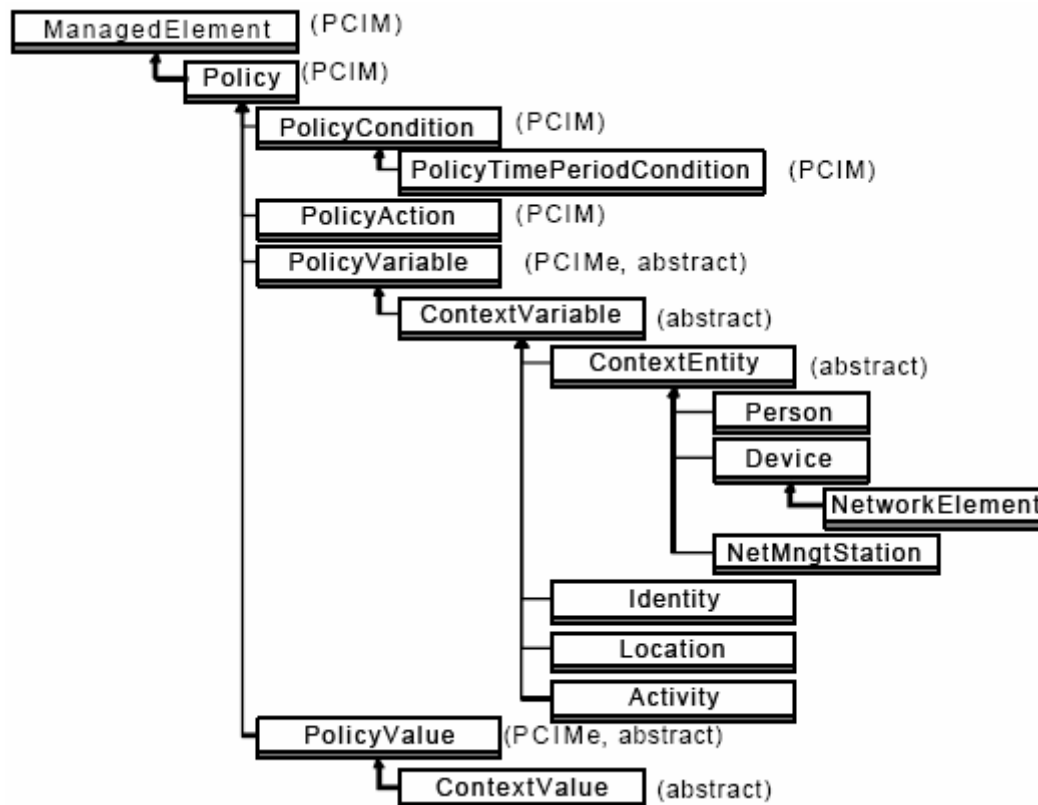


Figure 14: Class hierarchy for the context information model

3.3.4 World model

Requirements for context management in the Georgia Tech’s Aware Home environment are presented in [31]. The vision is that smart everyday devices communicate and cooperate to provide services and information to users. This means that the devices need to have a common representation of context.

It is important that applications can access context information in the context model. This access is often based on identity and time or location and time as indexes for context information. In [6], a context model using only location as index is presented. The index represents the spatial relations between entities. Location based context models can be divided up into topographical, topological and hybrid models. Topographical models use geometry to model space. Topological model describes the

relations between spatial objects directly without placing them in a coordinate system. Hybrid models combine these approaches. The context model in [6] is based on a spatial structure of the real world, and is therefore called a world model.

3.3.5 ContextMap model

In [33], a scenegraph schema, called a ContextMap, is presented for context information modelling. This solution for context information modelling has been developed to satisfy the need for a consistent way to model context information and address correlation between and ambiguity of context data.

The ContextMap model separates the context information of an entity into intrinsic and relational context attributes. Intrinsic attributes can be represented without referring to others. Relational attributes can not be represented without specifying the relation to other entities.

A ContextMap is a directed acyclic graph, which is the traditional solution for scenegraph. The attributes are collected through depth-first traversal. The ContextMap represents a view of the world, which can be shared between several applications.

Entities are represented as nodes in the graph. Each node maintains its own intrinsic attributes. Relational attributes are represented as edges in the graph.

Activity is a type of node in the graph. This node represents the social semantics of one or more **entities** in the graph.

Place nodes represent **entities** which are places in the graph. This can be large regions or small areas.

Object nodes represent physical objects. These nodes can have “contain” relationships to its sub nodes.

Person nodes represent people entities in the graph. Edges from a **Person** node represents “conduct” or “use” relationships.

An example of a ContextMap is illustrated in Figure 15 [33]. In this illustration, Place nodes are represented as rectangles. Activity nodes are represented as diamonds. People nodes are represented as ellipses. Object nodes are represented as ellipses in grey.

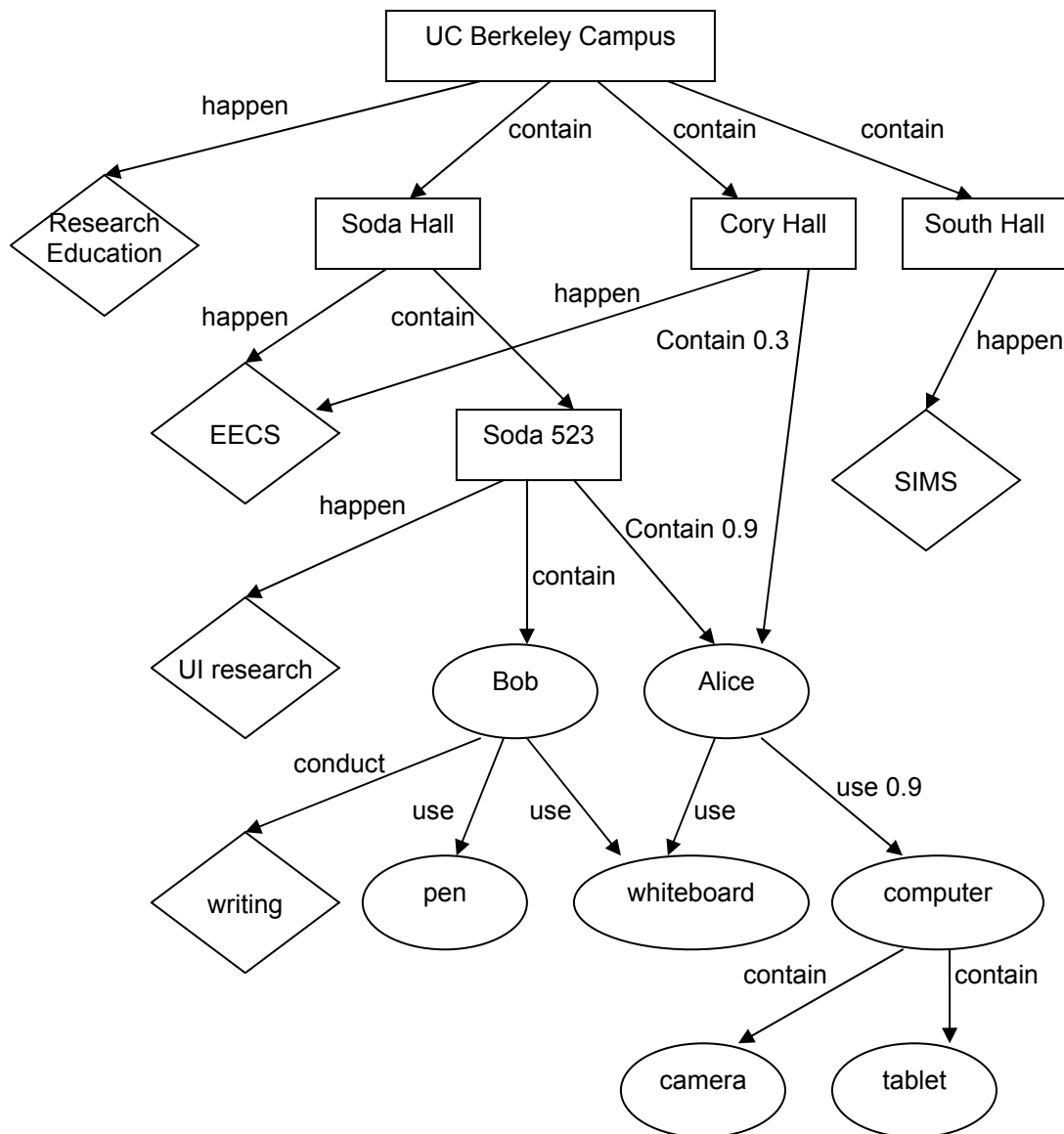


Figure 15: An example of a ContextMap

The ContextMap model also handles context information ambiguity by tagging edges in the map and intrinsic attributes with confidence values. These values are then used to calculate the confidence of the sensed context.

3.3.6 Event representation

Representation of context information as events separates itself from the solutions presented above since this solution does not represent context as an overall model. Instead this solution focuses on the fact that context-aware applications need to respond to changes in context. This means that context-aware applications need to have an event-driven structure, and that context changes need to be represented as events.

Events need to be represented in a format that both the event publisher and the event receiver can agree on. This means that the format can be almost any computer interpretable format. However, there exist four typical formats for events [15], which are presented below.

Events can have a format which is encoded binary. In this format, events are represented as a simple data structure, like a C “struct”.

Object representation allows for a more complex data structure utilising the object oriented facilities of a high level programming language.

Attribute-value event representation is the most limiting solution. This has to do with the difficulty of creating a good structure. However, the solution is language- and platform independent.

XML encoding of events has the advantage of enhancing interoperability and extensibility of events, but XML also requires substantial processing and bandwidth overhead. In [20], an XML based solution for event format is presented. This solution uses the resource description framework (RDF) [102]. RDF has a simple and powerful model and syntax definition. This means that RDF is a good choice for delivering sensed context information irrespective of the application using the context. The RDF model consists of three object types, which form subject, predicate and object triples, called RDF Statements. These object types are:

1. Resources: All objects described in RDF are called resources.
2. Properties: Properties describe the attributes of the resources.
3. Statements: Subject (resource), predicate (property) and the property value (object) triple, build an RDF statement.

3.3.7 Summary

The presented context modelling techniques are closely linked with the context classification supported and the context information, which has been considered in our approach. It is therefore necessary to consider the application domain carefully before deciding on a context modelling technique. The context modelling technique used should provide easy access to the context information where it is needed.

3.4 Sentient objects

Sentient objects represent an approach to context-awareness in distributed computer systems. Cheap sensors can be distributed throughout an environment and can be connected through wireless networking. Mobile software components, called sentient objects, can respond to events from sensors autonomously and act through actuators. To allow the sentient objects to respond intelligently, these objects need to have built in logic. In [21], a model for development of sentient objects is presented in addition to definitions of the involved entities.

Sentient objects acquire context information input from sensors and respond to changes in context information through output actuators. This means that the only interfaces a sentient object has are sensors and actuators. An object model of a sentient object is illustrated in Figure 16 [21].

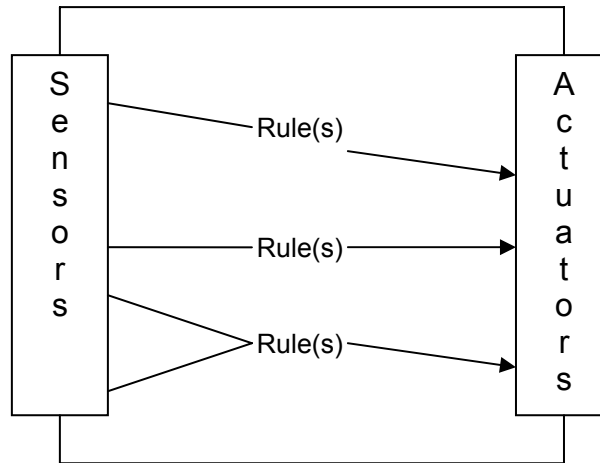


Figure 16: Simple sentient object model

A sentient object can perform meaningful actions based on context information through built-in logic, such as a rule based inference engine. The communication between sentient objects is event based allowing for loose coupling between objects. All events handled by sentient object are software events. This means that all events registered by sensors have to be converted into software events by the sensors. Actuators, which change the state of the environment, have to receive the software events from sentient objects and act accordingly. In short, a sentient object consumes software events from sensors, passes the events through control logic and produces new software events, which the actuators handle.

The context awareness in sentient objects comes from the fact that they sense the environment through sensors and act on the environment through actuators. The first step in achieving context awareness in a sentient object is the capture of sensor data. A sentient object must perform sensor fusion to get an overview of the environment. Secondly, the context data from sensors must be represented in a usable form. This can be done in the sensor or in the sentient object. The sentient object must then perform context reasoning before new software events are generated for the actuators. The inference component is responsible for the context reasoning. A knowledge base used within the inference component forms the basis for the reasoning. This knowledge base contains rules, which the sentient object can use in its reasoning.

3.4.1 Summary

The sentient object model is interesting in a context-aware workflow system both for the underlying context information framework and the workflow system components themselves. The ability to react based on sensor input through the use of a rule based inference engine and actuators can be useful when reacting in response to contextual conditions. Components in the underlying context information framework can be used

to initiate actions in a workflow system if the sentient object model is used for those components. Workflow clients, functioning as sentient objects, can also use own contextual conditions to initiate actions in other workflow components and in the environment. A workflow enactment service may also use the sentient object abstraction to respond in a similar fashion.

4. Situated actions and planning

This chapter will concentrate on situated planning and how this influence workflow systems. However, we will introduce another important term, situated actions, before we start describing situated planning. We will include a brief description of the differences between Trukese navigators and European navigators, which was first written by Thomas Galdwin and presented in [9]. European navigators start with a plan or a course, which has been charted according to well known principles. Every action the European navigator takes is related to that plan. If something unexpected happens, the plan is first changed before action is taken. The Trukese navigator starts with the goal instead of the plan. Information from the environment, like the wind, the weather and the sea, influence the decisions of the navigator. The navigator responds to changes in the environment in an ad hoc fashion. At any moment the Trukese navigator can describe the current goal, but not a plan or course. This example illustrates very well the difference between actions done *in situ* in the case of the Trukese navigator and the pre planned actions performed by the European navigator.

Workflow systems utilise predefined process models for controlling work. This approach has been criticised, because of the static, rigid representation of work processes [53] [59] [7] [25].

In [54] by Suchman, the difference between actual work and representations of work are highlighted. Suchman argues that work is essentially ad hoc and situated, and planning must therefore be a pre activity task or plans must be constructed after the activity is finished. This means that plans become resources for the work instead of controlling the work. In [5], a case is made for plans as mechanisms for giving order to work. To illustrate this, a scenario from a hospital is given. Patient's diagnosis and treatment plans are essential for collaboration and coordination between several health professionals. Without such plans it would be necessary with communication to inform everyone about the patient, the condition the patient has and how the physician intends to treat the patient. While considering both these aspects of plans we are left with a *planning paradox* [5]. On one side, work has an ad hoc nature, so plans do not form the basis for work. On the other side, plans do play an important part in giving order to work in almost any organisation. Based on this, one can say that plans are made out of situated action.

Activity theory is a philosophical framework for studying human work practise. This includes both the individual and social level. In [56], three main characteristics of human activity are presented:

- Directed towards a material or object
- Mediated by artefacts
- Social within a culture

Activity theory states that human activity is a hierarchy of three levels. **Activities** are realised through the use of **actions**, which are carried out by **operations**. The **motive** for an activity stems from the reflection of and expectation to, a material or ideal object. Actions result in objective **results**. Humans have anticipations for results of an action, and these anticipations form the **goals** for a human performing an activity. An activity exists as one or more actions, but the activity and action are not identical. The **conditions** of a concrete situation govern how an action can be performed. This means that actions are realised through a series of operations, where each operation is adapted to the physical conditions of the action. At all three levels activities are guided by anticipation.

Based on previous stated definitions and explanations of current workflow standards, it is easy to see that workflow systems do not handle unforeseen events and breakdowns easily. A lot of work has gone into making workflow systems capable of handling exceptions. However, the main point in [5] is that unforeseen situations are not exceptions, but are important parts of any activity. These situations instead serve to develop and enhance plans for future actions. The plan is a central resource in execution of activities and is enhanced based on the experience obtained during the execution of activities. These points are illustrated in [5] with a prototype called the PATIENT SCHEDULER. This prototype illustrates how coordination of patient care within hospitals can be supported by computer technology. Activity theory gives us a new definition of a plan [5]:

“cognitive or material artefact which supports the anticipatory reflection of future goals for actions, based on experience about recurrent structures in life”.

Based on the definition, one can see that a major challenge for planning tools is to support the anticipation of recurrent events in human work. This anticipation must also be used in human work. The author of [5] has used this conceptualisation of human activities and the experience gained from the PATIENT SCHEDULER prototype, to discover some guidelines for design of computer support for planning:

- Producing and altering plans in the course of work:
In order for plans to become resources for future realisation of an activity, the plan should be made as part of the activity.
- Sharing plans within a work practice:
Plans function as coordination mechanisms between several actors involved in an activity.
- Executing plans according to the conditions of the work:
One should consider the difference between plans as anticipated results of actions and the realisation of these actions as operations according to the conditions of the situation.
- Inspecting plans and their potential outcome:
All plans within a work practice should be available for inspection. The potential outcome for applying a particular plan should also be revealed.
- Monitoring the execution of plans:
It is important to monitor the progress in work according to the plan.

Plans as sequence of actions are central to human work. However plans must be realised according to contextual conditions. Plans function as a way to anticipate and

pre handle recurring events, and to store for reuse the experience gained from handling these events. This understanding of plans implies that workflow systems should support mediation of the anticipatory reflection of recurrent events in human work, instead of simply supporting routing of information. So, a planning tool should support building, altering, sharing executing and monitoring plans; situated planning in other words. A planning tool should therefore not promote a rigid match between process models and work.

Workflow systems are essentially polymotivated. On one side we have workflow systems as the mediator of work. On the side we have workflow systems as technology of accountability. A definition of “technology of accountability” was given in [53]:

“By technologies of accountability I mean systems aimed at the inscription and documentation of actions to which parties are accountable [...] in the sense represented by the bookkeeper’s ledger, the record of accounts paid and those still outstanding”.

The accountability aspect is primarily a concern of the management and not the main goal of the organisation. From an Activity theory perspective, both these aspects should be considered and satisfied if possible.

4.1 Summary

There are several research challenges connected to workflow systems supporting situated activities and situated planning. The following list presents some of the relevant research challenges:

- How can a workflow system support situated process and activity planning?
- How can situated process and activity plans be recorded, to allow the workflow system to function as a “technology of accountability” [53].

5. Mobile workgroups with computer based support of their work

This chapter describes an application that has been implemented for supporting companies and their employees in work requiring mobility. By looking at how mobile computer systems are used by mobile workgroups, we achieve an understanding of what functionality is already present and what functionality is needed. We are also able to get an understanding of what mobile computing equipment is used in the industry today.

5.1 HandyMan

HandyMan is a software application created by ePocket Solutions ASA to support electricians in their work. The system was created for the Pocket PC platform. It was tailored for electricians since their work usually involves paper work that later have to be entered into a computer system.

HandyMan handles memos, time spent on particular work activities and material management. In addition, it has workflow process management through the use of check lists. These check lists help the electrician to manage all parts of work order, to be able to complete the work order. The main functionalities of the HandyMan system are the following functions:

- **Tasks:** The user can access all information about each task to be carried out next. This includes information such as name and address of the customer, office messages and priority of the task. This function also keeps track of the work progression by making sure the user follows a pre-defined workflow.
- **Inventory and orders:** This function allows the user to access all information about inventory in the service car, the main company stock, and the merchants. If a certain product is not available in the service car, an order can be sent to the main company or a merchant.
- **Hour usage:** This function allows the electrician to register the time spent on each task. It makes it possible to register how much should be charged. HandyMan also provides all information necessary to make an invoice.
- **Synchronisation:** The system allows for synchronisation with both the customer support and the economy system.
- **Preferences:** This function allows for tailoring of each Handyman for each individual user.

An illustration of the HandyMan system is shown in Figure 17 [57].

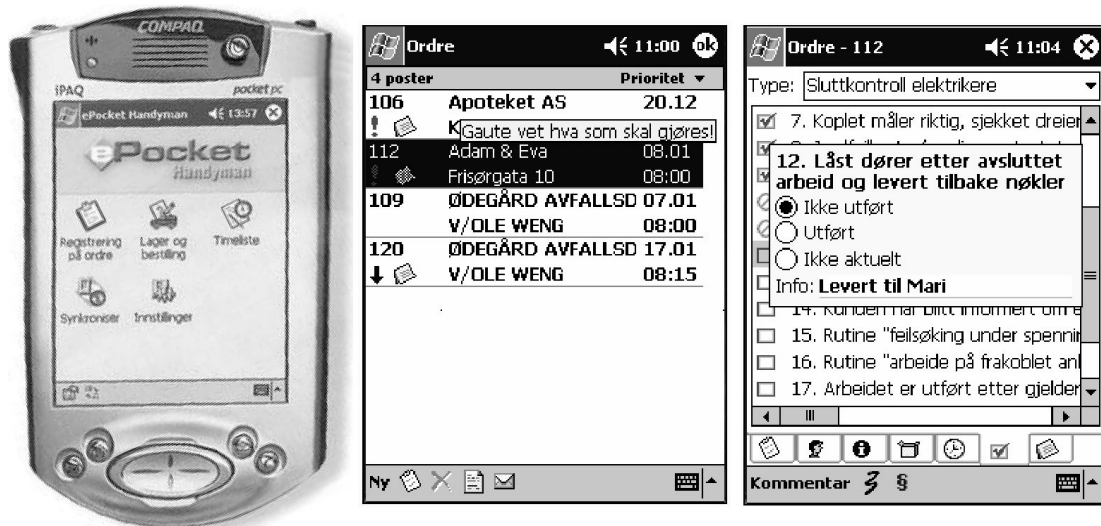


Figure 17: Screenshot from HandyMan

In [57], a case study was presented of the HandyMan system. This study looked at how the system performed in five Norwegian electrician companies. The goal of the case study was:

“Analyse the usage of tool HandyMan for the purpose of identifying requirements and problems for support systems for mobile work from the perspective of software developer in the context of the working environment of electricians.”

The data for the case study was collected through interviews with employees at each of the five companies.

Based on the answers provided by the electricians, the authors of [57] found five guidelines that must be considered when designing a system for mobile work:

1. **Working environment and device:** One have to consider the working environment before starting to design a system for mobile work. The working environment has to be compared against the available devices. In some cases it might not be possible to find a usable mobile device.
2. **The system must give the worker additional functionality:** It is important the system provides additional useful support that makes the work more efficient for the employees using the system.
3. **Usability must be top priority:** Usability is often more important on mobile devices than on desktops because of the small screen and limited input devices.
4. **A mobile work tool must be flexible:** Since mobile work often is characterised by ad-hoc work, it is important that a mobile work support tool can cope with such tasks in an efficient manner.
5. **Organisational procedures must be in place:** Mobile workers often work alone, making it difficult to get help when using the system. This makes it very important that the employee using the system is properly trained. The work processes of the workers should also be adapted to the mobile work support tool. This means that there should be procedure rules for when to synchronise, recharge and for how to handle ad hoc tasks.

5.2 Summary

It is clear, based on the system mentioned above, that computer systems made for mobile workgroup only support limited parts of the work of a mobile workgroup participant. Limitations of connectivity, usability pose restrictions on the usage of such computer systems. Users want flexible computer systems that provide the worker with added functionality, compared to what was available without a mobile computerised work support tool. We would claim that the usage of context information is necessary to provide such functionality.

Part II: Our contribution

This part presents application scenarios for a context-aware workflow system and our contribution to the understanding and development of context-aware workflow systems.

6. Application scenarios

The main objective of this chapter is to draw up some realistic scenarios based on the case reviews and literature study. The reason for providing these application scenarios is that we want to give a picture of how our workflow prototypes, which are presented later, are related to actual work processes. The scenarios presented in this chapter are based on existing processes performed in the industry today. The previous chapters of the state-of-the-art part provide background information for our enhancement of these processes. There are several existing scenarios in different environments in the industry (for instance the process industry, the oil and gas industry, the mill industry, high-tech industry, software service industry and other service industry sectors), which are relevant for our work. Travel booking, trouble ticket application, airplane design process and RainMain (a workflow system for the Internet) are some examples of workflow scenarios that have been presented in [26] [39] [40] [42].

In our thesis, it is interesting to look at application scenarios, which emphasise the issues of maintenance and safety. These issues have played an important role for the activities that should be taken into consideration when designing a system. In addition, we want to focus on these issues because they give us the feasibility to implement possible functionality to promote the concepts of workflow, context-awareness and situated planning.

In the following sections we will present two scenarios. The first scenario focuses on maintenance as carried out on an oil production platform. The second scenario explains how safety conditions are maintained in a chemical storage area. In the last section we will discuss how aspects of the presented scenarios can be enhanced and reused in a context-aware workflow system.

6.1 Scenario 1: Maintenance performing on an oil platform

Our first scenario is based on a case review [35] provided by MARINTEK¹⁷. The scenario describes how maintenance work is performed at an oil production platform. By following different steps of the work processes, we will see how a task is triggered, the flow of communication between platform and onshore, until the maintenance work is finally performed on the platform. The generic work processes for corrective maintenance on traditional installations are illustrated in Figure 18 [35]:

¹⁷ MARINTEK, the Norwegian Marine Technology Research Institute, does research and development in the maritime sector for industry and the public sector. Their website can be found at <http://www.marintek.sintef.no>

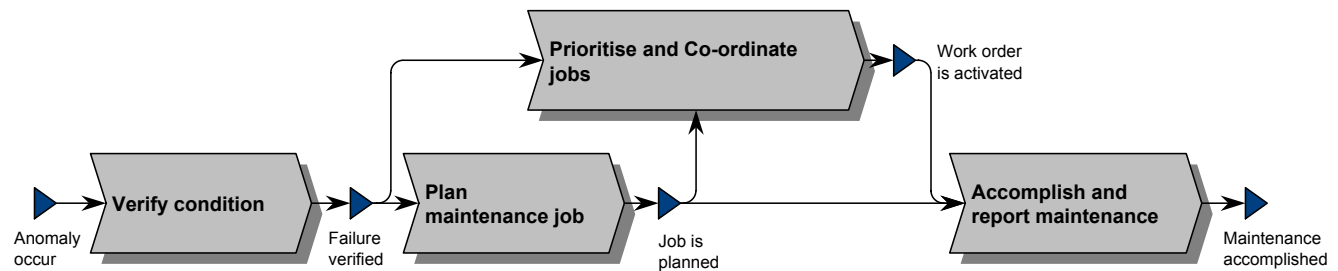


Figure 18: Maintenance related business processes

As can be seen in Figure 18, to perform a maintenance job three processes have been identified; Verify condition, Plan maintenance job and Accomplish & Report. This sequence forms the lower line in the figure. The sequence can be described as a typical situation, which contains one single maintenance job in the “pipeline”. The process Prioritise and Co-ordinate jobs looks at the challenges of handling a high number of maintenance jobs in parallel. A situation with many jobs running in parallel is normal for most complex technical systems.

Further, all processes are decomposed to a workflow process level, identifying roles, activities and sequence/information flow. Four roles have been identified and they can be described as follows:

- Offshore staff: consists of Field operators, Maintenance workers and Organisers.
- Onshore staff: consists of technical support, planners and purchasing.
- Management: is located both onshore and offshore.
 - Onshore management: consists of daily decision makers.
 - Offshore management: consists of Operations Team Leader, Maintenance manager, and Technical coordinator.
- Central Control Room (CCR).

Figure 19 [35] illustrates the first process of the maintenance sequence: Verify condition workflow.

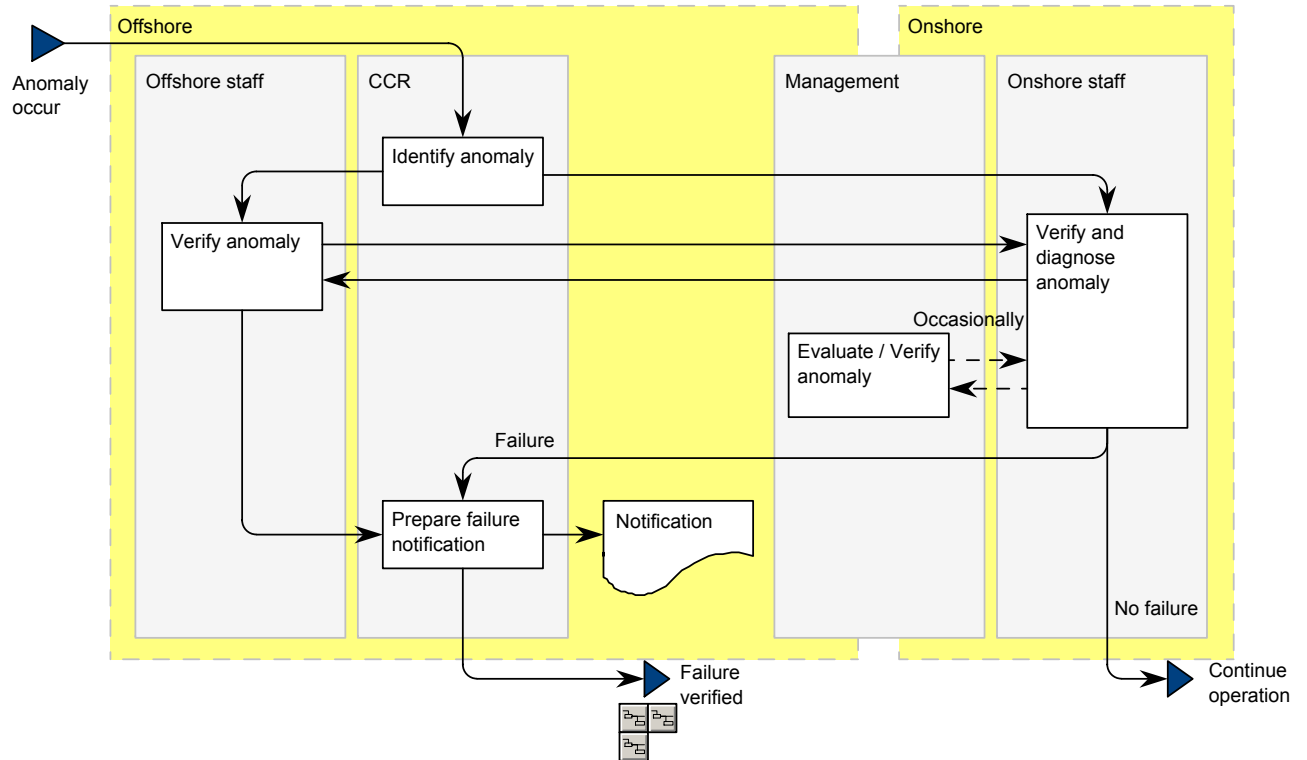


Figure 19: “Verify condition” workflow

A task is triggered after the Central Control Room (CCR) has registered an anomaly from the local environment. When the anomaly has been identified by the CCR, it is sent to either the Onshore or Offshore staff to verify and diagnose the anomaly. When the verification has been accomplished, two situations can occur. If the failure has been verified, the CCR shall respond by preparing the failure notification. In the second situation, where no failure has been found, the operation can continue.

Figure 20 [35] shows the second process of the maintenance sequence; Plan maintenance job workflow process:

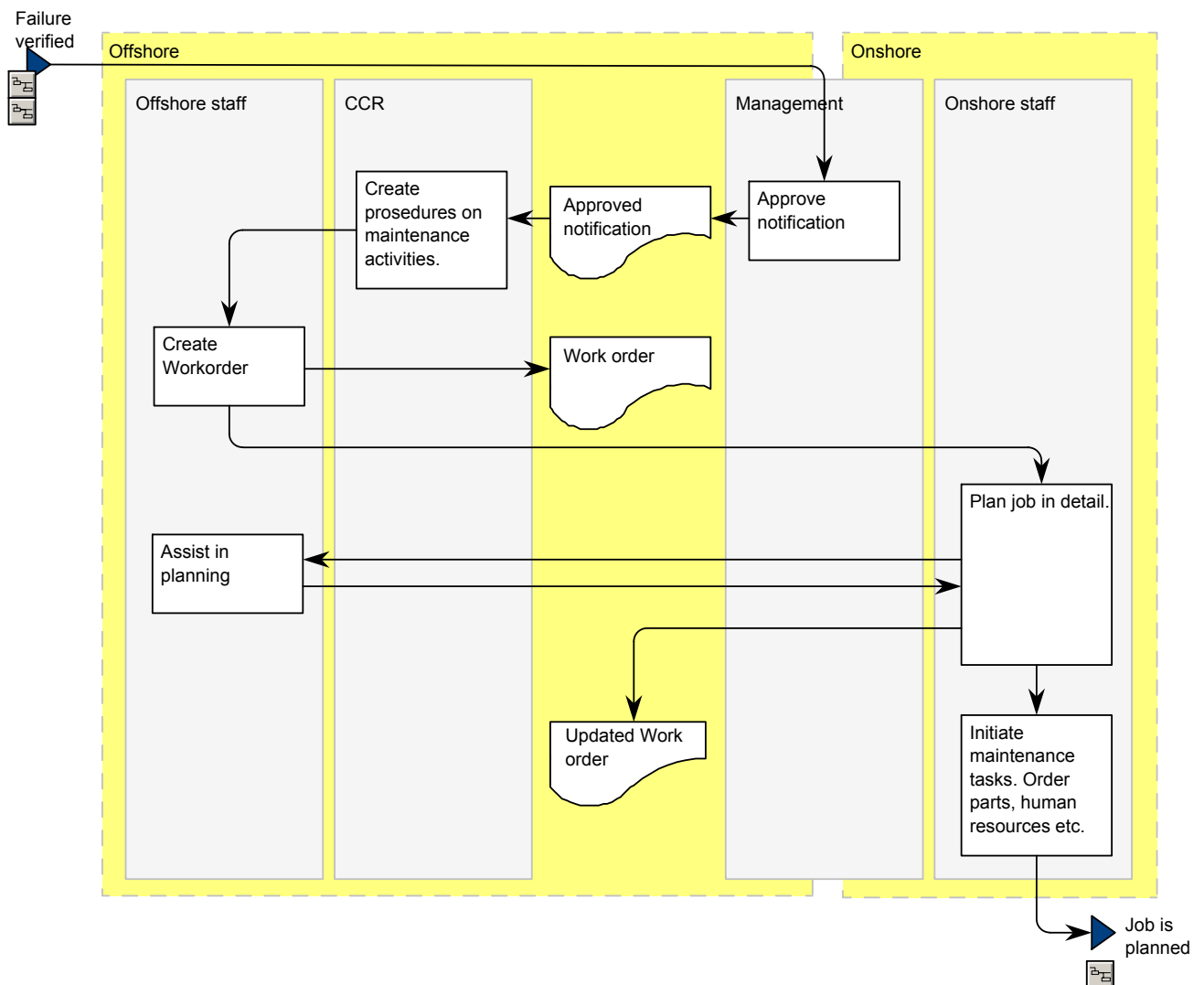


Figure 20: “Plan maintenance job” workflow

The “Failure verified” message is sent to the management staff. The main activity for the management is to approve the notification from the CCR. After the notification has been approved, the CCR will be able to create procedures on the maintenance activities. The information will then be sent to the offshore staff to create a work order. The plan for the execution of a maintenance job is completed when the onshore staff has planned the job in detail. The onshore staff then initiates the maintenance tasks. As seen in Figure 20, most of the planning work is done onshore. The resulting work orders are often incomplete when they arrive offshore, and have to be updated there. The planning process may go several rounds between onshore and offshore before the work order is issued. A lack of communication onshore/offshore and the fact that planners are not present offshore are the most apparent reasons. A solution for this problem is suggested in a later section (see Chapter 6.3.2).

Figure 21 [35] shows the last process in the maintenance sequence; **Accomplish and Report** workflow:

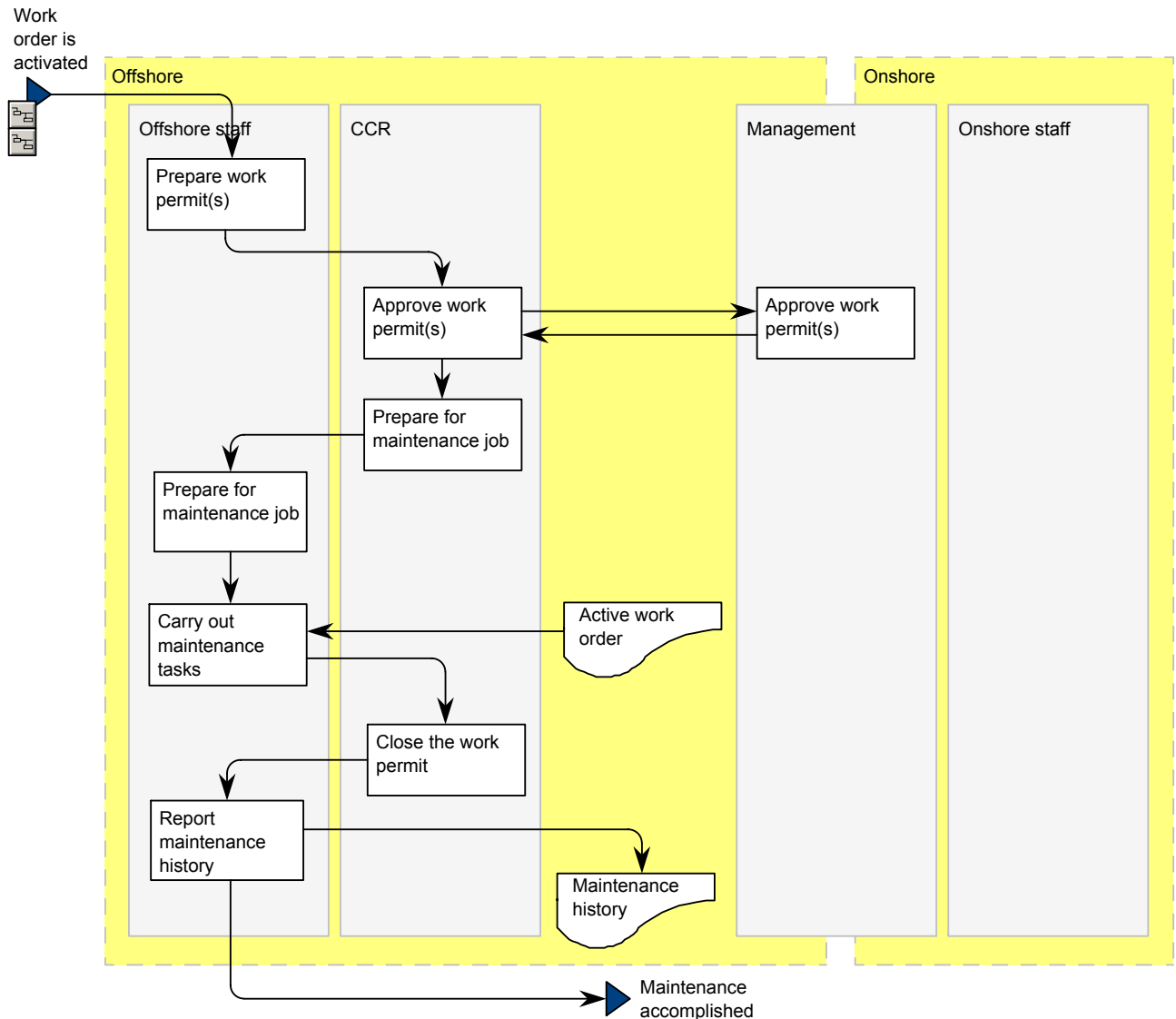


Figure 21: “Accomplish and Report” workflow

As shown in Figure 21, all activities of the actual maintenance job are performed offshore. After a work order is activated, the offshore staff will begin to prepare the work permit(s). The CCR is responsible for approving these work permit(s) in conjunction with the management. The CCR also makes the necessary preparations for the maintenance job. After this has been completed, the offshore staff is able to carry out the maintenance tasks. A maintenance job is accomplished when the CCR closes the work permit. The only task which remains for the offshore staff, is to update the maintenance history for the involved equipment in the maintenance task.

As illustrated in Figure 18, there is an “alternative” path in addition to the main sequence for performing a maintenance job. Normally, there will always be a set of parallel maintenance jobs in the “pipeline”. The main added challenge from this fact is prioritising and co-ordination of the activities to meet overall objectives of safety, production performance and cost effectiveness. The workflow illustration of the Prioritise and Co-ordinate process is shown in Figure 22 [35].

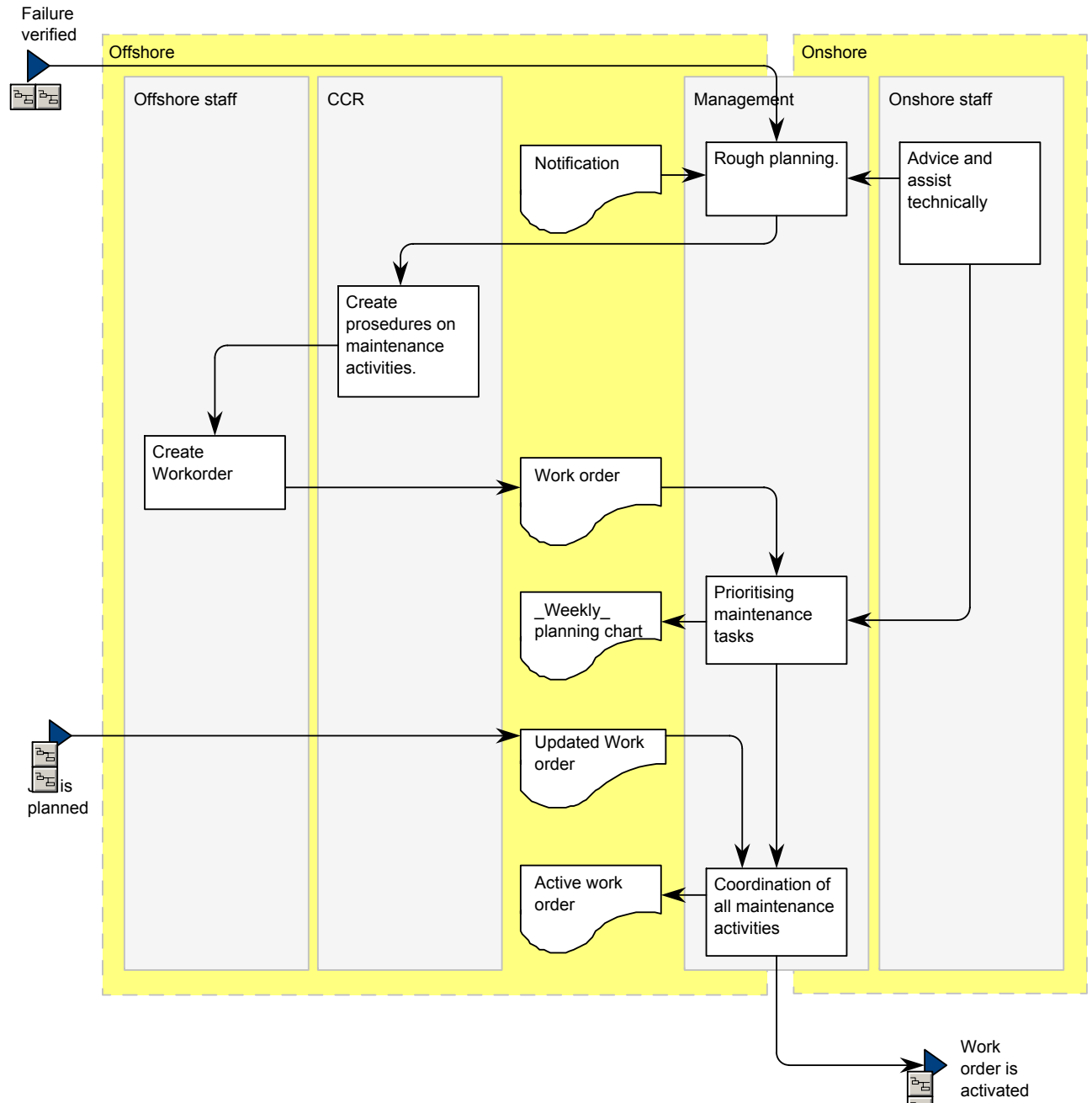


Figure 22: “Prioritise and Co-ordinate” workflow

As illustrated in this workflow process, the main activities here are planning, prioritising tasks and coordination of all maintenance activities. All of these activities belong to the Management. To support these activities, there are in addition other activities like creating procedures on maintenance activities, which are assigned the CCR. The onshore also assist in this process by providing technical advice. The offshore staff’s contribution is the creation of work orders. An important issue when coordinating all maintenance activities is that a work order shall be updated when a parallel job is triggered (in this case a job has been planned and sent into to the workflow process). The work order is finally activated when the Management has prioritised their tasks and coordinated all their maintenance activities.

6.2 Scenario 2: Intelligent chemicals containers

This scenario is based on [52] by Strohbach et al. The issue of safety will be the focus of this scenario. The scenario will give a description of different situations to see how the environment will influence the system in which chemicals containers are able to detect and alert potentially hazardous situations concerning their storage. The situation is critical when two containers with different incompatible contents are stored close to each other.

The motivation for describing this type of scenario, is that it is taken from a concrete application domain, chemical processing. In this application domain, context-aware services are developed against real needs and under consideration of realistic constraints.

The authors of [52] have called the proposed system for **Intelligent Artefacts**. An important aspect here is that the artefacts shall be able to cooperatively assess their situation in the world, without the need for supporting infrastructure in the environment. The **Intelligent Artefacts** concept is also based on embedded domain knowledge, perceptual intelligence, and rule-based inference engines in moveable artefacts. In order to experiment with possible hazardous situations, there has been set up a test bed, which can be described as a scaled-down prototype of a chemical storage facility as it may exist in a chemical processing plant. The test bed is shown in Figure 23 [52] and consists of:

- Intelligent chemicals containers.
- Infrared beacons mounted on cones used for defining approved storage areas.
- A set of software tools for remote monitoring of the inference process and communication of intelligent containers.



Figure 23: Left: physical view of an intelligent chemicals container. Right: Intelligent Container Test Bed

As shown in Figure 23, the approved storage area indicates that chemical containers may be stored in this area for an indefinite time. The unapproved area, in contrast, indicates that chemical containers may temporarily be located in this area but must be moved to an approved area after a certain amount of time.

The test bed is also set up with three containers a1, a2, and b. The two containers a1 and a2 are assumed to contain peroxide, while container b is assumed to be filled with an acid. Acids are incompatible with peroxides. A sequence of container arrangements is shown in Figure 24 [52]:

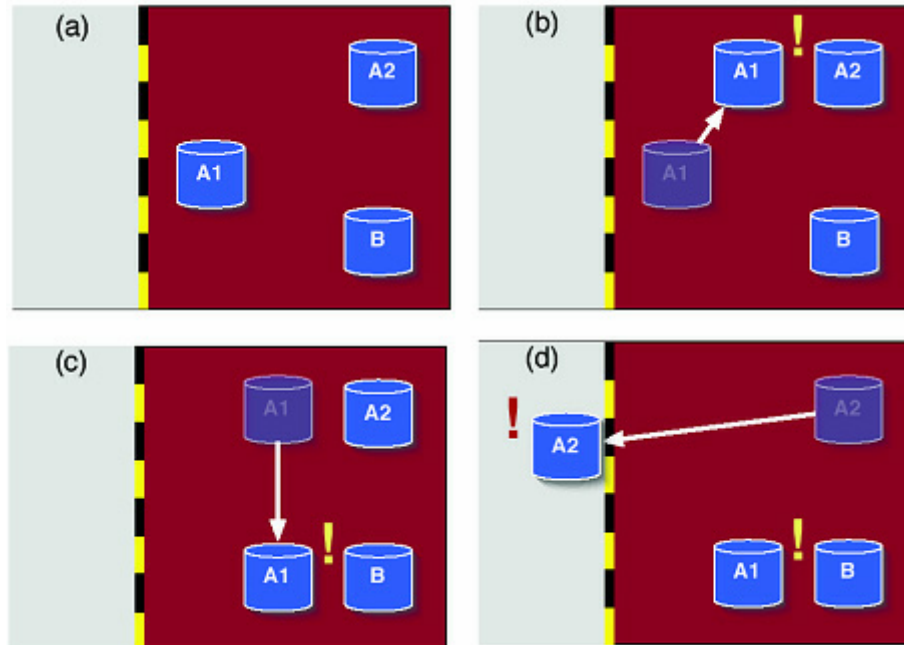


Figure 24: Example arrangement illustrating different hazards: (a) no hazard, (b) critical mass exceeded, (c) reactive chemicals in proximity, and (d) container stored in a disapproved area too long. The exclamation mark indicates which containers are involved in a hazardous condition.

To illustrate possible hazardous situations for the container arrangements, the following scenarios have been identified:

- No Hazard (a)
 - In this situation all containers are stored in an approved area. The containers have been stored there with different time durations. A hazard condition can not be found because all three containers are not placed close enough to each other (which shall be detectable by the ultrasound transceivers when the condition is true).
- Chemical exceeds critical mass (b)
 - In this situation a hazardous situation is detected, when two containers are placed close to each other so that too much of one chemical is stored in one place. During the inference process, both containers wirelessly send queries to each other to determine each others content and mass.
- Reactive chemicals stored next to each other (c)
 - In this situation, a hazardous situation is detected when two containers with two different contents (in this case, the first container contains peroxide and the other contains acid) are stored close to each others.

- Container stored in unapproved area for too long (d)
 - In this situation, a hazardous situation is detected when a container is stored in an unapproved area over a maximum time limit.

- Return to safe situation
 - In this last scenario, all containers are moved back to the original arrangement (Figure 24 (a)). Information about location (where the containers have been stored) and proximity (distance to each others) have also been updated in order to indicate that the containers are again located within an approved area.

The **Intelligent Artefact** approach has emphasised an important aspect in which that information gathering and reasoning can be accomplished in a decentralised way to enable each artefact to determine the state of the world (for instance safety) by itself. Consequently, there is no need for an external database or infrastructure.

6.3 Discussion

In this section, we will discuss how aspects of the presented scenarios can be adapted and reused in a workflow system that incorporates context information. The reason for this is that we want to draw up some generic functionality, which can be implemented in our prototypes. Using this approach, we think we are able to illustrate the main concept behind the proposed workflow system that we will discuss in this section.

Some of the common features that we can draw from the scenarios are that they all are dealing with planning, coordination and enactment of activities, which are being performed in the environment. Situated planning and coordination of activities should also be taken into consideration when designing the overall process. This means that context information from the environment should be integrated with the workflow system. From this point of view, the workflow system will be able to plan according to current contextual conditions, coordinate according to the contextual state of the workflow clients and respond to contextual changes.

6.3.1 Context-awareness

It is clear that the information gathered from the environment plays an important part in the processes described in scenario 1. This means that it is necessary to integrate context information into the workflow enactment service to provide computerised processing of this information. The workflow enactment service must be able to poll or subscribe to context sources to get an overview of the current contextual conditions and access context information gathered in individual activities of the processes.

In scenario 1, the use of context information is illustrated in Figure 19, when the CCR has identified an anomaly that has occurred in the locale environment. This type of triggering of an activity is an example of an ad hoc start of processes and activities.

To support this ad hoc triggering, the workflow enactment service should be able to receive contextual events, for example through a subscription mechanism.

Contextual conditions are not always absolute since variance in sensors and sensor malfunctions may indicate undefined contextual states. It is important that a context-aware workflow system is capable of handling such situations. The handling of undefined context states could involve human workflow participants verifying states or ascertaining the context state through other automated means.

In Figure 19, the activity “Verify anomaly” is performed by the offshore staff. This is done manually by the staff. In this case a tool (for instance handheld computer like PDA¹⁸, wearable equipment etc.) can be used by a maintenance worker to gather information from the environment and send the verified information back to the CCR. Further, augmented artefacts (for instance **smart sensors** [46] or **Intelligent artefacts** [52]) and inference engine and rules in the mobile device, can provide the mobile user assistance in verifying the anomaly. The more information that can be gathered from the environment before a worker from the offshore staff is sent out, the more specific the provided inference rules for the activity can become as to what context information to verify. However, complete automated verification by the CCR may not be possible because of limitations of sensor range and sensor sensing capabilities.

Location is another type of context information that has been illustrated in Figure 21 for the **Accomplish and Report** workflow process. To know what place to go to for carrying out the maintenance tasks, a workflow system should support the workflow participant in getting to the right location for the activity in question.

6.3.2 Situated planning

A workflow system should be built in such a way that it both supports pre-planned and ad hoc/unplanned activities. In Figure 20, the activity “Plan job in detail” is a typical example of a pre planned activity. In this process, the onshore staff is making a plan to carry out the maintenance tasks. This is also a situation that can be compared with the example of the European navigator in Chapter 4.

In the instance of the Turkeke navigator in Chapter 4, the activity “Plan job in detail” can relate to a workflow sub process where all possible sub activities are defined in the workflow enactment service. The activity “Plan job in detail” would then involve specification of how this sub process is to be performed based on current contextual conditions. This specification could consist of inference rules using context information as facts. The workflow client and the workflow participant should then be able to carry out activities by rule based inferring on the current context information and retrieving the related activities from the workflow enactment service.

It is also possible to draw the situated planning scenario further by completely removing the “Plan job in detail” from Figure 20 for scenario 1. This would mean that all planning would have to be performed situated or the activities would have to be

¹⁸ Personal Digital Assistant

specified by the environment. A worker would then arrive at the site where the activity is to be performed. The activity specification would then only specify a goal. By transmitting this goal to the context services in the environment, the context services would have to build in whole or in part a workflow process consisting of workflow activities provided by these context services. This means that we have augmented artefacts containing both workflow activity definitions and rules for how these activities are related to each other based on current contextual conditions.

The first scenario, where we have limited planning of an activity by specifying sub activities and rules for the completion of an activity, is one extreme of situated planning. The second scenario, where all planning and activity specification is done in the environment where the activity is to be performed, makes up the other extreme of situated planning. In between the scenarios, lies the scenario where all the possible sub activities is specified by workflow planners in the workflow enactment service and the rules for sub process building is provided by the augmented artefacts in the environment. Another possible scenario is that the rules for how to build a sub process is provided by the workflow planners, while the environment through augmented artefacts provide the activities for the sub process.

Figure 21 illustrates two sub processes for “Prepare for maintenance job” and “Carry out maintenance tasks”. In this case, we want to provide a solution in which activities are planned in the execution environment to a as high degree as possible. Communication between offshore and onshore about these plans should be decreased to achieve the benefit of cost effectiveness. This means that a sub process called “Plan job in detail” can also be added between these two activities (see Figure 25).

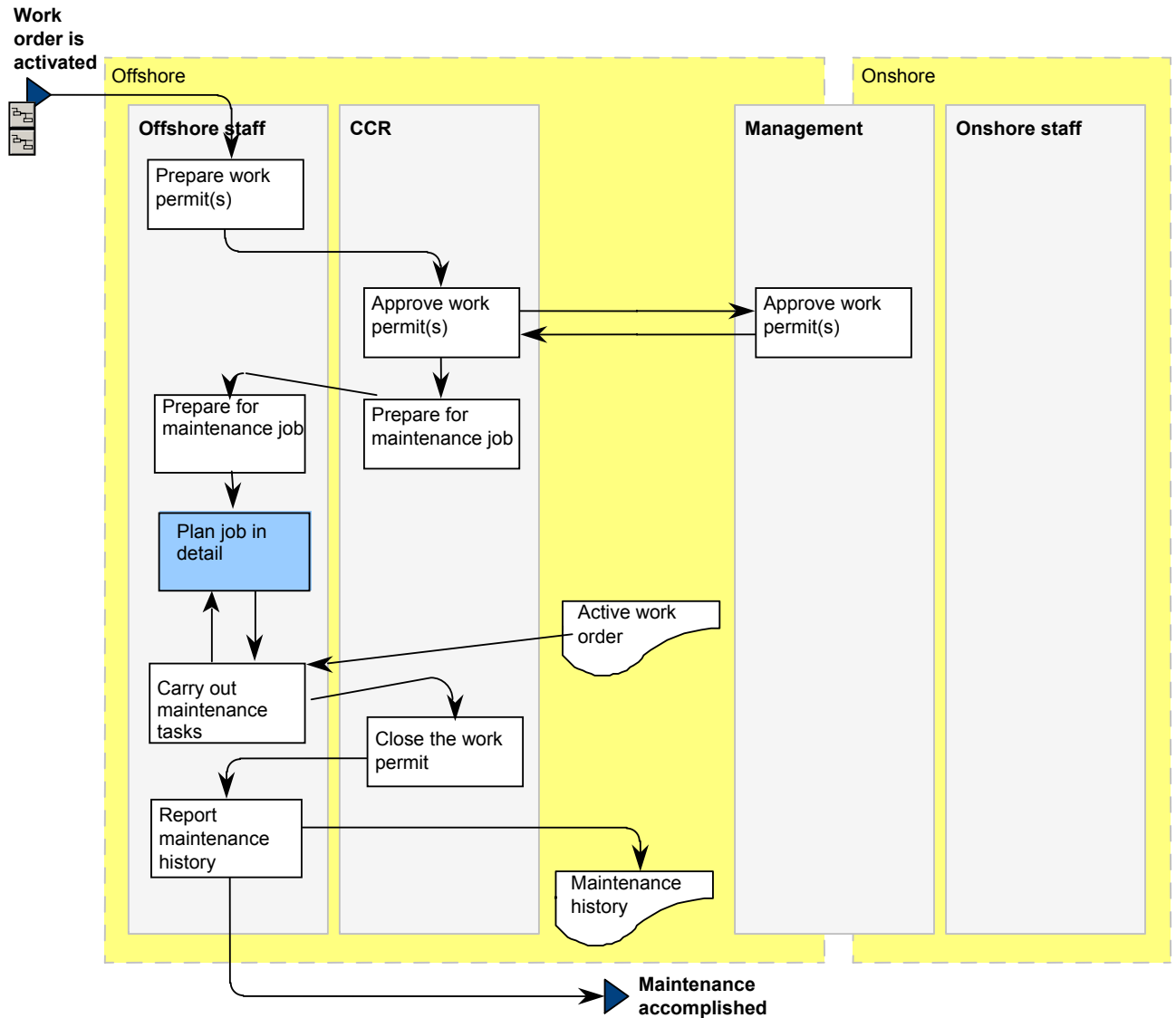


Figure 25: “Accomplish and Report” workflow, revised figure.

As seen in this revised workflow process, the benefit for placing the sub process “Plan job in detail” to the offshore staff is that we will be able to plan activities in situ. The arrow that points back to the process “Plan job in detail” from “Carry out maintenance tasks” indicates that the workflow system should support re-planning, as work is progressing based on changing contextual conditions (for example when unanticipated events occur in the environment). In this situation, the process is being defined in the context of dynamic changes in the environment. The maintenance worker should perform the tasks until a defined post condition is achieved, since the maintenance worker obviously functions as an actuator in the environment altering the contextual state. The workflow system should also provide for learning based on the plan created and new activities defined from the detailed job planning to improve workflow processing at a later stage. All of these issues should be taken into consideration when designing context-aware workflow systems.

6.3.3 Coordination of activities

Coordination of activities is illustrated in Figure 22 for scenario 1. In this scenario, the management is responsible for prioritising tasks and coordination of incoming activities. To achieve this, the management is dependent on supporting processes such as: create procedures, create work order and update work order. When we relate this to a workflow system, we can assume that coordination of activities should be supported to delegate activities to workflow clients. For this coordination, individual workflow participants performing activities become context sources for the workflow system and each other. In this case, a workflow system should be able to coordinate the activities so that maintenance workers do not have to perform activities that interfere with each other. For activities performed in situ, rules specifying process or activity planning should also describe the activity's relations to other activities being performed. The workflow clients will then become context sources for each other, which can be used during situated planning.

6.3.4 Summary

Based on the issues that we have discussed in this section, we will at this point summarise some possible generic functionality that we think should be taken into consideration for the overall design of a context aware workflow application:

- Integration of context information into workflow processes. This could be done using existing technology and standards provided by WfMC.
- Ad hoc activity and process enactment.
- Undefined context state exception handling.
- Situated planning.
- Process and activity coordination.

In addition, it should be considered how a context information infrastructure can be built to support context information retrieval and exchange between a workflow system, participants in the system and the environment. How context information history can be maintained is another important aspect. The sensors in the environment need to be integrated with the context information framework and this integration is another important aspect of a context information infrastructure. We have decided to focus on the actual usage of context information in workflow system, but we will discuss these aspects briefly.

Based on our discussion in this chapter, we will specify requirements and design considerations for our prototypes.

7. Requirements for workflow systems integrating context information

In this and the following chapter, our contribution in the development of context-aware workflow systems will be presented. We start by presenting the requirements for such a system. The following chapter will present prototypes of workflow systems, which focus on different aspects of the dynamic nature of a context-aware workflow system.

This chapter specifies both the functional and non-functional requirements for a context-aware workflow system. The presented requirements are related to the previously presented WfMC standards and the application scenarios discussed. By using the standards and the scenarios as a starting point, it is possible to specify some requirements for a general workflow system supporting context-aware processing of processes and activities.

7.1 Basic workflow system requirements

It is not possible to describe the requirements of a context-aware workflow system without first specifying the requirements of a general workflow system. We will attempt to follow the workflow standards as presented by the WfMC as far as possible. This includes the Workflow Reference Model [62] and the specified interfaces. Our prototypes should only extend the WfMC specifications, when absolutely necessary. The most important of the interface definitions for us is the WfMC's Interface 1 specification [63], which specifies the interface between the workflow enactment service and the process definition. This specification will be used as the basis for our workflow enactment engine. The other interface specifications are not directly relevant for our work, and we will only mention these briefly and when our solutions are incompatible with these specifications.

As we attempt to follow the WfMC's Interface 1 specification [63], the foremost requirement is that the workflow enactment service is able to interpret a workflow process specified in XPD and execute according to this definition. The execution of a process definition entails sending and receiving activities, evaluation of transitions and updating workflow relevant data. The enactment service may have several concurrently running processes with different process definitions. The enactment service must be able to communicate with workflow clients over a network to send and receive workflow activities. This constitutes our requirements of a basic workflow system. Figure 26 illustrates the actors and use cases in such a system. The actors are the workflow enactment service, the workflow client application and the

workflow participant. These three actors perform the tasks in a basic workflow system.

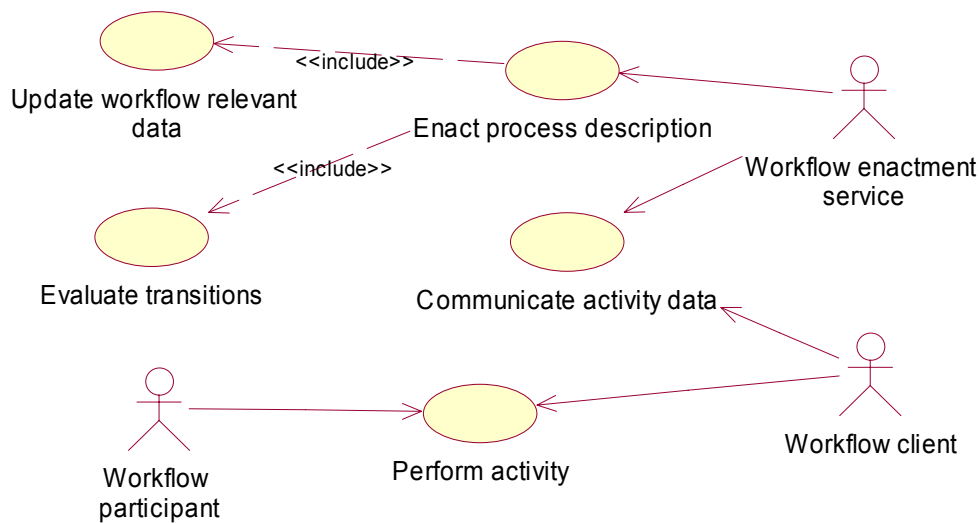


Figure 26: Use case diagram for a basic workflow system

As was concluded in Chapter 2.4, we have decided to represent our workflow processes using both the state chart modelling technique and the activity diagram model.

7.2 Context information representation and retrieval

The underlying context support system is an important part of any context-aware system. A general context-aware system poses several requirements to such a system. We have two key requirements related to the context support system. The first requirement is to remove low-level input handling from the context-aware system. The second requirement is for context interpretation and aggregation to be provided by the context support system.

The requirement for removal of low level input handling from the main application code is necessary to provide for reuse of existing context handling code. This is important in a workflow system where different workflow processes and workflow clients may use the same context sensors.

A level of context interpretation and aggregation provided by a context-support system is necessary to allow for reuse of context interpretations. The higher the level of context interpretation and aggregation becomes, the more difficult it becomes to present users with the reasons behind the actions taken by the system on behalf of the users [8]. Nevertheless, for this report, the need for reuse of context interpretations will be considered superior to the need for showing users the “reasoning” behind the actions performed based on context information. This has to do with the fact that most aspects of the execution of workflow processes are performed autonomously by the

workflow engine. In addition, workflow processes are highly planned sequences of activities, so the addition of context information in the transitions of a workflow must be planned by the workflow designer to obtain the desired effect. Workflow transitions can already be based on context information, so the goal must be to enable workflow systems to handle the dynamic contextual environment that exists in the mobile world.

Dey et al. have presented a context information framework in [17], which contains several abstractions for context information collection and transformation. A summary of this article was presented in [41]. A **widget** abstraction communicates directly with context sources, such as sensors, and provides the lowest level of abstraction. We chose to use this abstraction for our context sources. A **widget** transforms low-level input from sensors into the context information representation used by the system. Since we achieve the effect of having underlying context sources by using this abstraction for our overall system, there is no need for actual sensors in our prototypes. This means that the **widgets** provide our context information. The context information framework in [17] also uses a **discoverer** service that satisfies the requirement for locating distributed context sources. The **discoverer** service works by allowing context sources to register with attributes describing the context information they provide. The context-aware system can then query this **discoverer** service to get a handle to the context source. This service is useful when trying to locate context sources, so our prototypes also include this service.

Context sources should provide two means of context information retrieval. This means that both context source polling and subscription to context sources should be supported. The polling approach gives us the ability to retrieve context information when it is needed, but we are not notified of changes to the context information. The publish/subscribe paradigm used when subscribing to context sources does not provide context information at once, but listeners receive context events when such events occurs. It is also necessary to allow subscribers to specify conditions for when they should receive contextual events, at least when workflow clients are receivers of these events. Workflow clients running on limited mobile devices may easily be overloaded under such conditions. A workflow system should also specify conditions for receiving these events on some level, to avoid unnecessary processing.

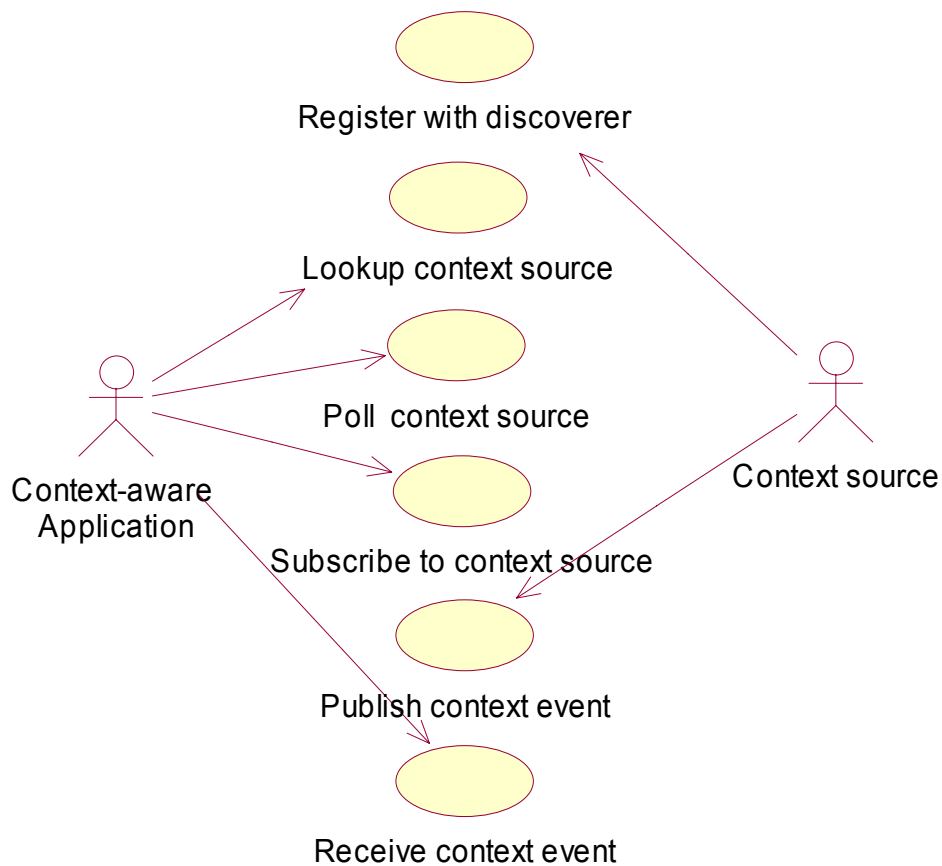


Figure 27: Use case diagram for context framework and context-aware application

Figure 27 shows the actors and use cases involved in usage of the context information framework. The context source provides the context information. The context-aware application uses the services of the other actors. The discoverer is not shown in the illustration, since it does not perform actions. It simply provides access to context sources.

7.3 Context-aware functionality in a workflow system

The goal of this report is to integrate and make use of context-information in our workflow prototypes. This includes the acquisition and usage in both the workflow enactment service and the workflow client. This means that the previously given requirements for a workflow system needs to be extended with requirements for the context-aware functionality needed.

7.3.1 Workflow enactment service context-awareness

We will first look at context-aware functionality in the workflow enactment service. The first requirement for context-awareness in a workflow enactment service is the

ability to use context information in the processing of workflow transitions. This is possible using the current standards. However, the modelling of context source lookup, polling/subscription and conversion of the context information between the context information representation and the workflow representation, must be as easy as possible.

Responsiveness to contextual changes is another important requirement for a context-aware workflow system. For the workflow enactment service part, this means that the enactment service must be able to handle several different types of events. Ad hoc start of processes and activities are an important requirement related to responsiveness to contextual changes. This means that context events can be considered as pre-conditions for process and activity start. Since activities may influence the environment and the environment changes over time, context events may also function as terminators to process enactment. This means that context events can be considered as post conditions to processes. Context events may also invalidate previous satisfied conditions for the current process path. In such an invariant scenario, it is necessary for the enactment engine to revalidate the current process path or choose another. A workflow enactment service must therefore be able to set up subscription and receive context events. These events must then be mapped to the correct process and the correct response must be executed.

Contextual changes may also open new process paths, which earlier in the process enactment had been discarded. A workflow enactment service should therefore be capable of finding other ways to finish a process if the current process path does not lead to the process goal with the current contextual situation.

The contextual environment is not always in defined states. It is therefore sometimes impossible for the underlying context sources to decide the concrete state the environment is in. This situation may require the workflow system to decide the contextual state with more elaborate means or the situation may require human intervention. A workflow enactment service must therefore be able to handle this exception situation and provide solutions for the management of such situations.

Activity theory [56] and situated planning [5] also lay out important requirements for our prototypes. It is necessary to allow for both pre-planned and unplanned workflow process enactment. Pre-planned workflow enactment is already provided for in the WfMC specifications. Unplanned workflow represents more of a challenge since this is not yet part of any specification. The building of workflow processes based on current contextual state is therefore necessary functionality of a workflow enactment service. This functionality can be based on rules provided for context state evaluation and linking the current state(s) with one or more activities. The other possibility is that augmented artefacts are capable of returning activities to be performed based on input of the overall goal from the workflow enactment service and communication between the relevant augmented artefacts to ascertain the current contextual state. The environment itself may also provide goals instead of the workflow client based on its own perception of the current state. The new process should become part of the overall process definition to allow the context-aware workflow system to work as a “technology of accountability” [5].

Figure 28 illustrates the previously mentioned requirements for context-awareness in a workflow enactment service.

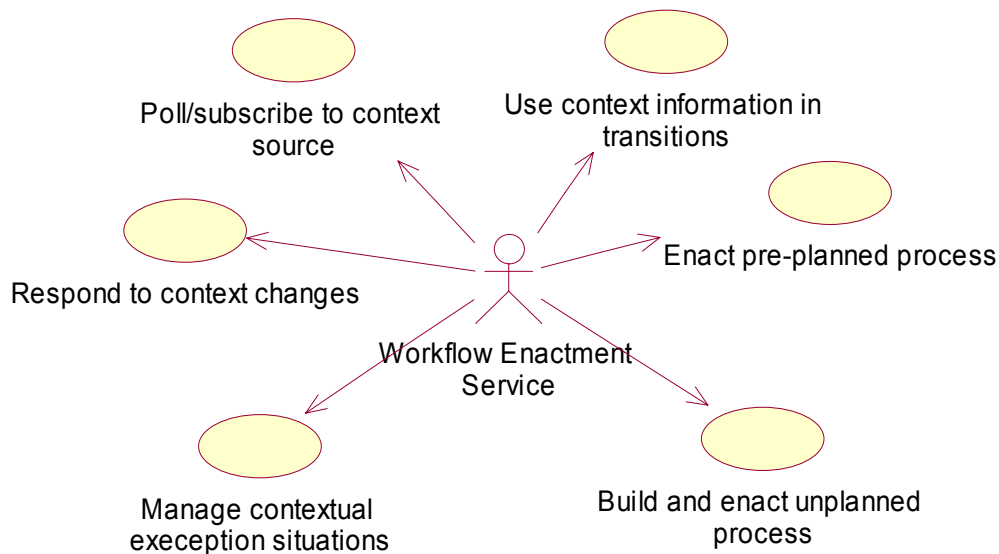


Figure 28: Use case diagram for workflow enactment service context-awareness

7.3.2 Workflow client based context awareness

The workflow clients have a relatively simple function according to the WfMC Interface 1 [63] specification. They receive activity definitions with related data and update the data as necessary and then return the activity to the workflow enactment service.

The relation between workflow systems and Activity theory [56] and situated planning [5] mean that we have to enhance the functionality of the workflow client. When a workflow client receives an activity, which specifies a job to be done at a remote location, activity theory specifies that this action is best done in situ. A workflow client should therefore benefit from being able to enact a process definition by itself. Of course, the workflow enactment service should be capable of processing this for the workflow client, but it is possible to envision a situation in a mobile scenario where the necessary communication is not possible or undesirable. All possible contextual conditions at the site, where the activity is to be performed, cannot be known in advance by a workflow planner. Several rules, specifying how the workflow client should interpret the contextual situation, must be provided along with the activity definition. All activities, which are to be performed in situ to solve the overall activity, can either be defined in advance if they are known or by the environment itself. A workflow client must be able to carry out process enactment, poll or subscribe to context sources and interpret the contextual state based on rules. A process path based on activities received from augmented artefacts in the environment or the local or central workflow enactment service must also be built.

Activity coordination is another important aspect of workflow process enactment. It is possible to have several concurrent activities running in a workflow system. Sometimes these processes directly affect each other. In existing workflow systems, the coordination is handled by the workflow enactment service. In the situated planning scenario, this is not as easy if one want to achieve the highest possible efficiency. Individual situated activities performed by one workflow participant as a result of situated planning may function as pre-conditions for other situated actions performed by another workflow participant. In this scenario, the workflow clients of the workflow participant would work as context sources for each other. The rules for each activity, to be performed in situ, must therefore specify the relation to other activities, also performed in situ. The polling or subscription to another workflow client would not be any different from the subscription to any other context source. The current state of the local enactment service in the workflow client works as the context source.

Contextual post-conditions for activities are best handled by workflow clients. This applies to both normal activities and situated activities. A workflow client should therefore handle such conditions.

Figure 29 illustrates the use cases for the workflow client and the participant using the client based on the previously mentioned requirements for client based context-awareness.

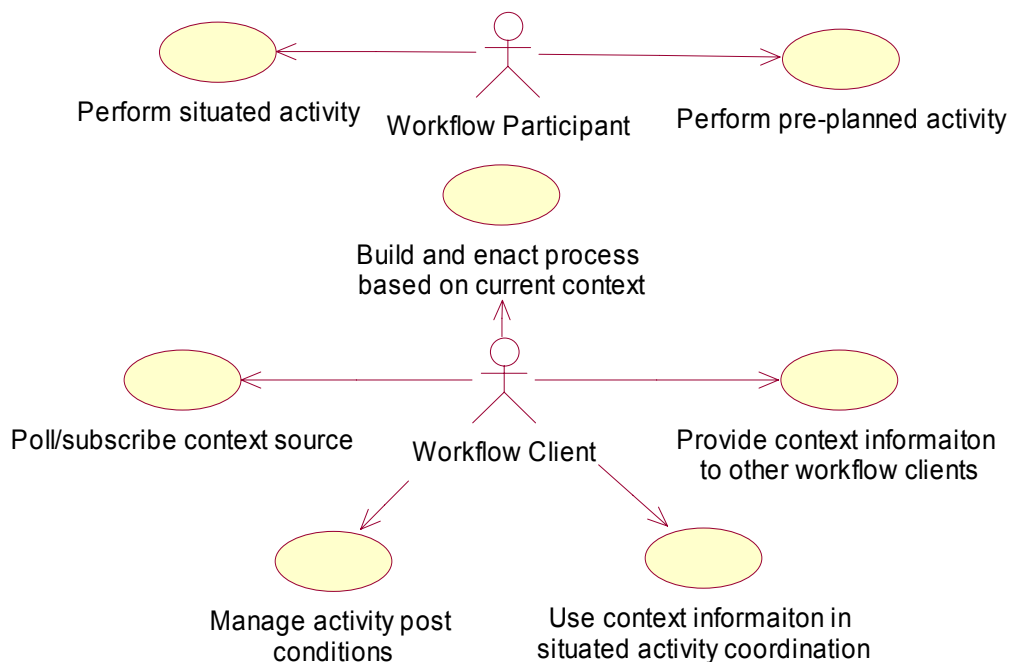


Figure 29: Use case diagram for workflow client based context-awareness

7.4 Summary of functional requirements

The previous sections presented several functional requirements for a context-aware workflow system and its support components. We will summarise these requirements in this section.

The following list summarises the requirements for a basic workflow system:

- Adhere to the WfMC standards and the Interface 1 specification in particular.
- Interpret and enact process definition specified in the XPDL language.
- Send and receive activities.
- Evaluate transitions.
- Update workflow relevant data, based on completed activities from workflow participants, which has updated such data.
- Perform concurrent enactment of processes.
- Communicate with workflow clients to send and receive activities.

To summarise the requirements for context information representation, interpretation and retrieval, the following list is provided:

- **Widget** abstraction for context sources to remove low-level input handling from the main application code (separation of concerns) and to provide limited context interpretation.
- **Discoverer** service for context source lookup.
- Support both polling and publish/subscribe mechanisms for context information retrieval from context sources.

To summarise the functional requirements related to the use of context information in workflow systems, the following list is provided:

For a workflow enactment service:

- Context information used in the evaluation of workflow transitions.
- Easy process definition of context source lookup, polling and conversion of the context information between the context representation and the workflow representation.
- Responsiveness to contextual changes or events:
 - Ad hoc start of processes and activities.
 - Context as post conditions for processes.
 - Correct handling of invariant conditions.
- Support revalidation of selected process paths, if the current path does not lead to the process goal.
- Exception handling of undefined contextual states.
- Support both pre-planned and unplanned process enactment, by providing rule-based process building with learning.

Workflow client based context-awareness:

- Perform situated planning based on current contextual conditions.
- Perform situated activity coordination between workflow participants.
- Context as post-condition for activities.

7.5 Mobility requirements

The requirement for mobility is of lesser concern for our prototypes. We will however mention these here, since they form a basis for a discussion of our prototypes. The background for these requirements was presented in [41]. Generally, a workflow system supporting mobility should try to fulfil the following requirements. These requirements are related to both the underlying technology as well as the workflow system itself:

- Support for physical mobility and network mobility.
- Support for unreliable communications.
- Support for disconnected operations and asynchronous communications.
- Activity locking with support for reassignment of activities.
- Flexible task assignment support.
- Support for session mobility.
- User able to select the data elements to be transferred to the mobile device.
- Device independence.

7.6 Non-functional requirements and design considerations

We base our work on vertical prototype development and the prototypes provide proof-of-concepts. The stakeholders in a context-aware workflow system are the workflow participants, the workflow planners, system developers and support personnel for system operation and management. However, we are not considering a fully implemented context-aware workflow system. The main stakeholders are therefore reduced to us as prototype developers and possibly other developers, who want to extend our prototypes. This means that in practice all of our non-functional requirements to our prototypes are related to us as developers.

Each prototype builds on the earlier created prototypes. This makes the non-functional requirements for modifiability and reusability essential for our system design. This means that our system design should promote:

- Separate components.
- Loose coupling between components.
- Separation of concerns, by separating functionality between components.

We will follow these requirements in the architecture and design of all our prototypes.

8. Prototypes

In this chapter we will present the prototypes, which represent our solutions to the stated requirements. Each prototype focuses on different sets of requirements and all requirements will unfortunately not be covered by our prototypes. The prototypes are developed iteratively, so each prototype extends the functionality of the previous prototypes. The prototypes are presented in the order they were created. The source code, the Java documentation and the prototype programs can also be found in the attached CD-ROM. A simplified test report for the prototypes is provided in Appendix D.

8.1 Design overview

It is first necessary to know how we will represent context information in our prototypes. Chapter 3.3 lists several context information representation and modelling techniques. We have chosen to go with an event representation of context information. This stems from the fact that we wanted to represent context information in the simplest way possible. The information itself should not be complex either, so a context source generating Boolean values is ideal. The other reason why an event representation of context information satisfied our needs is that the workflow system needs to be responsive to events coming from an external source. When we considered both of these requirements, the choice of representation became easy. Further, events can be represented in several ways. Since our context information is not very complex, a simple data structure containing the context information is sufficient.

The non-functional requirements for separate components and separation of concerns lead us to separate the prototypes into several components at the outset. We created five components, which run as separate processes. These are:

- **BooleanWidget** – Process which provides Boolean context values.
- **Discoverer** – The discoverer service, which provides service discovery for context sources.
- **CII**¹⁹ – The link between the workflow enactment service and the context framework components.
- **WorkflowClient** – A workflow client, representing a human workflow participant.
- **WorkflowEnactmentService** – The workflow enactment service, which carries out enactment of workflow process descriptions.

¹⁹ CII: Context Information Integrator component.

The processes are connected using Java RMI²⁰ [88] over a TCP/IP²¹ [76][77] network. The choice of Java RMI was based on the fact that we wanted an easily available system for distributed computing to illustrate how context information can be integrated into a workflow enactment service. The main drawback of Java RMI is the fact that it is based on synchronous communication, and as such is not well suited for mobile systems. We chose to disregard this drawback at this stage, since it is not critical to illustrate the main point behind the prototypes.

The following sections specify a more detailed design for each prototype.

8.2 Context information used in workflow transitions

This prototype is made in response to the following functional requirements:

- Adhere to the WfMC standards and the Interface 1 specification in particular.
- Interpret and enact process definition specified in the XPD L language.
- Send and receive activities.
- Evaluate transitions.
- Update workflow relevant data, based on completed activities from workflow participants, which has updated such data.
- Perform concurrent enactment of processes.
- Communicate with workflow clients to send and receive activities.
- Context information used in the processing of workflow process transitions.
- Easy process definition of context source lookup, polling and subscription and conversion of the context information between the context representation and the workflow representation.

Context information can be used in the evaluation of workflow transitions. Process paths in a workflow processes can be dependent on certain specific conditions being valid, before the workflow system commences enactment of the process path. An example of such a condition is to check customer credit before an order is fulfilled. Another example is that a certain system has to be off-line before maintenance is initiated.

To illustrate how context information can be used in the evaluation of workflow transitions, a simple prototype has been constructed. This prototype uses the WfMC's Interface 1 specification [63] to build a workflow enactment service. As specified in this specification, an XPD L document is used to create the workflow processes. The workflow enactment service reads this document as part of its initialisation procedure. The parsing of the XML is based on the Xerces parser [91]. The XPD L document used in this prototype specifies a process with three activities and two transitions. Activity 2 and 3 each has one specified performer. Activity 1 is the actual query of the context source, which forms the basis for the transition that follows. In this example, a context source providing Boolean values is used. Based on the returned Boolean value, the workflow enactment service evaluates which transition to use. The

²⁰ RMI: Remote Method Invocation

²¹ TCP/IP: Transmission Control Protocol/Internet Protocol

workflow enactment service sends the correct activity to the specified performer based on the context information provided. An activity diagram illustration is provided in Figure 30. The complete XPDL specification can be found in Appendix E.

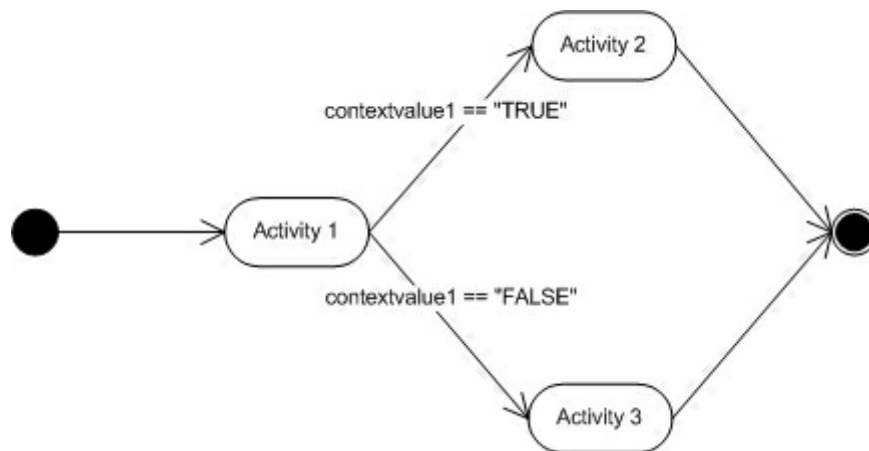


Figure 30: Workflow process illustration

8.2.1 Prototype packages

The prototype consists of eleven packages. These packages are illustrated in Figure 31. The dependencies between the packages illustrate which packages use the facilities of the other packages. We have collected several of the interfaces used in communication between the different processes in the “common” package. This was done to make the processes separate from each other by allowing each process to have everything it needs without needing to have the complete implementation of each class. By using this approach, one can create separate program packages. Each program package would only contain its own package in addition to the “common” package. Common interfaces used between context related processes such as the Discoverer, CII and BooleanWidget are available in the “contextsource” package. These interfaces are used for context information query and subscription.

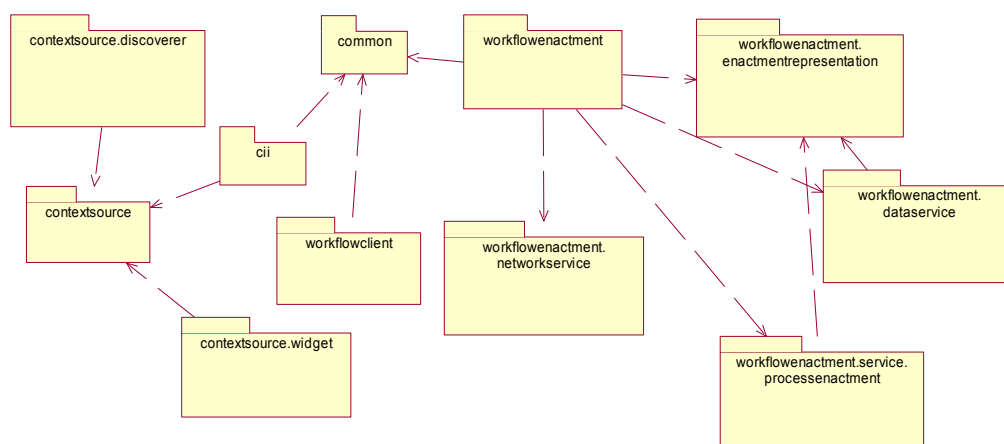


Figure 31: Component view of the complete prototype

The following packages are part of the workflow enactment service process:

- `WorkflowEnactment` – contains the functionality to initiate the other components of the workflow enactment service. It is also responsible for control of the running workflow processes.
- `WorkflowEnactment.NetworkService` – contains functionality which makes the services of the workflow enactment service available in network mode.
- `WorkflowEnactment.EnactmentRepresentation` – contains the classes which represent the specification of a workflow process.
- `WorkflowEnactment.DataService` – contains functionality which maps the XPDL process specifications into the internal process representation.
- `WorkflowEnactment.ProcessEnactment` – contains the functionality to enact a workflow process.

The other packages are named according to the processes they belong to.

8.2.2 Workflow enactment service

The workflow enactment service does two main tasks. At start up, it reads a specified file, which contains the process definition. This file is important since it contains the definition of how the workflow enactment service is supposed to execute the workflow process. A description of the process used in this prototype was given in the introduction to this chapter. The complete process definition in XPDL is provided in Appendix E. The workflow enactment service acquires the process definition through the “DataService” interface, which is part of the `WorkflowEnactment.DataService` package. This separates the workflow enactment classes from the data acquisition classes, and thereby satisfies the need for separation of concerns. We are in essence creating two separate layers, a data service layer and a workflow enactment layer. The main class in the data layer is the `BasicDOM` class. It first builds a W3C DOM [95] tree utilising the Xerces DOM parser [91]. This tree is then traversed to build a hierarchy of objects used in the workflow process enactment.

A class diagram, illustrating some of the classes involved in representing the process definition and executing this definition, is provided in Figure 32. The “BasicDom” class is responsible for reading the XPDL based process definition and traversing it. It creates one or more instances of the “WorkflowProcess” class, which is located in the `WorkflowEnactment.EnactmentRepresentation` package. This is the main class representing a workflow process. The “Activity”, “Participant”, “Transition” and “Condition” are some of the other classes involved in representing a workflow process. These classes inherit several utility methods from the abstract “EnactmentRepresentation” class. These classes are located in the `WorkflowEnactment.EnactmentRepresentation` package.

The “WorkflowProcessExecuter”, which is located in the `WorkflowEnactment.ProcessEnactment` package, is responsible for carrying out the actual enactment of a workflow process. The implementation of this class follows the principles behind a state machine. It runs as a separate thread in the workflow

enactment service. The “WorkflowEnactmentService” class, which is located in the WorkflowEnactment package, works as a controller class for the entire workflow enactment service. This class starts the enactment of new processes and functions as gateway to each “WorkflowProcessExecuter” instance, which wants to send and receive activities.

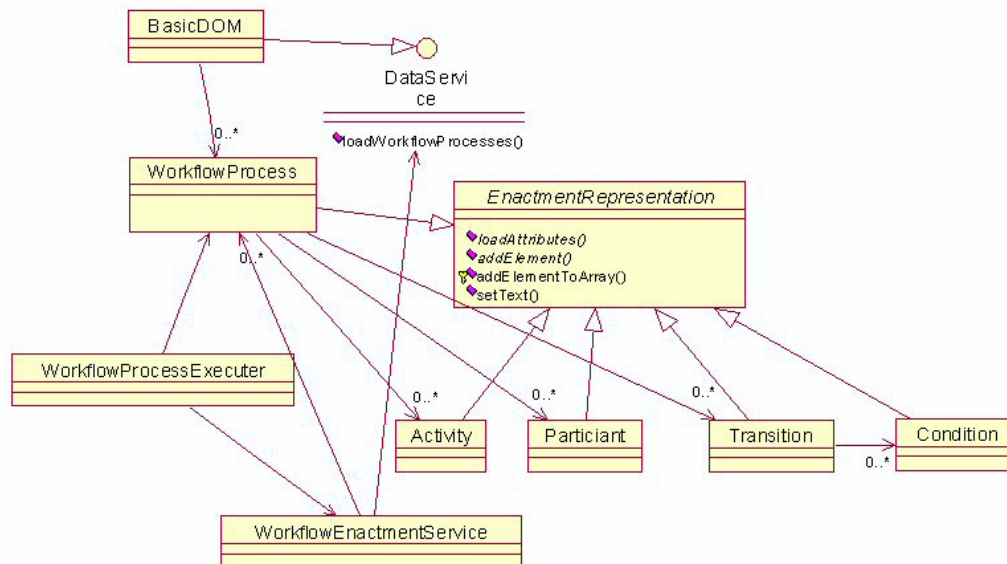


Figure 32: Class diagram of the most important classes in the workflow enactment service

This class diagram forms only a limited view of the entire prototype. Class diagrams for all packages are presented in Appendix F.

8.2.3 Inter-process communication

This section will illustrate how the separate processes in this prototype communicate. To support this illustration, several sequence diagrams showing method calls and returned information over RMI [88] are presented.

8.2.3.1 Context framework implementation

The context information used in this prototype is available through a **widget**. In addition a **discoverer** has been implemented to facilitate service discovery. This implementation is a partial implementation of the context framework specified in [17] by Dey et al. Both the **discoverer** and the **widget** have been implemented in Java RMI [88]. The **widget** registers itself at the **discoverer** by providing its remote interface to the **discoverer**. It also specifies its key attribute to the **discoverer**. This attribute is used to perform a lookup of the **widget**. A process, which wants to get a reference to a context source, would then have to query the **discoverer** to get a reference to the remote object of the context source it wants to use. A context source is accessed through its remote interface, the “ContextSource” interface, which is a generic interface for all context sources. This interface provides possibilities to poll

the context source or to set up a subscription to the context source. The subscription follows the publish/subscribe paradigm [78].

8.2.3.2 Workflow client integration

The registration of workflow clients and the CII with the workflow enactment service follows the principles behind the Factory pattern [22]. Each connected client has a reference to its own “WorkflowClientConnection” object located within the workflow enactment service. This is strictly not necessary for this prototype, but can be used to identify each connection in later prototypes. In addition it can also be used to handle correct deregistering of each client when network failures occur through use of the Unreferenced interface. The “WorkflowClientConnection” interface also servers to adapt certain methods of the “WorkflowEnactmentService” class to network mode according to the Adapter pattern [22]. However, the main mission of the “WorkflowClientConnection” is to provide call-back functionality to the clients. By having the possibility to initiate calls from the client to the workflow enactment service, we give workflow clients the ability to return activities upon completion. The “WorkflowEnactmentService” class maintains a Vector of all registered clients. The process of registering a workflow client is illustrated in the sequence diagram in Figure 33.

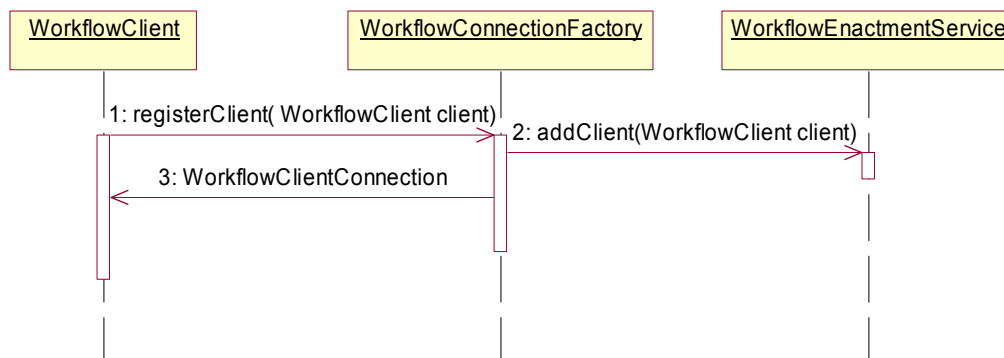


Figure 33: Workflow client registration

The sequence diagram for the delegation of activities from the workflow enactment service to the workflow clients is a limited version of the context source polling diagram provided later. The reason for this is that both the CII and the workflow client implements the same WorkflowClient interface. The main task for the client is to receive an activity from the workflow enactment service and process this activity by interaction with the users and/or other applications. In terms of this prototype, this functionality has not been implemented for the workflow clients. Instead the workflow clients simply return the workflow activity back to the workflow enactment service. Prior to receiving activities, the workflow clients have to register with the main workflow enactment service. Each client has a unique id, which serves to uniquely identify the client when activity assignment takes place. The id of the performer to execute each activity is specified in the workflow process definition. To illustrate the process of workflow client activity processing, the sequence diagram in Figure 34 is provided. It is also worthy of note that the workflow enactment service

accesses the clients through a NetworkService interface. This interface serves to separate the workflow enactment service from RMI implementation details. This separates in practice, the workflow enactment service in a new layer, the network layer. In total, the workflow enactment service is divided into a data service, workflow enactment and network layer. This has been done to fulfil the requirement of separation of concerns.

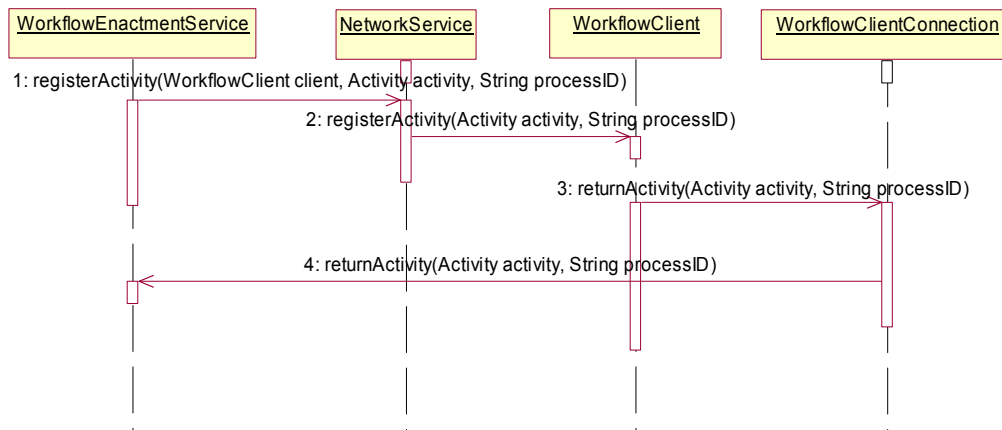


Figure 34: Workflow client activity processing

8.2.3.3 Context information integration

A context information integration component was discussed in [41] as a link between the context framework specified in [17] and a workflow enactment service. The main reason to have such a component is easier process definition. Particularly, this component addresses the requirement for easy process definition of context source lookup, polling and conversion of the context information between the context information representation to the workflow representation. Since this component handles all interaction with the context framework, the process of looking up, polling and subscribing to a context source can be modelled as a single activity. In the case of subscriptions to context sources, this component can be set up to initiate actions in the workflow enactment service based on context changes. This is important, since it allows the workflow enactment to be responsive to changes in the environment. In [41], this component was called **Context Information Integrator (CII)**. In this prototype only the context source polling aspect of the CII is explored. The CII appears as just another client to the workflow enactment service. The activity received by the CII specifies which context source to use. The CII then contacts the discoverer to find the correct context source. Upon receiving a reference to the context source to use, the CII polls the context source and returns the result to the workflow enactment service. Figure 35 illustrates the interaction between the workflow enactment service, CII, discoverer and context source. The workflow enactment service is represented by the “WorkflowEnactmentService” class, the “NetworkService” interface and the “WorkflowClientConnection” interface, which is a call-back interface for workflow clients. The CII component is represented by the “WorkflowClient” interface, which it implements, and the CII main class. All interaction between separate processes is over RMI [88], same as the rest of the prototype.

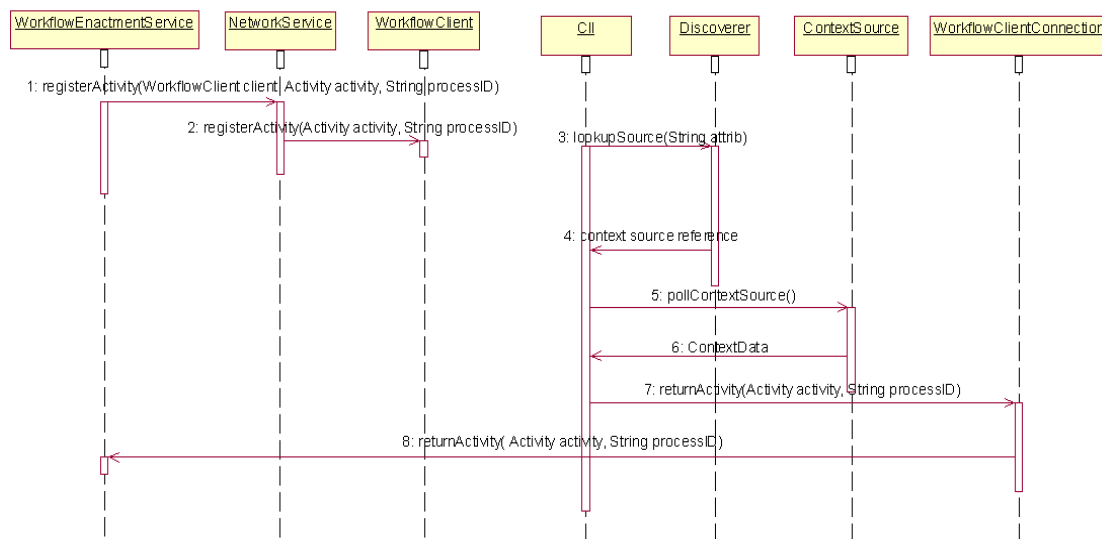


Figure 35: Sequence diagram for context source polling from a workflow enactment service

8.3 Workflow actions based on context changes

This prototype is made in response to the following functional requirements:

- Responsiveness to contextual changes or events:
 - Ad hoc start of processes and activities.

The state of the contextual environment surrounding a workflow system is dynamic. Sudden changes in the environment may require actions on behalf of a workflow system in response to these events. In the specification of our application scenarios in Chapter 6, we suggested that the detection of an anomaly could be such a context event requiring actions on behalf of the workflow system.

It is necessary to allow the workflow system to set up subscriptions to context sources in the environment and react responsively to the events coming from those context sources. In [41], it was suggested that the CII component could fulfil this task. The workflow enactment engine is not capable of handling events by it self on the level which is required. However, the WfMC's has defined Workflow Application Programming Interface (WAPI) functions in Interface 2 & 3 in the workflow reference model [62]. The WAPI functions can be utilised to achieve flexible handling of context changes by the workflow enactment service. The CII would receive an activity to set up a subscription to a context source according to the publish/subscribe paradigm. The CII would then wait for a specified condition to be fulfilled, before it initiates a specified action in the workflow enactment service through a WAPI call. The most relevant functions in WAPI for responsiveness to context changes are:

- WMCreateProcessInstance – creates a new workflow process instance.
- WMStartProcess – starts a new workflow process instance.
- WMTerminateProcessInstance – stops a workflow process instance.

- WMChangeActivityInstanceState – changes the state of a workflow activity instance.
- WMAssignProcessInstanceState – sets a piece of workflow relevant data.

The CII component needs the following information to effectively handle context subscription and WAPI calls:

- Properties of the context source to subscribe to.
- Condition to be fulfilled before action is taken.
- Name of the WAPI function to call.
- Parameters for the WAPI function call, such as activity id and process id.

This information can be given in the in-parameters to the activity.

To illustrate how a workflow enactment engine can be made responsive to context changes, a simple prototype has been created. This prototype is an extended version of the previous prototype. This prototype consists of two workflow processes each containing an activity. The first process sets up the context source subscription with the CII. As before the context source is a **BooleanWidget**. The CII will initiate execution of the second process through a function call to the workflow enactment engine as soon as the **BooleanWidget** initiates an event indicating the value “TRUE”. The processes are specified in standard XPDL. An illustration is shown in Figure 36 and the complete XPDL document can be found in Appendix G. The illustration for process 1 uses an activity diagram, while the illustration for process 2 uses a state chart diagram.

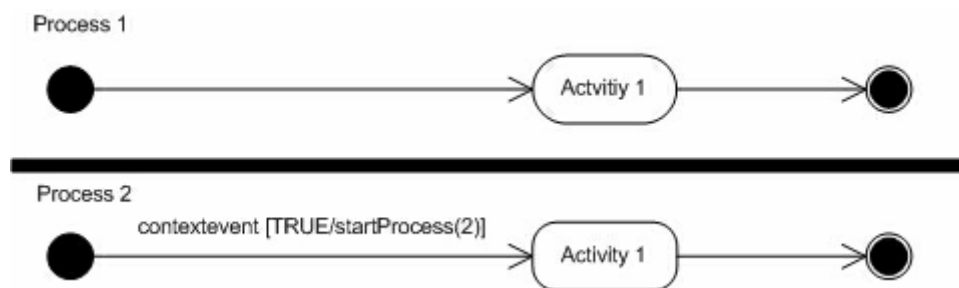


Figure 36: Workflow processes illustration for context source subscription

The implementation of the **BooleanWidget** has been extended to allow for context value generation at specified intervals. The “ContextSource” interface implemented by the **BooleanWidget** supports subscription. The **BooleanWidget** maintains a Vector of all subscribed clients. When a context change occurs, the **BooleanWidget** initiates an update command to all registered listeners. All listeners must implement the “ContextSourceListener” call-back interface. This interface allows the **BooleanWidget** to publish the contextual event. Upon receiving an event fulfilling the specified condition, the CII initiates the specified function call in the “WorkflowClientConnection” interface. In this case, the function to be called launches the second process within the workflow enactment service, as illustrated in the lower half of Figure 36. An illustration of the sequence of events has been provided in the sequence diagram in Figure 37. The **BooleanWidget** is represented by its “ContextSource” interface. The CII component is represented by the

“ContextSourceListener” interface, the CII main class and the “WorkflowClient” interface. The workflow enactment service is represented by the “WorkflowClientConnection” interface and the “WorkflowEnactmentService” main class.

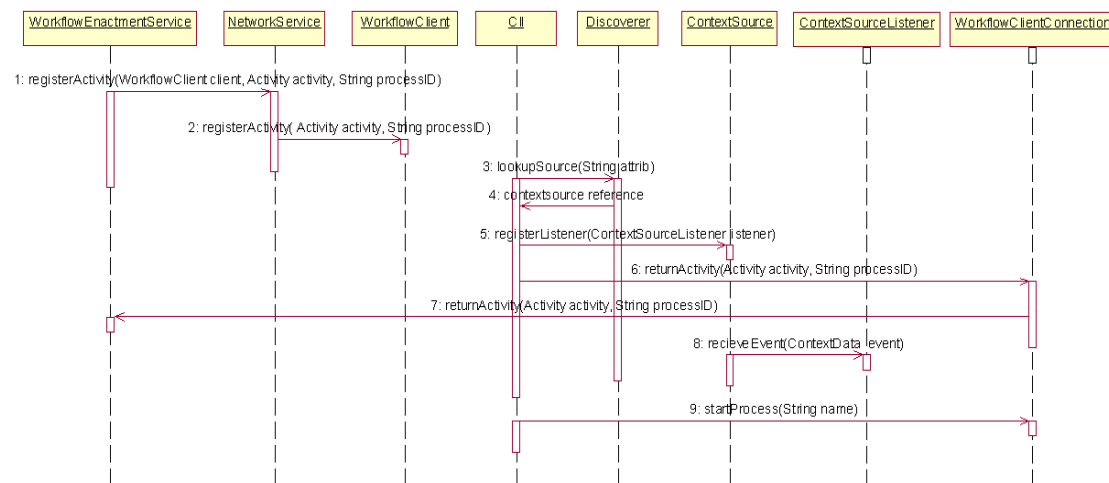


Figure 37: Context source subscription with process initialisation

8.4 Context exception states handling in workflow systems

This prototype has been made to satisfy the following requirements:

- Exception handling of undefined contextual states.
- Support both pre-planned and unplanned process enactment, by providing rule-based process building with learning.

Sensors may provide unreliable results or the state the sensors produce may be unknown, which may prevent **widgets** or other context framework components from determining the exact state of the environment. An example of such a situation can for example be when a sensor connected to a power switch says the power is on because the switch is in that position. However, the equipment using the power says the power is off. The real problem is that the power is not connected, because of a failure outside the sensed contextual environment. This is a simple example, but it serves to illustrate that context information can often be in a state that is unknown to the context-aware application.

When we consider the first prototype, it is easy to see that this prototype requires an absolute response from the CII indicating either the value “TRUE” or “FALSE”. This may not be possible to achieve in every case. Based on the unreliable nature of sensors, one may get an ambiguous reply where absolute values are required. If the value received for the transition attribute is undefined, the workflow processing will stop at the first activity. It is therefore necessary to have the means to handle situations where context information does not follow expected parameters. In existing workflow systems, this is called exception handling. However, exception handling

requires the workflow planner to model the flow of activities in exception situations. We will in this section elaborate on how workflow systems can handle exception situations involving context information without requiring pre-modelling of activities and transitions. The idea is to make workflow standards more in line with the ideas behind activity theory [56] and situated planning [5]. We have developed a prototype to better illustrate our ideas.

8.4.1 Handling context related exceptions

In this section, we will describe a prototype, which handles context related exceptions. This prototype uses our earlier prototype, where context information was used in workflow transitions. The prototype will use ideas from the approach used in the FAR [14] system, presented in Chapter 2.5, for the exception handling part. However, we will expand this approach, since we want the prototype to generate new activities in an exception situation involving context information. This new activity must also become part of the plan, the workflow process definition in this case, according to the learning aspect of situated planning [54].

An illustration of the scenario for this prototype is provided in Figure 38.

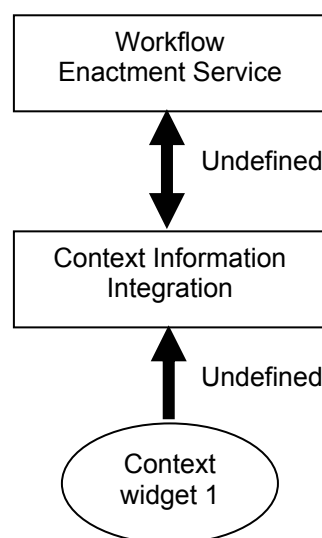


Figure 38: Exception scenario

The scenario we will use in this prototype is similar to the one in the first prototype where we used context information in transitions. The process for this prototype was illustrated in Figure 30. However, the CII will not be capable of generating an absolute response to the context polling activity it is assigned in this instance. This has to do with the fact that the `BooleanWidget` it uses, responds with different return values for the polling request than the values the CII expects. The end result is that the CII is not capable of making an absolute decision on the requested contextual state, and an undefined value is returned to the workflow enactment service. In response to this undefined value, an exception is thrown within the workflow enactment service, which in turn is handled by the exception handler. The exception handler generates a

new activity for a human to verify the requested state, according to specified rules. Depending on the result found by the human workflow participant, the system makes the final transition to one of the final activities, finishing the workflow process. The original workflow process flow diagram is the same as in Figure 30. After the new activity involving a human participant verifying the contextual state, the process diagram is changed. The revised process is illustrated in Figure 39. Activity 1 is the context polling activity. Activity 2 and 3 are activities with human performers. Activity 4 is the verification of the context information activity. The complete generated XPDL for the new process description is available in Appendix H.

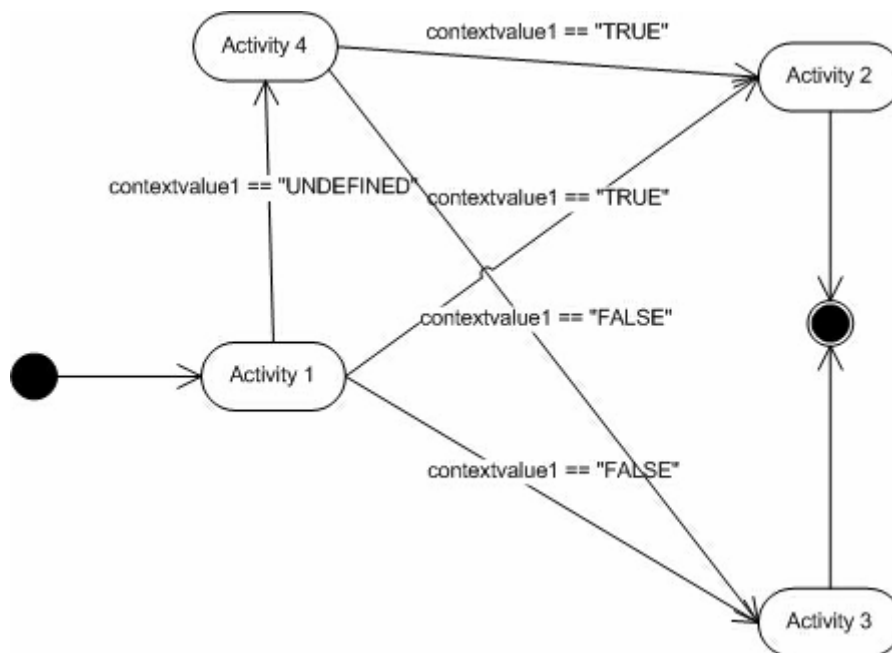


Figure 39: Exception handling revised process illustration

8.4.1.1 The Exception handler component

The approach used for the exception handling component for this prototype is similar to the FAR architecture [14]. A class diagram for the WorkflowEnactment.ExceptionHandler package is illustrated in Figure 40.

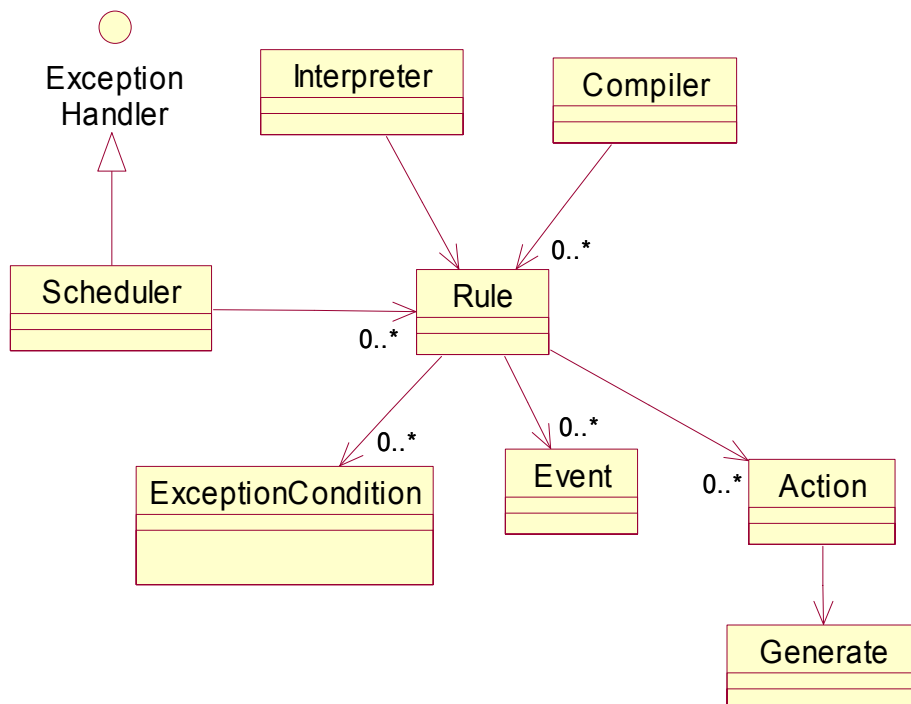


Figure 40: Class diagram for the exception handler package

This exception handler uses Event-Condition-Action (ECA) rules, similar to rules in Chimera-Exc [14]. These exception handling rules have to be defined in XML according to a given XML schema presented in Appendix I. The rules follow the idea behind ECA rules, which have an event, condition and action part. The exception handling document specifies a TransitionException event in this scenario. This event occurs in response to the “ProcessExecuter” thread not being able to find any transitions to follow and the current activity is not a final activity. The condition part of the exception handling XML document specifies that the transition value has to be “UNDEFINED” in order for this rule to fire. The event part of the exception handling rule specifies a “Generate” action. This action specifies what activities and transitions to generate based on the current state of the “ProcessExecuter”. The full text for the exception handling XML document is available in Appendix J.

The “Compiler” class is responsible for interpreting the exception handler definition document. The document data is forwarded from the DataService component of the workflow enactment service to the “Compiler”, which interprets the data and creates an internal representation.

The “ProcessExecuter” thread generates transition exceptions as necessary. The “TransitionException” class has been defined in the WorkflowEnactment.Exceptions package together with other exceptions. This package also includes an ExceptionOwner interface, which the “ProcessExecuter” class implements. This interface serves two purposes. Firstly, it serves as the link for the exception handler to access the process in question. Secondly, it provides methods for the exception handler to manipulate the workflow process being executed by the “ProcessExecuter”. The “Scheduler” class of the exception handler evaluates which rule has fired, when a

request for exception handling comes in through the “ExceptionHandler” interface. The exception and the rule, when fired, are then forwarded to the Interpreter class. This class runs as a separate thread. It accesses the representation of the rule and manipulates the workflow process, where the exception arose, through the “ExceptionOwner” interface. The new process elements are built both based on the current state of the workflow process and the defined data in the exception handler definition document. Upon completion of the exception handling the “ProcessExecuter” thread is resumed.

We are in essence building a new process, by adding new process elements to the existing process to handle an exception situation. These new elements become part of the permanent process to satisfy the requirement for learning, which in turn allow the workflow enactment service to function as a “technology of accountability” [5]. The DataService component receives the new process elements and maps the process elements into the DOM representation internal to the “DataService”, which in turn can be written to a file as XPD. An interaction diagram for the processing of an exception has been provided in Figure 41. Please note that the “ProcessExecuter” is represented both by the “ProcessExecuter” class and the “ExceptionOwner” interface.

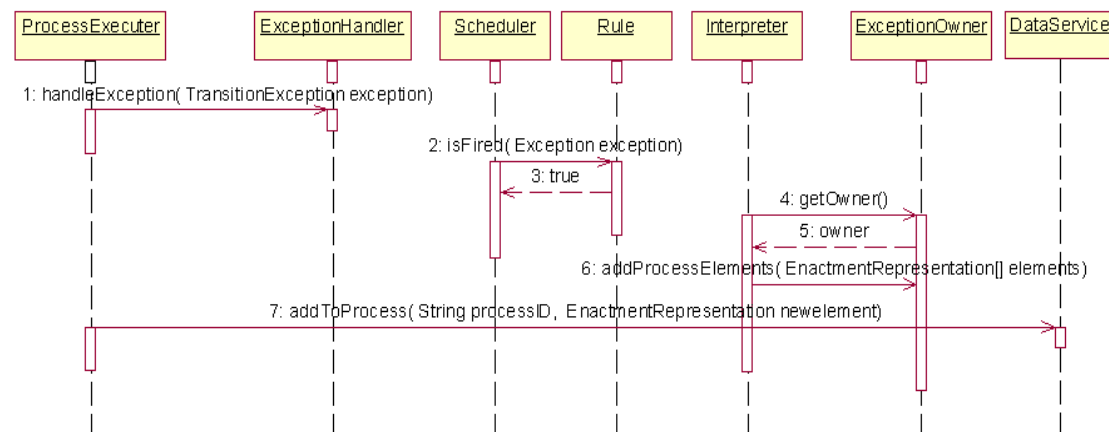


Figure 41: Sequence diagram for the exception handling scenario

8.5 Process path revalidation

This prototype has been made in response to the following requirements:

- Support revalidation of selected process paths, if the current path does not lead to the process goal.

It is easy to envision scenarios where specified transitions for the current process path are not satisfied in a dynamic contextual environment. Excepted conditional conditions may not be satisfied under previously found contextual conditions, which has lead the process enactment into a process path that cannot be completed. It would therefore be advantageous for the workflow enactment service to re-evaluate the previous context based transitions made. For example, during the execution of a maintenance process it is discovered that a needed part is not available. If the specific

maintenance process path was selected on the assumption that all parts were available, other possible maintenance process paths may open up at the process split point. This can for example be to replace an entire system.

We will use a process definition with six activities for this prototype. Figure 42 illustrates the workflow process. Activity 1 queries the CII for the value of contextvalue1, while Activity 4 and 5 queries for the value of contextvalue2. Activity 2, 3 and 6 are activities performed by other workflow participants.

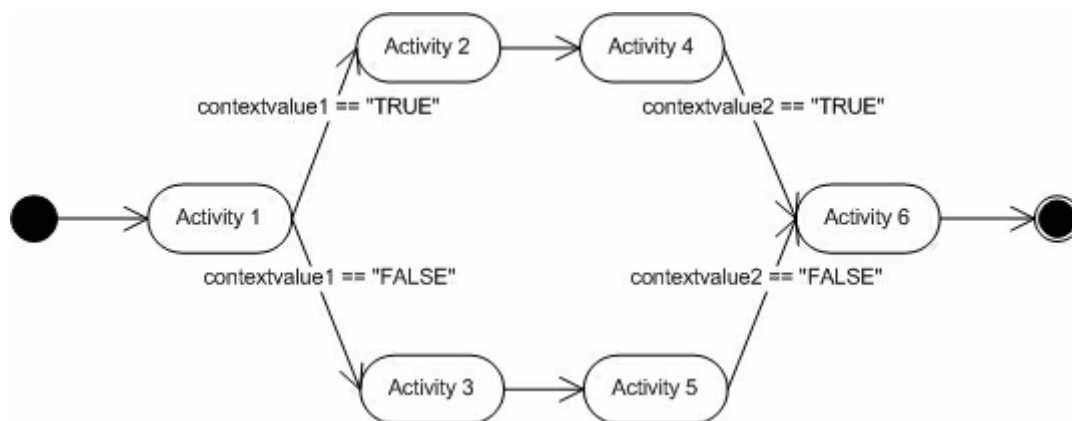


Figure 42: Process path revalidation process illustration

This process will not lead to process completion if contextvalue2 is of the opposite value of the specified value for either of the process paths. A process path revalidation can therefore be initiated. The complete process specification is given in Appendix K.

The task of revalidating the process path and possibly rolling back to an activity before a split is left up to the exception handler developed in the previous prototype. We specify an additional rule for this exception condition. This exception is also caused by a TransitionException event, but the action specified is a RevalidateProcessPath action. The full text for the exception handler rule description can be accessed in Appendix L.

The process path is then re-evaluated by the “Interpreter” in the exception handler through the use of the methods in the “ExceptionOwner” interface. Re-evaluation is accomplished by reversing the activity history located within the “ProcessExecuter” class. If a split which evaluates differently is found, the current state of the “ProcessExecuter” is set to the activity which had the split. The state change is done by rolling back the activities step by step. Activities, which are not context polling activities, are rolled back by sending a “rollback” activity to the workflow client, which previously had this activity. The “ProcessExecuter” will then re-evaluate the transition split and follow another process path.

8.6 Invariant scenario

This prototype has been made in response to the following functional requirements:

- Responsiveness to contextual changes or events:
 - Correct handling of invariant conditions.

Some workflow processes can be dependent on previously verified contextual states, called invariants, which state remains unchanged until process end. If for some reason they are altered, the entire process is invalidated and enactment has to be interrupted. This interruption can involve either restarting the entire workflow process or restarting enactment from the state where the invariant invalidation occurred. Invariants are especially important when dealing with security related conditions. Such conditions can for example be the existence of gas in an area where work is performed or the pressure acting on valves, which has to remain within tolerance when work is performed.

We have made a prototype, which addresses the requirement for correct handling of invariants. The workflow process, which is enacted in this prototype, is illustrated in Figure 43. Here activity 1 specifies that contextvalue1 is an invariant for the entire process. Activity 4 and 5 are normal context polling, while activity 2, 3, 6, 7, 8 and 9 are activities performed by workflow participants. The entire process description in XPDL is provided in Appendix M.

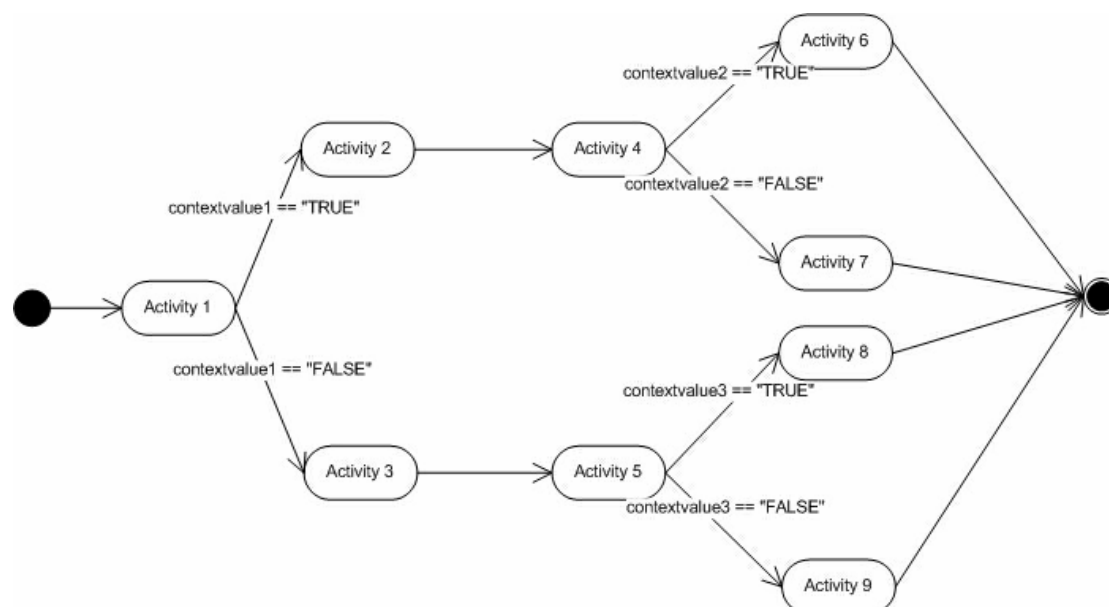


Figure 43: Invariant scenario process illustration

Activity 1 specifies both a context source polling for the value of contextvalue1 and a subscription to the same context source. Similarly to the prototype, where a subscription event caused process start, this subscription also specifies a method to call in the workflow enactment service, when an event satisfying a condition occurs. The condition is specified using a notation, which causes the CII to replace the value

specified with the opposite of the current value. This causes the CII to only notify the workflow enactment service of an invariant exception when the value has changed. Once such an event occurs, the CII notifies the correct process in the workflow enactment service. This forwards an “InvariantException” to the exception handler component. When the “ProcessExecuter” enters a non-critical section such as when an activity has been sent to a workflow client, the “Interpreter” thread of the exception handler takes control of process. This is achieved by synchronising the threads on the “ProcessExecuter” object itself.

The exception handler rule for this scenario specifies an “InvariantException” event. The action portion specifies a “RevalidateProcessPath” action, and the exception is handled similarly. The full text for the rule is available in Appendix N. The workflow process is rolled back to the activity specified in the exception object. This activity is the same activity which set up the subscription to the invariant in the first place. This activity is then redone to remove the old invariant subscription and instantiate a new subscription with the CII. When exception handling is completed the “ProcessExecuter” thread is notified and processing continues from the place where it left off with the new process state.

8.7 Client based context-awareness

In this section, we will examine our prototypes related to workflow client based context-awareness. The main issues examined are situated planning with process building and post-conditions for activities.

To specify the usage of context information in the building of the local workflow process for the current activity, it is necessary to specify rules for this usage. A workflow client would then need to have a rule based inference engine, which process these rules and builds the process.

8.7.1 Inference engine

Inference engines are common parts of expert systems and other systems using knowledge bases. An inference engine infers new knowledge from existing knowledge by using previously defined sets of rules.

An inference engine can use several methods of inferring new knowledge. Backward chaining [92], forward chaining [93] and search trees [94] are all possible ways of inferring new knowledge. In forward chaining, one starts with one or more facts, then these facts are processed against the defined set of rules. Each rule that is satisfied, infer new facts. When no more rules can be satisfied, one has arrived at the goal state. Backward chaining takes a different approach in that the system starts with the desired goal and tries to find rules, which are satisfied and leads to the known facts, and thereby proves the goal. Search tree methods take advantage of the fact that most knowledge bases can be represented as trees. An inference engine iterates through the search tree using a variety of techniques. This iteration starts with either the given data or the goal, such as was described for backward chaining and forward chaining.

8.7.2 Workflow client with local process enactment and rule based building of the situated process

This prototype has been built to satisfy the following requirements:

- Perform situated planning based on current contextual conditions.

Situated planning [5] involves building the process in situ. We provided scenarios where situated planning might be advantageous in Chapter 6.

It is not possible to always know all possible states of a contextual environment, so a complete process can be built. This means that we build the process in the field. However, it is not possible to build a process without some guidelines so we are going to use a knowledge base, which a simple inference engine can infer new knowledge from. The inference engine ascertains the current contextual state and builds the process by stating transitions as facts during the inferring process until the goal of workflow process completion is reached. The situated activities are already provided by the enactment service.

The overall process in the enactment service specifies only one activity. This activity is an activity to be performed situated with rule-based building of the local process. Sub-flows can be specified as part of activities. These are normally enacted by the workflow enactment service. We will, however, allow a workflow client to carry out this enactment by saying that a specified performer can carry out such a process definition. This violates the WfMC Interface 1 [63] specification, but does not require changes in the XPD L language itself. The first activity in this sub-flow has an in-parameter with a knowledge base specifying the inference rules. This is a separate XML document and can be specified as workflow relevant data. The tool for this activity is the inference engine. All possible activities are specified as normal in this sub process. Once rule based inferring has been completed a process has been specified, with activities and transitions between them. Please note that the transitions will have no conditions connected to them. The overall process is illustrated in Figure 44. Above the black line, the overall workflow process is illustrated. This process only contains one activity. This activity has a sub process implementation and is sent to a workflow client for enactment. Below the black line is the local sub process after it has been built, based on the result provided by the inference engine. The inferring process is based on the context values generated from two context sources. The sub-process defines six activities, to which transitions can be defined based on the results of the inferring. Activity 1 in the local client based sub process is enacted in every case, since this activity specifies the rule based inferring. The complete process definition including both the overall process and the sub process is available in Appendix O.

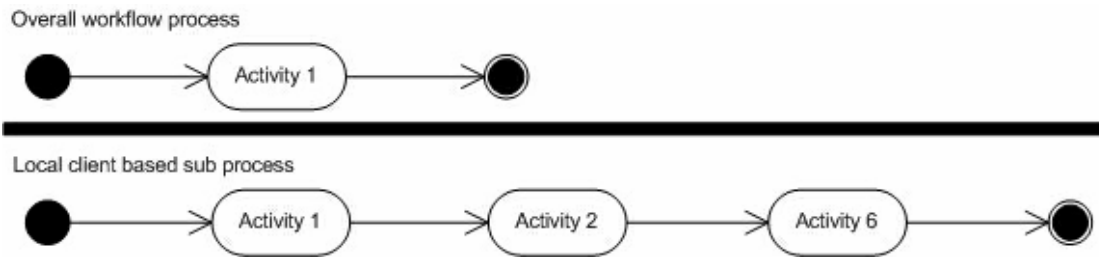


Figure 44: Processes for client based context-awareness prototype

The inference component is located in the “InferenceEngine” package under the “WorkflowClient” package. The “WorkflowClient” class is also joined by the “LocalProcessExecuter” class in the “WorkflowClient” package, which enact a single local process. The class diagram for the updated workflow client is shown in Figure 45.

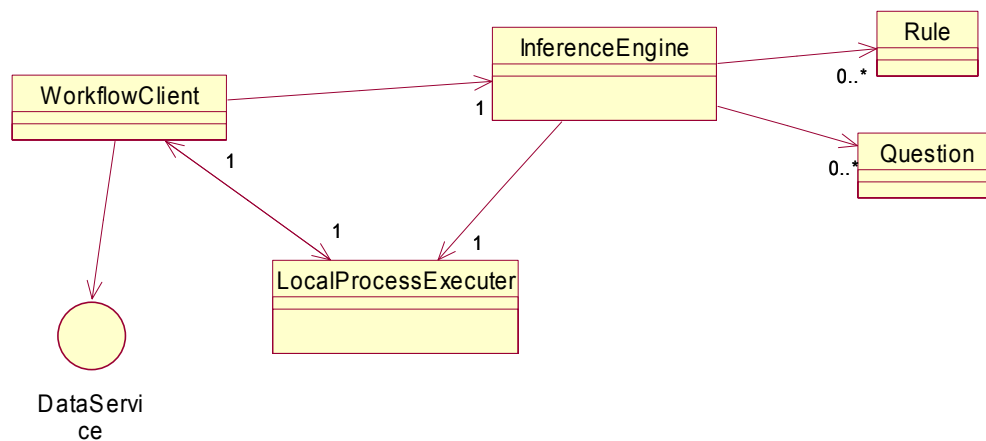


Figure 45: Class diagram for the updated workflow client

The inference engine is based on forward chaining [93]. Facts are acquired from the contextual environment. During rule processing the transitions between activities are created through the use of methods in the “LocalProcessExecuter” class. Transitions are built based on the activity mentioned in the inference rule and the last activity, which has a transition linking it to the other activities of the process. The support classes “Rule” and “Question” specify respectively inference rules and context sources to poll.

The “WorkflowEnactment.DataService” and “WorkflowEnactment.EnactmentRepresentation” packages had to be moved from the “WorkflowEnactment” package to the “Common” package. This has to do with the fact that the workflow client has to represent a workflow process and load the process data from XML. In addition the inference rules also have to be loaded from XML. This means that the data service component and the enactment representation classes are shared between the workflow client and the workflow enactment service.

The inference rules used during inferring is provided in Appendix P.

8.7.3 Workflow client with activity contextual post conditions

The enactment of workflow activities may change the contextual environment. This is the basis for this prototype, which satisfy the following requirements:

- Context as post-condition for activities.

Most activities influence the environment in some form. However, some activities may be directly dependent on context sensors to know when to finish the activity. An example of this is manipulation of valves to fill tanks. Sensors monitor the state of the tanks and form the basis for post-conditions for the activity.

The XPDL language does not allow specification of post-conditions internal to activities. However, we can use in-parameters to specify post-conditions for activities. In this prototype, we will specify a context information based post-condition for an activity and the workflow client will only return the activity when the condition is satisfied. This means that the workflow client has set up a subscription to a context source. It is necessary to have the client implement the “ContextSourceListener” interface, to allow it to receive context events. This means that the workflow client implements the same interfaces as the CII component.

The context source used by the workflow client must be capable of accepting conditions for context event sending to avoid overloading the workflow client. Without such conditions the workflow client might easily be overloaded with context events.

The process used in this prototype is similar to the overall workflow process illustrated in Figure 44. However, activity 1 represents a single activity for a specific workflow client. The activity definition also includes a post-condition for the activity defined in parameters to the activity. Once the event satisfying the post-condition has been received, the activity is returned to the workflow enactment service. The complete process description is provided in Appendix Q.

Figure 46 illustrates the interaction between the components in this prototype. The workflow client is represented by the “WorkflowClient” interface, the “ContextSourceListener” interface. The workflow enactment service is represented by the “WorkflowEnactmentService” class, the “WorkflowClientConnection” interface and the “NetworkService” interface.

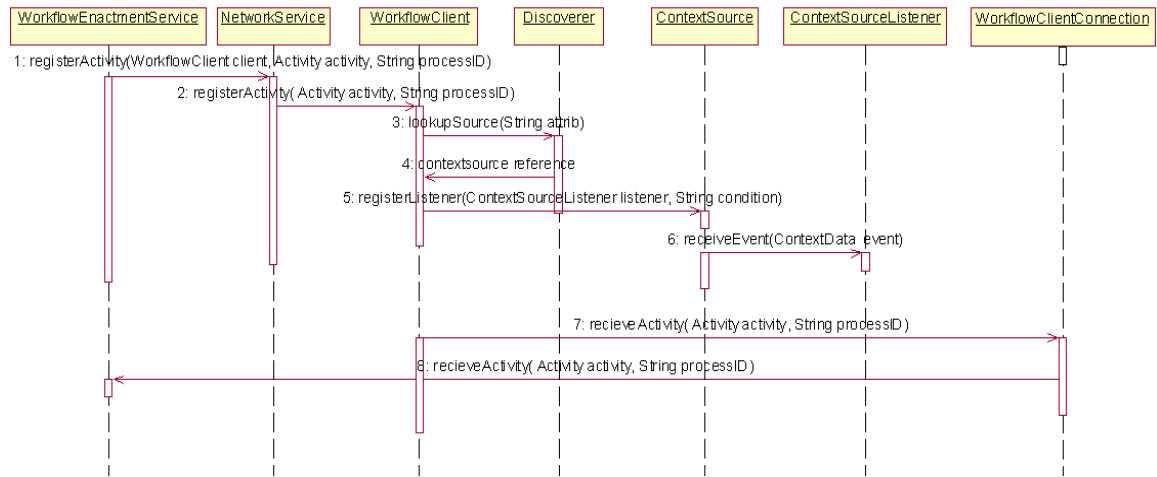


Figure 46: Sequence diagram for activity post condition

Part III: Discussion and conclusion

This part presents the discussion based on the represented requirements and prototypes from the previous part. Based on the discussion, suggestions for further work are presented. Finally, we make some conclusions based on the discussion and previous parts.

9. Discussion

We will discuss the prototypes developed in this chapter. The discussion will be related to the research questions. How the functionality presented in the prototypes fit with workflow standards is another important topic of this chapter.

9.1 Discussion of prototypes

We implemented several prototypes, which each focus on a different set of requirements for a context-aware workflow system.

9.1.1 Context information used in the processing of workflow transitions

Our first prototype presented in Chapter 8.2, focused on how context information can effectively be integrated with a workflow system and used in workflow enactment. The centralised workflow enactment service was based on the Interface 1 specification [63] by the WfMC. The basic idea was to acquire and use context information while following WfMC standards. This was achieved by using an external component. Process paths selection is based on the state of workflow relevant data according to the WfMC Interface 1 specification. The workflow relevant data is updated during activity enactment at the workflow clients by specifying in- and out-parameters to activities. We made use of this functionality when we designed our external support component. This component, named CII, is capable of receiving workflow activities and updates the necessary data by polling the components of the underlying context framework. This solution for context integration, while satisfying only the requirement for context information acquisition and usage in process reasoning, is achieved completely within WfMC specifications. A solution without using an external component would not have been possible without the context framework components being able to accept activity objects directly from the workflow enactment service. Even if the context framework components supported receiving activities directly, the necessary process description would increase. This is based on the fact that a simple poll of a context source consists of both the lookup operation of a discoverer service and the actual polling of a context source. This means that context source polling would have to be modelled as two separate activities with different performers.

9.1.2 Workflow actions based on context changes

For the second prototype, the functionality of the CII component was extended. It is now capable of setting up subscriptions with context sources. The same approach used for the previous prototype was reused for this prototype. An activity specifies the

details of the subscription and the CII initiate the necessary operations based on the activity description. Once the subscription is set up, the activity is returned to the workflow enactment service. The subscription is specified with a simple condition to be fulfilled before action is taken. WfMC specifies several methods in an API for Interface 2 and 3 in the workflow reference model [62], called the WAPI [60]. These methods enable external applications to alter the state of individual processes and manipulate the workflow enactment service directly. This functionality was used in this prototype. We implemented methods, which were made available to the CII. The naming of the methods and the parameters were not identical with the WAPI definition, but equivalent methods to the methods implemented by us exist in the WAPI definition. The CII could then use these methods to do the specified operation in the subscription activity.

The chosen solution for this prototype has a few drawbacks. First of all, the use of a simple contextual condition as the basis for operations in the workflow enactment service does not allow for responsiveness to complex contextual situations. A context situation is usually complex with multiple context sources generating information. The sum of this information forms the overall state of the environment, which in turn may result in the need for operations performed to the workflow enactment service. This means that we need to specify rules for each operation, which can be used in an inference engine to discover whether the operation is necessary. The sentient object [21] abstraction becomes appropriate for the CII component now. The input for the CII component is sensor information. With an inference engine and the workflow enactment service as actuator, it fits the definition perfectly.

By separating the CII from the workflow enactment service itself, loose coupling and separation of concerns are supported. However, it is possible to envision scenarios where a direct connection between the workflow enactment service and the CII component would be beneficial. The WAPI covers operations connected to pre-planned processes. In the case of unplanned, situated processes, no support is offered. If situated process planning based on context information is to be performed, direct access to update the process definitions is needed. Such functionality is best achieved with the component responsible running as part of the workflow enactment service. This component would then form a new interface to the workflow enactment service.

9.1.3 Handling context related exceptions

Contextual exception situations were the focus for this next prototype. To enable handling of such exception situations, a workflow exception handler was created. This exception handler used rules built similarly to ECA rules. The rules specifies events, conditions and actions to be performed when an event satisfying the conditions have been received. The exception in the prototype was generated internally during process enactment in the workflow enactment service. The CII simply returned a value indicating an undefined state. An exception handler is ideal for handling pre-defined exceptions. Such exceptions can potentially arrive from a variety of sources. The chosen solution for exception generation in this prototype means that the exception situation has to be “discovered” during process enactment. Exception situations requiring more direct action would have to wait to be discovered by the workflow

enactment service during process enactment. A possible solution to this problem would be to generate the exception in the CII component and send it to the workflow enactment service. The exception handler could then handle the exception immediately. The exception handler interface could also be directly available as a separate interface to the workflow enactment service. However, this means that we open another interface to the workflow enactment service, which is strictly not necessary since only the workflow enactment service and the context information integration component need access to the exception handler. If the CII is integrated with the workflow enactment service itself, it would be able to send the exception directly to the exception handler.

Another focus area for this prototype was accountability. By updating the process description based on the result of the exception handling, we were able to make what the system knew about the context situation available for review by the user.

9.1.4 Process path selection and invariants

The prototypes, which focused on process path revalidation and invalidation of invariants, are examples of exception situations. The process path revalidation prototype focused on exceptions internal to the workflow enactment engine. The idea was that greater dynamic behaviour would be achieved, if the workflow enactment service would be able to revalidate process paths selections made earlier. The previously developed exception handler work perfectly in that respect. The exception handler “rolled back” the process to a point where the context situation opens another process path. This dynamic behaviour is not found in existing workflow systems, but would be necessary where dynamic context is used as the basis for process path selection.

Invalidation of context related invariants is an example of an exception situation, which requires immediate action by the workflow enactment service. In the invariant scenario prototype, we used our external CII component to set up subscriptions to context sources, which constituted invariants in the process. The activity specifying the invariants also specifies which method to call in the workflow enactment service if the invariant was invalidated. Once the method is called, an exception is forwarded to the exception handler. The chosen solution with an external CII and an internal exception handler means that we have to implement methods to which there are no equivalents in WAPI.

9.1.5 Client based context-awareness

We developed two different prototypes, which focused on workflow client based context-awareness. The first prototype put the workflow client in the situated activity scenario.

Mobile workflow participants moving through a pervasive computing environment can receive information from many context sources. Much of this context will be irrelevant for the current activity the participant is performing. In addition the mobile

device used by the participant may be incapable of processing all context information received from the environment. At the very least too many context sources will be a heavy strain on the bandwidth and battery capacity of the mobile device. It will therefore be necessary to limit the context sources used by the mobile device. The best way to do that is to find the context source attributes desired from the current executing activity. The context information gained from the context sources can then be used to build a more detailed mini process flow for the goal of completing the current activity based on the current state of the environment.

The problem of creating a workflow process flow based on the current state of the environment can be approached from several angles. In traditional workflow systems, a central server processes all workflow enactment. This approach may not be possible dependent on the scenario in question. In a scenario where all sensors are connected to the central workflow system, the central workflow enactment server would be the obvious place to build the process flow for each activity based on the current environmental state. However, in a scenario where the location is remote and not connected to the main workflow enactment server, this approach has disadvantages. It would still be possible for the mobile device to collect the context information and forward the information to the central server. The mobile device would then receive activities from the workflow server as usual. This requires, however, that the mobile device has a connection to the workflow server at the particular location, something which may not be possible. Considering this requirement, it would be advantageous to allow the workflow client in a mobile device to perform local enactment based on the current activity.

In [41], it was suggested that it would be necessary to provide a policy for the management of context information for each activity. This policy definition must include the attributes of the contextual sources to be used, the polling frequency or context source subscriptions and the usage of the context information. The context acquisition technique best suited for client based context-awareness varies according to the needs of the client. If the client only needs to access a specific context value at a specific time, a polling approach is obviously best suited. However, if a client needs to receive updates on contextual changes, a subscription approach may be better suited. This requires that a context source is capable of receiving conditions for subscriptions. If this functionality is not in place, a workflow client on a mobile device will get swamped with possibly useless context updates. Even with such a condition based subscription mechanism, the client may still be overloaded with events. A high frequency of incoming context information updates from the environment may also pose problems for resource limited mobile devices. The mobile device will burn valuable energy receiving and processing context updates. The polling approach allows the mobile workflow client to control the context acquisition frequency.

The actual prototype implemented used a previously specified sub-flow together with a knowledge base connecting activities to context states. The knowledge base was used in an inference engine located within the client. The knowledge base specified context sources to poll, to be able to build the transitions in the situated sub-flow. According to the Interface 1 specification [63] by the WfMC, sub-flows are meant to be executed by the workflow enactment service. In that respect we violated this specification. However, we were able to use the performer attribute of the owner

activity of the sub-flow as the link between the sub-flow definition and the executing entity.

We only utilised context source polling in the prototype, which focused on situated process building. This is limiting since the situated process will remain static until the process is completed. If a subscription policy had been used, the process could have been reconfigured dynamically throughout the process.

Workflow client based context source subscription was implemented in our final prototype, which focused on contextual post-conditions for activities. The added implementation of context source subscription meant that the workflow client had all the functionality of the CII component. For situated activities, the workflow client does require all the functionality of a fully integrated CII as specified for the workflow enactment service. This component will have to be able to perform situated planning and context information acquisition with a supporting inference engine. In addition context exception situation must also be handled in some form of exception handler for the workflow client. This is based on the fact that the workflow client needs an independent local enactment service for situated activity enactment.

9.2 Evaluation of research method

Through the use of a combination of scientific and the engineering research method, we were able to see the problem of context-awareness in workflow systems from both a high-level view of the problem to the low-level implementation view. We could realise our theories in implementations of a workflow systems and actually see how well our solutions fitted with workflow standards. Even though we could probably have stopped at the design level and gotten the same results, the context-aware workflow prototypes functions as a good demonstration tool for our ideas.

We chose vertical prototyping as our prototyping technique. This prototyping technique is good to demonstrate the features of a product. However, a complete product does not have to be made. The implementation of a complete product requires a much more extensive set of requirements than what was needed for our prototypes. We found it impossible to satisfy the requirements for a complete product in the allotted time for this project. A fully implemented and usable context-aware workflow system tested in an environment containing smart sensors [46] and augmented artefacts would have given us valuable insights into the usefulness of this technology.

10. Future work

Several of our research questions were not properly answered based on our work with the prototypes and our discussion. A few simplifications were also made. This chapter will discuss some of these issues, to provide a reference for future work.

We used a Boolean context source. This solution had the advantage of limiting the possible values to true and false. In addition, we had an undefined value, indicating an exception condition. However, it is not sufficient to limit context values to Boolean values. In fact, all possible data types defined in XPDL [63] should be supported. Additional data types can also be defined using XML Schemas [103]. With more possible values for context information, the number of possible process paths increases. For example with N possible context values from a specific source and M process paths splits based on this context source the number resulting process paths would be N^M . This is of course the worst case scenario if no grouping of values is possible. To avoid such scenarios one should try to aggregate context values to reduce the number of permutations. This aggregation can both be conducted in the underlying context information framework and the context inference engine connected to the workflow system.

Inference engines deal only in absolute values of true or false. Fuzzy logic [100] is a super set of normal Boolean logic. It is possible to have partial truths in fuzzy logic. This is useful when with dealing exception conditions for context values, where the exact state cannot be ascertained. It would be possible to use fuzzy logic in the inference engines of a context-aware workflow system to handle a wider array of context states without involving an exception handler. This should be investigated further.

The activity coordination aspect is not fully explored in this report. Activity coordination should be possible in the workflow enactment service for both pre-planned and unplanned situated processes. The same is true on the workflow client level for situated activities. A workflow process can contain both cooperating and competing workflow participant. The workflow enactment service provides process coordination of activities already by controlling the movement of activities. When using dynamic process building for situated processes, the coordination is implicit.

For the workflow client, the coordination of situated activities is more complex. Each situated activity should specify its relation to other situated activities. The other workflow clients then become context sources for the workflow client running a situated action sub process. By using the state of the other workflow clients, which runs situated processes, as context in the local situated process enactment, we are able to control how work progresses. However, this aspect of coordination requires further study.

We have not performed any functional test of a context-aware workflow system. A functional context-aware workflow system should be tested in a pervasive computing environment containing smart sensors [46] and augmented artefacts. This will help us

to understand the usefulness of context-aware work processes. Preferably, the test should be conducted using real and relevant process descriptions.

11. Conclusion

We will summarise some conclusions based on our work in this chapter. The conclusions will be based on the discussion and our prototypes, but we will also relate our conclusions to the state-of-the-art part.

We discovered that it possible to use context information in workflow transitions without changing any of the WfMC standards. We accomplished this with an external component, which gathered the context information from the environment on behalf of the workflow system and made this information available by activity parameter passing. Simple responsiveness to context changes was also achieved by using the functionality of the WAPI [60] and an external component. The dynamic behaviour achieved by this approach is limited to the methods defined in the WAPI.

Based on discussion of situated planning and situated activities, we must conclude that a context information integration component is best integrated with a workflow enactment service, since this component needs to make radical changes to processes as they run and build processes based on current contextual conditions. Such a component should be built around the principle of a sentient object [21]. This component, which is named CII, would effectively form a new interface for the workflow enactment service. This is illustrated in Figure 47.

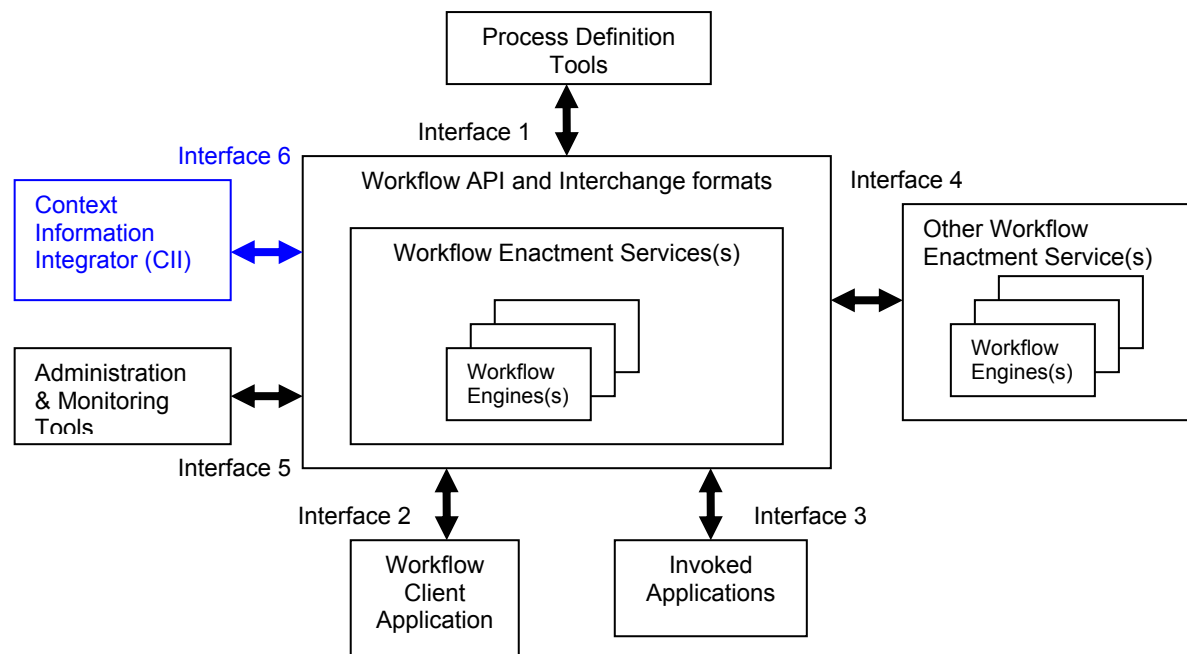


Figure 47: WfMC workflow reference model with new interface.

The workflow client needs a local workflow enactment service to satisfy the requirement for situated actions. This means that similarly to the workflow service, the client needs to build situated local processes, based on local contextual conditions.

This situated planning should also be based on inference rules. So the workflow client needs a component with the same functionality as the CII component of the workflow enactment service.

The handling of context related exception situations is another important aspect of a context-aware workflow system. An exception handler for a context-aware workflow system should be able to handle exceptions generated during process enactment and exceptions based on context. The exception handler can be used to increase the dynamic behaviour of a workflow enactment service with regard to process path selection. The same is true for the local enactment service of the workflow client.

The WfMC defines two types of activities. Activities, which cannot be run by calling applications, are called manual activities. Automated activities can use applications called directly. Situated activities fall in between these two definitions. A situated activity is done manually by a human participant, but the way the activity is completed is controlled by automatic means. We will therefore suggest a semi-automated activity, which is defined in between the manual and automated activities. This is illustrated in Figure 48, which is an edited version of the basic terms and relationships illustrated in Figure 3.

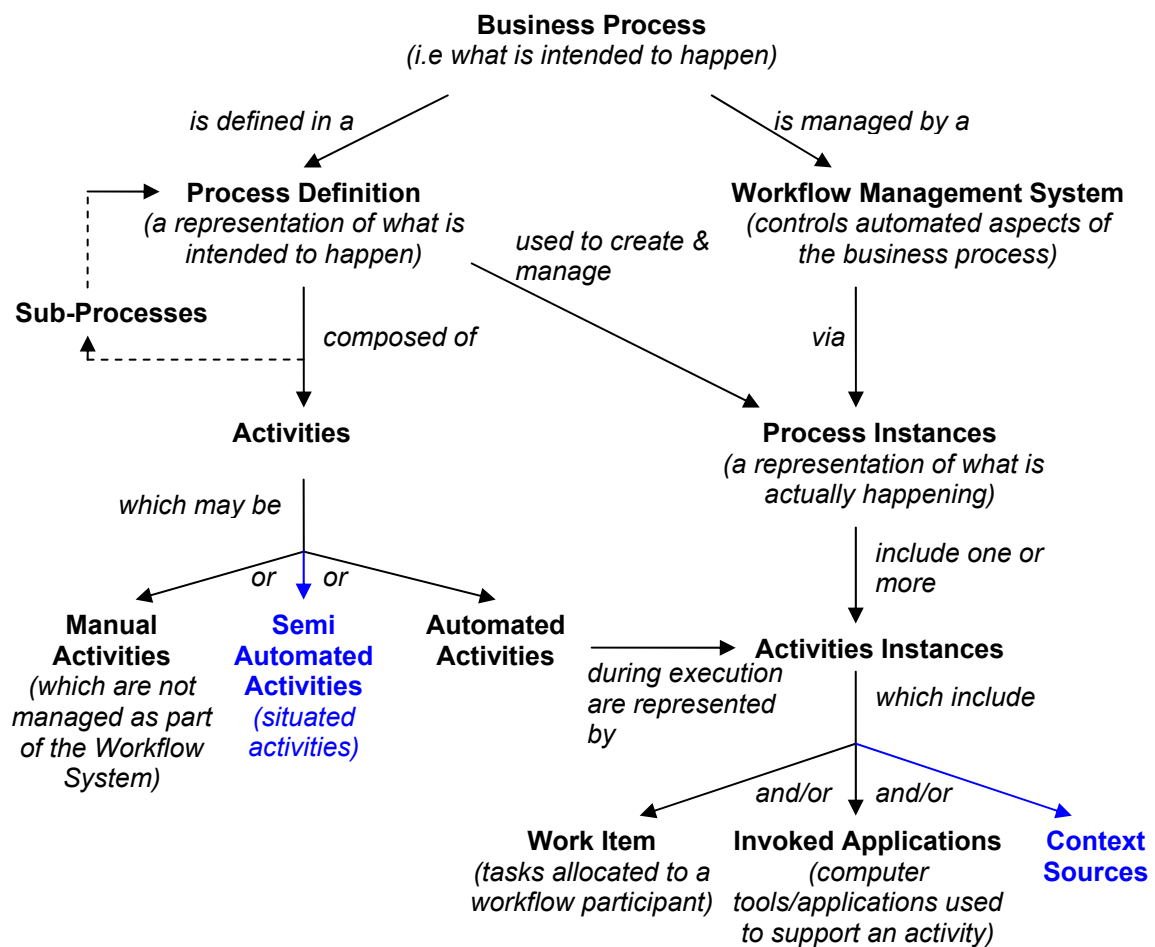


Figure 48: Revised basic terms and relationships

Activities consist of work item data and invoked applications. For workflow activities to be performed situated, context sources perform a vital role. We have therefore chosen to include context sources as a part of activity instances, which is illustrated in Figure 48.

Glossary

A	
API	(Application Programming Interface) A set of routines that an application uses to request and carry out lower-level services performed by a computer's operating system. For computers running a graphical user interface, an API manages an application's windows, icons, menus, and dialog boxes.
Augmented artifact	A device with built-in processing capability, memory, wireless communications and its own power source.
B	
C	
COM	(Component Object Model) An architecture for making component based programs on the Windows system.
CORBA	(Common Object Request Broker Architecture) An architecture allowing communication between components in a system.
CSCW	(Computer-Supported Cooperative Work) CSCW should be conceived as an endeavour to understand the nature and requirement of cooperative work with the objective of designing computer-based technologies for cooperative work arrangements.
D	
DCOM	(Distributed Component Object Model) An extension of the Component Object Model (COM) that allows COM components to communicate across network boundaries.
DOM	(Document Object Model) The DOM is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.
E	
ECA rule	Event-condition-action rule
E-commerce	(Electronic Commerce) Business that is conducted over the Internet using any of the applications that rely on the Internet, such as e-mail, instant messaging, shopping carts, Web services etc.
EPOS	(Expert System for Program and System Development) A Software Engineering Environment with emphasis on Process Modeling, Software Configuration Management and support to cooperative work.
F	
G	
Groupware	Computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment

H	
HTTP	(Hyper Text Transfer Protocol) The protocol used between web servers and browsers on the WWW.
I	
IIOP	(Internet Inter-ORB Protocol) Enables browsers and servers to exchange integers, arrays, and more complex objects, unlike HTTP, which only supports transmission of text.
IP	(Internet Protocol) A protocol, which specifies the formats of packages and an addressing scheme.
J	
Java	Java is a object-oriented programming language developed by Sun Microsystems Inc. Java is an interpreted language. This means that in order for a Java program to run on a computer, a run-time system (interpreter) will need to have been installed on the computer.
K	
L	
LAN	(Local Area Network) A computer network that spans a relatively small area.
M	
Microsoft .NET	A Microsoft operating system platform that incorporates applications, a suite of tools and services and a change in the infrastructure of the company's Web strategy.
MOWAHS	Mobile Work Across Heterogeneous Systems
N	
O	
P	
PDA	(Personal Digital Assistant) A handheld device which combines mobile computing and networking features.
Pervasive computing	See Ubiquitous computing.
Q	
R	
RDF	(Resource Description Framework) A general framework for describing a Web site's metadata, or the information about the information on the site.
RMI	(Remote Method Invocation) A set of protocols, which enables distributed Java object to communicate.
S	
SOAP	(Simple Object Access Protocol) A lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before sending them over a network.
SPADE	(Software Process Analysis, Design and Enactment) Is a research process modelling environment. Its main objectives are to support the analysis, design, and enactment of software processes.
T	
TCP	(Transmission Control Protocol) The end to end protocol used in TCP/IP networks.

TCP/IP	TCP over Internet Protocol (IP).
U	
Ubiquitous computing	Is a term meaning the seamless integration of computing into the fabric of everyday life.
UML	(Unified Modeling Language) A general-purpose notational language for specifying and visualising complex software, especially large, object-oriented projects.
V	
W	
W3C	(World Wide Web Consortium) An international consortium of companies involved with the Internet and the Web.
WAN	(Wide Area Network) A computer network that spans a relatively large geographical area. Typically, a WAN consists of two or more local-area networks (LANs).
WAP	(Wireless Application Protocol) A specification allowing immediate access to information on mobile devices.
WAPI	Workflow Application Programming Interface.
Web	See WWW.
WfMC	Workflow Management Coalition.
WfMS	Workflow Management Systems.
WML	(Wireless Markup Language) XML based language which is used to specify content and user interface in WAP devices.
WWW	(World Wide Web) A system of Internet servers that support specially formatted documents.
X	
XML	(Extensible Markup Language) A specification which allow the addition of new tags, enabling the definition, transmission, validation and interpretation of between applications.
XPDL	XML Process definition language
Y	
Z	

References

- [1] W.M.P. van der Aalst, K.M. van Hee, and G.J. Houben, "Modelling workflow management systems with high-level Petri nets", In G. De Michelis, C. Ellis, and G. Memmi, editors, *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pages 31-50, 1994.
- [2] W.M.P. van der Aalst, "Petri-net-based Workflow Management Software", In A. Sheth, editor, *Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems*, pages 114-118, Athens, Georgia, May 1996.
- [3] Rob Allen, editor. *Workflow: An Introduction*. WfMC, 2001.
- [4] Sergio Bandinelli, Alfonso Fuggetta, Carlo Ghezzi, "Software Process Model Evolution in the SPADE Environment", GOODSTEP Technical Report No. 014, Dec. 1993.
- [5] Jakob E. Bardram, "Plans as Situated Action: An Activity Theory Approach to Workflow Systems", *Proceedings of the 1997 European conference on Computer Supported Cooperative Work (ECSCW'97)*, p. 17-32, 1997.
- [6] Martin Bauer, Christian Becker, Daniela Nicklas and Othmar Lehmann, "From Home to World - Supporting Context-aware Applications through World Models", *IEEE International Conference on Pervasive Computing and Communiactaions (PerCom'04)*, March 14-17, 2004.
- [7] J. Bowers, G. Button and W. Sharrock, "Workflow from within and without: Technology and Cooperative Work on the Print Industry Shopfloor", In *Proceedings of the Fourth European Conference on CSCW*, Stockholm, Sweden, Kluwer Academic Publishers, p. 51-66, 1995.
- [8] Victoria Bellotti and Keith Edwards, "Intelligibility and Accountability: Human Considerations in Context-Aware Systems", *Human-Computer Interaction (HCI) Journal. Special Issue: Context-Aware Computing*, 16(2-4):193-212, 2001.
- [9] G. Berreman, "Anemic and emetic analyses in social anthropology", *American Anthropologist*, 68(2)1:346-54, 1966.
- [10] G.A. Bolcer and R. N. Taylor: "Endeavors: A Process System Integration Infrastructure in *Proceedings of the Fourth International Conference on the Software Process*", Brighton, England, December 1996.

-
- [11] P. J. Brown and J. D. Bovey and X. Chen, "Context-Aware Applications: From the Laboratory to the Marketplace", IEEE Personal Communications, 4(5), pp. 58-64, 1997.
- [12] S. Ceri and R. Ramakrishnan, "Rules in database systems", ACM Comput. Surv. 28, 1, 109-111, 1996.
- [13] S. Ceri and J. Widom, "Deriving production rules for constraint maintenance", In Proceedings of the 16th International Conference on Very Large Data Bases, VLDB Endowment, Berkeley, CA, 566-577, 1990.
- [14] Fabio Casati, Stefano Ceri, Stefano Paraboschi and Guisepe Pozzi, "Specification and implementation of exceptions in workflow management systems", ACM Trans. Database Syst., volume 24, nr. 3, p 405--451, ACM Press, 1999.
- [15] Guanling Chen and David Kotz, "Context Aggregation and Dissemination in Ubiquitous Computing System", In Proceedings of the 4th Workshop on Mobile Computing Systems and Application (WMCSA 2002), pages 105-114, Callicoon, New York, June 2002.
- [16] J. Eder and W. Liebhart, "The workflow activity model WAMO", In Proceedings the International Conference on Cooperative Information Systems, Vienna, Austria, May 1995.
- [17] Anind K. Dey and Gregory D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Human- Computer Interaction (HCI) Journal. Special Issue: Context-Aware Computing, 16(2-4):97-166, 2001.
- [18] Anind K. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing Journal, 5(1):4-7, 2001.
- [19] C. Ellis, S. Gibbs, and G. Rein, "Groupware - Some Issues and Experiences," Commun. of the ACM, vol. 34, no. 1, p. 38-58, 1991.
- [20] Alois Ferscha, Simon Volg, Wolfgang Beer, "Ubiquitous context Sensing in Wireless Environments", 4th Austrian-Hungarian Workshops on Distributed and Parallel Systems (DAPSYS), ISBN 1-4020-7209-0, Kluwer Academic Publishers, 2002.
- [21] Adrian Fitzpatrick, Gregory Biegel, Siobh'an Clarke, Vinny Cahill, "Towards a Sentient Object Model", Position Paper Workshop on Engineering Context-Aware Object Oriented Systems and Environment (ECOOSE), Seattle WA, USA, Nov. 2002.
- [22] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley Professional, ISBN 0201633612, 1994.

-
- [23] C. Ghezzi, D. Mandrioli, S. Morasca and M. Pezze, "A Unified High-level Petri Net Formalism for Time-critical Systems", *IEEE Transactions on Software Engineering*, Feb. 1991.
- [24] J.C. Grundy and J.G. Hosking: "Serendipity: integrated environment support for process modelling, enactment and work coordination", *Automated Software Engineering: Special Issue on Process Technology 5(1)*, Kluwer Academic Publishers, p. 27-60, Jan. 1998.
- [25] C. Heath and P. Luff, "Documents and Professional Practice: 'bad' organisational reasons for 'good' clinical records", In *Proceedings of the Conference on CSCW*, Boston, Massachusetts, USA, ACM, p. 354-363, 1996.
- [26] Marc Herrmann, "Technical Report 01: C3DS Requirements Analysis", C3DS Public Technical Report Series, 42 pages, 1999.
- [27] Huadong Wu, Mel Siegel and Sevim Ablay, "Sensor Fusion for Context Understanding", *IEEE Instrumentation and Measurement Technology Conference*, Anchorage, AK, USA, 21-23, May 2002.
- [28] G. Hulin, M. Lacroix, D. Roelants and M. Vanhoedenaghe, "Integrated Product and Process Modelling", in *Proc. of First European Workshop on Software Process Modeling*, Milan, Italy, May 1991.
- [29] G. Iacucci and K. Kuutti, "Everyday Life as a Stage in Creating and Performing Scenarios for Wireless Devices", *Personal and Ubiquitous Computing* (6:4), p. 299 – 306, 2002.
- [30] Kerry Jean, Kun Yang, Alex Galis, "A Policy Based Context-aware Service for Next Generation Networks", *8th London Communication Symposium*, 8-10. Sept, 2003.
- [31] C. Kidd, R. Orr, G. Abowd, C. Atkeson, I. Essa, B. MacIntyre, E. Mynatt, T. Starner, W. Newstetter, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research", *Proc. of 2nd International Workshop on Cooperative Buildings - CoBuild*, 1999.
- [32] David M. Kroenke, "Database Processing: Fundamentals, Design & Implementation" 8th edition, Prentice Hall, 2002.
- [33] Yang Li, Jason I. Hong and James A. Landay, "ContextMap: Modeling Scenes of the Real World for Context-Aware Computing", *5th International Conference on Ubiquitous Computing*, Seattle, Washington, 12-15 Oct 2003.
- [34] C. Liu and R. Conradi, "Process Modeling Paradigms: an evaluation", in *Proc. Of First European Workshop on Software Process Modeling*, Milan, Italy, May 1991.
- [35] MARINTEK, SINTEF, "Operational Experience Review, Optimal Operation and Control of Offshore Installations" (not published)

-
- [36] J. McCarthy, "Notes on formalizing context", Proc. of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), 1993.
- [37] T. Murata, "Petri nets: Properties, analysis and applications", Proceedings of the IEEE, vol. 77, pp. 541-580, April 1989.
- [38] M.N. Nguyen, A.I. Wang and R. Conradi, "Total Software Process Model Evolution In EPOS", Submitted paper for 4th ICSP, Brighton, UK, 1999.
- [39] Nortel, supported by the University of Newcastle upon Tyne, "OMG document bom/98-03-09," Nortel's Responce to "Workflow Scenario: Airplane Design Process, 1998.
- [40] Nortel, supported by the University of Newcastle upon Tyne, "OMG document bom/98-03-10," Nortel's Responce to "Workflow Scenario: Trouble Ticket", 1998.
- [41] Jon Ole Nødtvedt, "Mobility and context-awareness in workflow systems", <http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2003/fordypning2003-Jon-Ole-Noedtvedt.pdf>, 2003.
- [42] Santanu Paul, Edwin Park, and Jarir Chaar, "RainMan: A workflow system for the Internet", In USENIX, editor, USENIX Symposium on Internet Technologies and Systems Proceedings, Monterey, California, December 8-11, 1997.
- [43] James Lyle Peterson, "Petri Net Theory and the Modeling of Systems", ISBN 0136619835, Prentice Hall PTR, 1981.
- [44] W. Reisig, "Petri Nets. An Introduction", EATCS Monographs on Theoretical Computer Science, 4, 1985.
- [45] Nick Ryan, Jason Pascoe and David Morse, "Enhanced reality fieldwork: the context-aware archaeological assistant", Computer Application in Archaeology, (CAA97), Digest of Papers, British Archaeological Report Series, Archaeopress, Oxford, UK, 1997.
- [46] M. Satyanarayanan, "Of Smart Dust and Brilliant Rocks", IEEE Pervasive Computing, Vol. 2, no. 4, Oct.-Dec. 2003.
- [47] Bill Schilit and M. Theimer, "Disseminating Active Map Information to Mobile Hosts", IEEE Networks, 8(5), pp. 22-32, 1994.
- [48] Albrecht Schmidt, Michael Beigl and Hans-W. Gellersen, "There is more to Context than Location", Proceedings of the International Workshop on Interactive Applications of Mobile Computing (IMC98), Rostock, Germany, Nov. 1998.

-
- [49] Kjeld Schmidt and Liam Bannon, "Taking CSCW Seriously: Supporting Articulation Work", *Computer Supported Cooperative Work (CSCW)*, An International Journal, Vol. 1, Nos. 1-2, p.7-40, 1992.
- [50] M. Sloman, "Policy Driven Management For Distributed Systems", *Journal of Network and System Management*, vol.2, no. 4, pp. 333-60, Dec. 1994.
- [51] A. Spriestersbach, H. Vogler, F. Lehmann, T. Ziegert, "Integrating Context Information into Enterprise Applications for the Mobile Workforce - Case Study", *Mobile Commerce Workshop MOBICOM*, 2001.
- [52] Martin Strohbach, Hans Gellersen, Gerd Kortuem, Christian Kray, "Intelligent Artefacts: An Embedded Systems Approach for Cooperative Assessment of Situations in the World", To appear in *Proceedings Ubicomp*, 2004.
- [53] Lucy A. Suchman, "Do categories have politics? The language/action perspective reconsidered", *CSCW 2 (3)*, p. 177-190, 1994.
- [54] Lucy A. Suchman, "Plans and situated actions: the problem of human-machine Communication", ISBN: 0-521-33137-4, Cambridge University Press, 1987.
- [55] Manos Theodorakis, Anastasia Analyti, Panos Constantopoulos and Nikos Spyrtatos, "Context in information bases", *Proc. of the 3rd International Conference on Cooperative Information Systems (IFCIS)*, New York, USA, pp. 260-270, August, 1998.
- [56] L. S. Vygotskij, "Mind and Society", Cambridge, MA: Harvard University Press, 1978.
- [57] Alf Inge Wang, Carl-Fredrik Sørensen, Eldrid Schei and Thale Christina Fritzner, "Case study: Use of the Mobile Tool Handyman for Mobile Work", In *Proceedings of the IASTED International Conference on Software Engineering and Applications (SEA 2003)*, Marina Del Rey, USA, Nov. 03-05, 2003.
- [58] Mark Weiser. *The Computer for the 21st Century*. *IEEE Pervasive Computing*, 1(1):18–25, January-March 2002. Reprinted from *Scientific American*, 1991.
- [59] T. Winograd, "Categories, diciplines and social coordination", *CSCW 2 (3)*, p. 177-190, 1994.
- [60] The Workflow Management Coalition, *Workflow Client Application (Interface 2), Application Programming Interface (WAPI) Specification*, Document Number WFMC-TC-1009, Oct. 1997.
- [61] The Workflow Management Coalition, "Workflow Management Coalition – Terminology & Glossary", Technical Report WFMC-TC-1011, The Workflow Management Coalition (WfMC), Document Number WFMC-TC-1011, Feb. 1999.

-
- [62] The Workflow Management Coalition, “The Workflow Reference Model”, Technical Report WFMC-TC-1003, The Workflow Management Coalition (WfMC), Document Number WFMC-TC-1003, Jan. 19th 1995.
- [63] The Workflow Management Coalition, “Workflow Process Definition Interface – XML Process Definition Language”, Technical Report WFMC-TC-1025, The Workflow Management Coalition (WfMC), Document Number WFMC-TC-1025, Oct. 25th 2002.
- [64] Martin V. Zelkowitz and Dolores R. Wallace, "Computing Practices: Experimental Models for Validating Technology", IEEE, May 1998.

Web references

- [65] About.com, <http://3dgraphics.about.com/library/glossary/bldefAR.htm>, 2004.
- [66] About.com, What is E-Commerce?, <http://ecommerce.about.com/cs/faqstutorials/a/aa021502.htm>, 2004.
- [67] Scott W. Ambler, UML Activity Diagramming Guidelines, <http://www.agilemodeling.com/style/stateChartDiagram.htm>, 2004.
- [68] Scott W. Ambler, UML State Chart Diagramming Guidelines, <http://www.agilemodeling.com/style/stateChartDiagram.htm>, 2004.
- [69] Association for Computing Machinery (ACM), <http://portal.acm.org/>, 2004.
- [70] Carnegie Mellon Software Engineering Institute, Software Process Automation, <http://www.sei.cmu.edu/legacy/procauto/projects.products.html>, 2004.
- [71] Cite Seer, <http://citeseer.nj.nec.com>, 2004.
- [72] Dralasoftware, “An example of an order processing workflow”, <http://www.dralasoftware.com/products/workflow/engine/details.html>, 2004.
- [73] Elsevier Computing Science, <http://www.elseviercomputerscience.com/>, 2004.
- [74] IBM, FlowMark, <http://www.software.ibm.com/ad/flowmark>, 2004.
- [75] IETF, Policy Framework Working Group web page, <http://www.ietf.org/html.charters/policy-charter.html>, 2004.
- [76] IETF, Internet Protocol, <http://www.ietf.org/rfc/rfc0791.txt>, 1981.

-
- [77] IETF, Transmission Control Protocol, <http://www.faqs.org/rfcs/rfc793.html>, 1981.
- [78] Loosely Coupled, Publish-Subscribe, <http://www.looselycoupled.com/glossary/publish-subscribe>, 2004.
- [79] Microsoft, Microsoft .NET, <http://www.microsoft.com/net/basics/>, 2004.
- [80] Microsoft, Distributed Component Object Model (DCOM), <http://www.microsoft.com/com/tech/dcom.asp>, 2004.
- [81] MOWAHS. Mobile Work Across Heterogeneous Systems, <http://www.mowahs.com>, 2001.
- [82] Object Management Group, OMG's Internet Inter-ORB Protocol (IIOP), <http://www.omg.org/library/iiop4.html>, 2004.
- [83] Object Management Group, Common Object Request Broker Architecture (CORBA), <http://www.omg.org/gettingstarted/corbafaq.htm>, 2004.
- [84] Science Direct, <http://www.sciencedirect.com>, 2004.
- [85] Soapuser.com, What is Simple Object Access Protocol (SOAP)?, <http://www.soapuser.com/basics1.html>, 2004.
- [86] Software QA and Testing Resource Center, <http://www.softwareqatest.com/qatfaq1.html>, 2004.
- [87] Staffware, <http://www.staffware.com>, 2004.
- [88] Sun Microsystems, Java Remote Method Invocation (RMI), <http://www.sun.com/products/jdk/1.2/docs/guide/rmi/>, 1999.
- [89] Sun Microsystems, Java Technology, <http://www.sun.com/software/learnabout/java/>, 2004.
- [90] TeamWare, <http://www.teamware.com>, 2004.
- [91] The Apache XML project, Xerces2, <http://xml.apache.org/xerces2-j/index.html>, 2004.
- [92] The University of Missouri- Rolla EMC Consortium, Backward chaining, <http://www.emclab.umr.edu/consortium/Whatis/node19.html>, 2004.
- [93] The University of Missouri- Rolla EMC Consortium, Forward chaining, <http://www.emclab.umr.edu/consortium/Whatis/node18.html>, 2004.

- [94] The University of Missouri- Rolla EMC Consortium, Tree searches, <http://www.emclab.umr.edu/consortium/Whatis/node20.html>, 2004.
- [95] The World Wide Web Consortium, Document Object Model, <http://www.w3.org/DOM/>, 2004.
- [96] W3C, Extensible Markup Language (XML), <http://www.w3c.org/XML/>, 2004.
- [97] W3C, Hypertext Transfer Protocol (HTTP), <http://www.w3.org/Protocols/>, 2004.
- [98] Webopedia.com, What is Local Area Network (LAN)?, http://www.webopedia.com/TERM/L/local_area_network_LAN.html, 2004.
- [99] Webopedia.com, What is Wide Area Network (WAN)?, http://www.webopedia.com/TERM/W/wide_area_network_WAN.html, 2004.
- [100] Whatis.com, Fuzzy Logic, http://whatis.techtarget.com/definition/0,,sid9_gci212172,00.html, 2004.
- [101] Workflow Management Coalition
<http://www.wfmc.org/>
- [102] World Wide Web Consortium, "Resource Description Framework (RDF)", <http://www.w3c.org/RDF>, 2004.
- [103] World Wide Web Consortium, XML Schema, <http://www.w3c.org/XML/Schema>, 2004.

Appendix A: Vertical Prototyping

This chapter provides a description of the technique for vertical prototyping. In addition, a statement on why we used this technique in our development and other related techniques for prototyping will also be covered in this chapter.

Vertical prototype is an approach that can be used to demonstrate the exact functionality of a product, but for only small section of the entire product. For example, a vertical prototype of a word processor might demonstrate all of the spell-checking functions, but none of the formatting or text-entry functions. All of the functions in a vertical prototype mimic their real counterparts as much as possible.

This approach should be used when the design for a particular section is rather complete and merits testing as a complete unit. Since a vertical prototype needs to be practically fully functional (although just for a small portion of the product interface), the best way to obtain a vertical prototype is to use a fully functioning module of the product. For software programs that are written with a modular architecture, this can usually be done, although the interface to other modules won't work. For a car, it could be the seating and other interior furnishings that will be tested, while the drive train, body sensors, and other components which are not ready yet.

A complete workflow system is comprised by a complex structure, in terms of their functionalities and their relationship between different components. Furthermore, such systems can also be in operation although the fully functionalities are not implemented. For instance you can make a payment transaction in a net bank, but since the system are not fully implemented according to their requirements of the system, you will not be able to make a stock trading.

Because of the limited project period, we were not able to implement a complete workflow system. Our choice of this type of prototyping was also based on that this technique allowed us to cover more features and functions of a context-aware system in the given amount of time, than building an entire system.

There are a number of different terms we can hear in conjunction with prototyping methods. The following is a listing of some of these techniques:

Technique:	Purpose	Pros	Cons
Rapid Prototyping	Is a technique that quickly develops new designs	New design is quickly developed as the design cycle progresses.	Expensive, in terms of time and money.
Reusable Prototyping	Is a technique that makes use of parts (or all) of the prototype in the actual product.	Existing parts can be use in a product.	Expensive, in terms of time and money.
Modular Prototyping	Concerns with	Able to reconfiguration of	Expensive, in terms

	different modules of a product.	each module. New parts can be added into the product/ system.	of time and money.
Horizontal Prototyping	Covers a large breadth of features and functions, but most aren't working.	Best for testing breadth of scope but not actual use.	Does not provide the extensive functionality behind each function.
Vertical Prototyping	Covers a large breadth of features and functions.	Best for testing usage in a small portion of the product.	Does not cover the complete functionalities of the product.
Low-fidelity Prototyping	Is a technique that uses paper and pencil to mock-up interface screens	Not expensive to use, in terms of time and money. Provides lot of feedback about the interaction between the interface and the user.	Mimics the function of the actual product.
High-fidelity Prototyping	Concerns with the actual design/ interface of a product.	Not expensive to use, in terms of time and money. This technique is useful when the actual interface of a product is not finished yet.	Does not show the fully functionality of a product.

Table 1: An overview of prototyping techniques

As Table 1 describes, low-fidelity and high-fidelity prototyping are concerned with the design and interface of the actual product, which is not our focus for the development of the workflow prototypes. Our prototypes will just cover a small section of a whole system, which fits the technique for vertical prototyping.

Appendix B: UML State Chart Diagram

A state chart diagram shows the behaviour of classes in response to external stimuli. This diagram models the dynamic flow of control from state to state within a system.

Table 2 presents basic symbols and notations of the state chart diagram:




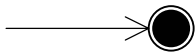
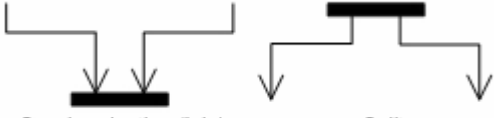
	<p>States States represent situations during the life of an object.</p>
<p>event [condition/action]</p> 	<p>Transition A solid represents the path between different states of an object. Label the transition with the event that triggered it, the condition which is satisfied and the action that results from it.</p>
	<p>Initial State A filled circle followed by an arrow represents the object's initial state.</p>
	<p>Final State An arrow pointing to a filled circle nested inside another circle represents the object's final state.</p>
 <p>Synchronisation (join) Split</p>	<p>Synchronization and Splitting of Control A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.</p>

Table 2: Basic state chart diagram symbols and notations.

Appendix C: UML Activity diagram

An activity diagram illustrates the flow of control from activity to activity, where transitions from activity to activity are done in response to internal stimuli such as activity completion.

Table 3 illustrates the key symbols and notations used in activity diagrams:

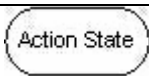
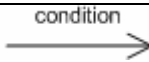

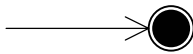
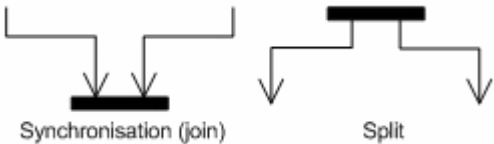
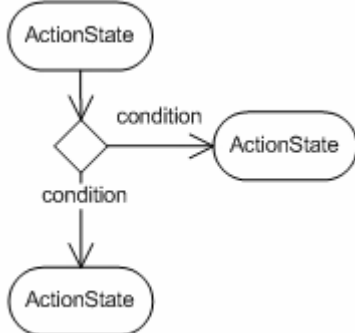
	<p>Action states States represent actions to be performed during the life of an object.</p>
	<p>Control flow A solid represents the path between different action states of an object. They can be labelled with the condition, which is satisfied.</p>
	<p>Initial State A filled circle followed by an arrow represents the object's initial state.</p>
	<p>Final State An arrow pointing to a filled circle nested inside another circle represents the object's final state.</p>
	<p>Synchronization and Splitting of Control A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.</p>
	<p>Branching A diamond represents a decision with alternate paths. The outgoing paths should be labelled with the conditions, which are satisfied.</p>

Table 3: Basic activity diagram symbols and notations.

Appendix D: Test report

Testing of software means that the software product is operated under controlled conditions. The results from this test are compared against the expected results to evaluate the results of the test.

During the development process of a software product, the product goes through verification and validation. Verification means that plans, requirements and specifications are reviewed by the customer. Validation involves the actual testing of the product and is done after verification phase.

It is possible to divide software testing into static and dynamic testing. Static testing means testing without running the software product. Code review is one of the most common strategies for static testing. Dynamic testing involves testing while running part of or the whole software product. It is possible to separate dynamic testing into several categories. Some of the dynamic testing categories are illustrated in Table 4 [86].

Test type	Description
Back box test	Testing without any knowledge of internal design.
Unit test	Testing of individual functions or modules. Usually done by the software developers themselves.
Integration test	Testing of combined parts of an application to determine if they function correctly together.
System test	Black box testing based on the overall requirements specification.
Acceptance test	Final testing based on the specifications of the end user/customer.

Table 4: Software test types

We have implemented vertical prototypes in our project. The prototypes function as a demonstration tools and proof-of-concepts. The requirements for the prototypes were prompted by our research questions and our study of the state-of-the-art. This means that the verification part of the development process is irrelevant for our prototypes. Since the prototypes mainly function as demonstration tools of the dynamic behaviour of a context-aware workflow system, we chose to limit our validation to limited system tests. The limited system test allows us to test the functionality of the system according to our requirements. We did not want to do a complete system test, since our prototypes are supposed to be used as demonstration tools only and require minimal input from the user. A fully implemented context-aware workflow system requires a more comprehensive system test than what we have performed. Additional tests may also be required.

Our previously stated requirements in Chapter 7 constitute high-level requirements and are not directly testable. We have therefore specified functionally testable requirements for each prototype. These requirements are stated in the following tables for our system tests. The system tests completed are stated for each prototype. The scenario for each system test is specified as part of the individual prototype specification. For the system test, the individual processes of the prototypes are considered as a single entity.

The prototypes consist of several command line applications that have to be started up in the correct order. This is handled by a start-up script. The behaviour of the prototype is shown by the textual output in the command window and by updates to the process description file for one prototype. The execution of each prototype is described in the attached CD-ROM.

The system test for the prototype specified in Chapter 8.2 where context information was used in workflow transitions is presented in Table 5

Test number	Requirement	Test result
1	The prototype type starts up correctly without error messages using the start-up script.	OK
2	The prototype indicates where activities have been received.	OK
3	The prototype indicates that workflow relevant data has been updated.	OK
4	The prototype operates according to the process definition.	OK

Table 5: System test for the prototype in Chapter 8.2

The system test for the prototype specified in Chapter 8.3 where workflow actions are initiated based on context changes is presented in Table 6.

Test number	Requirement	Test result
1	The prototype type starts up correctly without error messages using the start-up script.	OK
2	The prototype indicates where activities have been received.	OK
3	The prototype indicates that workflow relevant data has been updated.	OK
4	The prototype operates according to the process definition.	OK
5	A second process is started based on a context event	OK

Table 6: System test for the prototype in Chapter 8.3

The system test for the prototype specified in Chapter 8.4 where an exception handler is used to handle context exception states is presented in Table 7.

Test number	Requirement	Test result
1	The prototype type starts up correctly without error messages using the start-up script.	OK
2	The prototype indicates where activities have been received.	OK
3	The prototype indicates that workflow relevant data has been updated.	OK
4	The prototype operates according to the process definition.	OK
5	The prototype operates according to the exception handler definition file.	OK
6	The prototype generates a new process definition file according to the original process definition and the exception handler file	OK

Table 7: System test for the prototype in Chapter 8.4

The system test for the prototype specified in Chapter 8.5 where the exception handler is used to re-evaluate the process path is presented in Table 8.

Test number	Requirement	Test result
1	The prototype type starts up correctly without error messages using the start-up script.	OK
2	The prototype indicates where activities have been received.	OK
3	The prototype indicates that workflow relevant data has been updated.	OK
4	The prototype operates according to the process definition.	OK
5	The process path is re-evaluated according to the scenario definition.	OK

Table 8: System test for the prototype in Chapter 8.5

The system test for the prototype specified in Chapter 8.6 where an invariant is invalidated by the user is presented in Table 9.

Test number	Requirement	Test result
1	The prototype type starts up correctly without error messages using the start-up script.	OK
2	The prototype indicates where activities have been received.	OK

3	The prototype indicates that workflow relevant data has been updated.	OK
4	The prototype operates according to the process definition.	OK
5	The BooleanWidget with id "BOOLEAN1" accepts user input.	OK
6	The process is rolled back if the user writes "FALSE" in the BooleanWidget with id "BOOLEAN1" when the process is running.	OK

Table 9: System test for the prototype in Chapter 8.6

The system test for the prototype specified in Chapter 8.7.2 where a client based inference engine is used to build a local process for the client is presented in Table 10.

Test number	Requirement	Test result
1	The prototype type starts up correctly without error messages using the start-up script.	OK
2	The prototype indicates where activities have been received.	OK
3	The prototype indicates that workflow relevant data has been updated.	OK
4	The prototype operates according to the process definition.	OK
5	The workflow client indicates that local process activities has been received and returned.	OK
6	The local process activities enacted is based on the contextual state and the client knowledge base	OK

Table 10: System test for the prototype in Chapter 8.7.2

The system test for the prototype specified in Chapter 8.7.3 where a context source forms the post-condition for the enactment of an activity is presented in Table 11.

Test number	Requirement	Test result
1	The prototype type starts up correctly without error messages using the start-up script.	OK
2	The prototype indicates where activities have been received.	OK
3	The prototype indicates that workflow relevant data has been updated.	OK
4	The prototype operates according to the process definition.	OK
5	The BooleanWidget with id	OK

	“BOOLEAN1” accepts user input	
6	The executing activity is returned as soon as the user type “TRUE” in the command window for the BooleanWidget with id “BOOLEAN1”.	OK

Table 11: System test for the prototype in Chapter 8.7.3

Appendix E: Context information used in workflow transitions process definition

```

<?xml version="1.0" encoding="us-ascii"?>
<Package xmlns="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xyz="http://www.xyzeorder.com/workflow"
xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0
http://wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd" Id="0" Name="test context query
workflow process">
  <PackageHeader>
    <XPDLVersion>0.09</XPDLVersion>
    <Vendor>Man og Jon Ole</Vendor>
    <Created>1/21/2004 5:27:17 PM</Created>
  </PackageHeader>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <Script Type="text/javascript"/>
  <TypeDeclarations/>
  <Participants/>
  <Applications/>
  <DataFields/>
  <WorkflowProcesses>
    <WorkflowProcess Id="1" Name="ContextValueQuery" AccessLevel="PUBLIC">
      <ProcessHeader/>
      <FormalParameters/>
      <DataFields>
        <DataField Id="contextvalue1" IsArray="FALSE">
          <DataType>
            <BasicType Type="BOOLEAN"/>
          </DataType>
          <InitialValue>TRUE</InitialValue>
          <Length>0</Length>
        </DataField>
      </DataFields>
      <Participants>
        <Participant Id="CII">
          <ParticipantType Type="SYSTEM"/>
          <Description>Reference to Context Information Framework</Description>
        </Participant>
        <Participant Id="P1">
          <ParticipantType Type="HUMAN"/>
          <Description>Human client</Description>
        </Participant>
        <Participant Id="P2">
          <ParticipantType Type="HUMAN"/>
          <Description>Human client</Description>
        </Participant>
      </Participants>
      <Applications>
        <Application Id="pollContextSource">

```

```

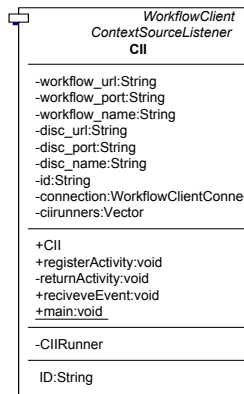
<FormalParameters>
  <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
    <DataType>
      <BasicType Type="STRING"/>
    </DataType>
  </FormalParameter>
  <FormalParameter Id="contextvalue1" Index="2" Mode="OUT">
    <DataType>
      <BasicType Type="BOOLEAN"/>
    </DataType>
  </FormalParameter>
</FormalParameters>
</Application>
</Applications>
<Activities>
  <Activity Id="1" Name="Activity1">
    <Implementation>
      <Tool Id="pollContextSource" Type="APPLICATION">
        <ActualParameters>
          <ActualParameter>BOOLEAN</ActualParameter>
          <ActualParameter>
            </ActualParameter>
        </ActualParameters>
      </Tool>
    </Implementation>
    <Performer>CII</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Split Type="XOR">
          <TransitionRefs>
            <TransitionRef Id="1"/>
            <TransitionRef Id="2"/>
          </TransitionRefs>
        </Split>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="2" Name="Activity2">
    <Route/>
    <Performer>P1</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="XOR"/>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="3" Name="Activity3">
    <Route/>
    <Performer>P2</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="XOR"/>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes/>
  </Activity>
</Activities>
<Transitions>

```

```
<Transition Id="1" From="1" To="2">
  <Condition>contextvalue1 == "TRUE"</Condition>
</Transition>
<Transition Id="2" From="1" To="3">
  <Condition>contextvalue1 == "FALSE"</Condition>
</Transition>
</Transitions>
</WorkflowProcess>
</WorkflowProcesses>
</Package>
```

Appendix F: Class diagrams for all packages in the initial prototype

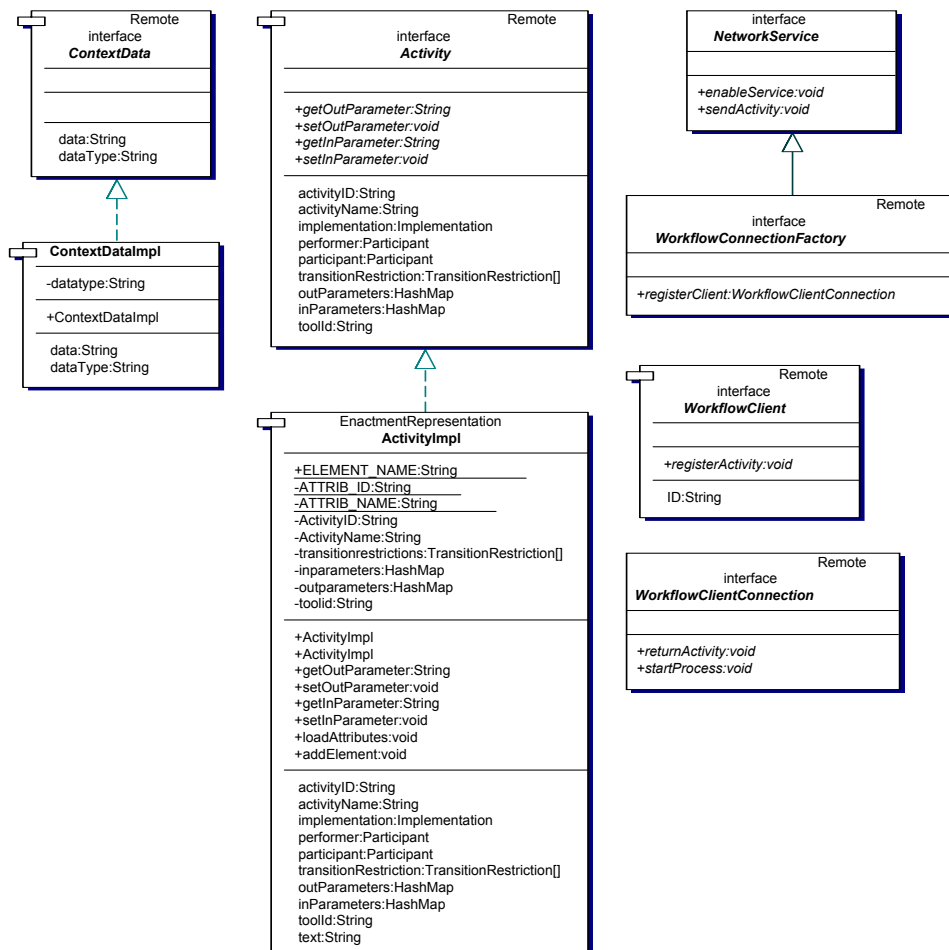
The cii package:



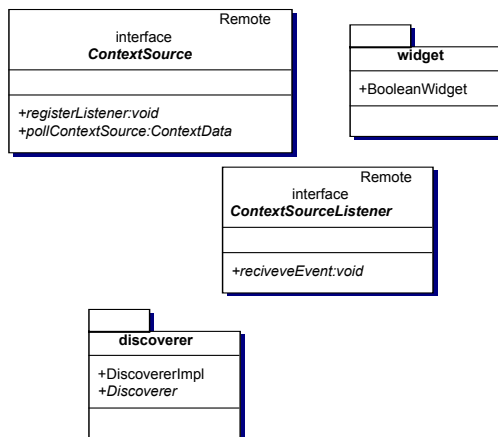
This package consists of the “CII” class. It implements the functionality of the CII component.

The common package:

This package consists of several classes and interfaces used by several independent processes in our prototypes

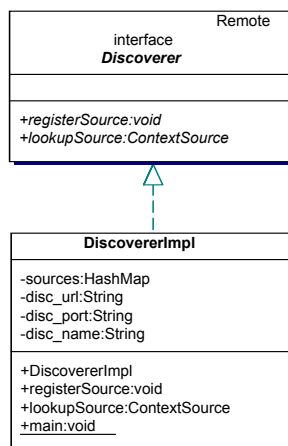


The contextsource package:



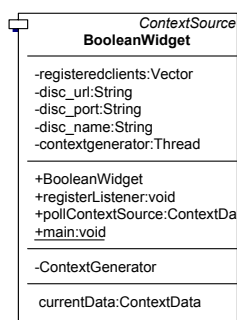
This package contains the interfaces of the context framework components. The sub-packages for the discoverer and BooleanWidget are located in this package.

The contextsource.discoverer package:



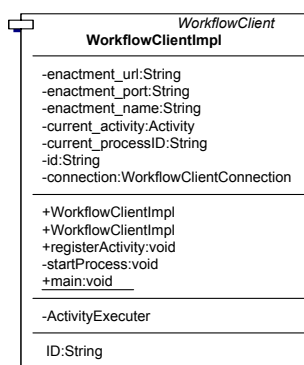
This package contains the interface and class for the discoverer component.

The contextsource.widget package:



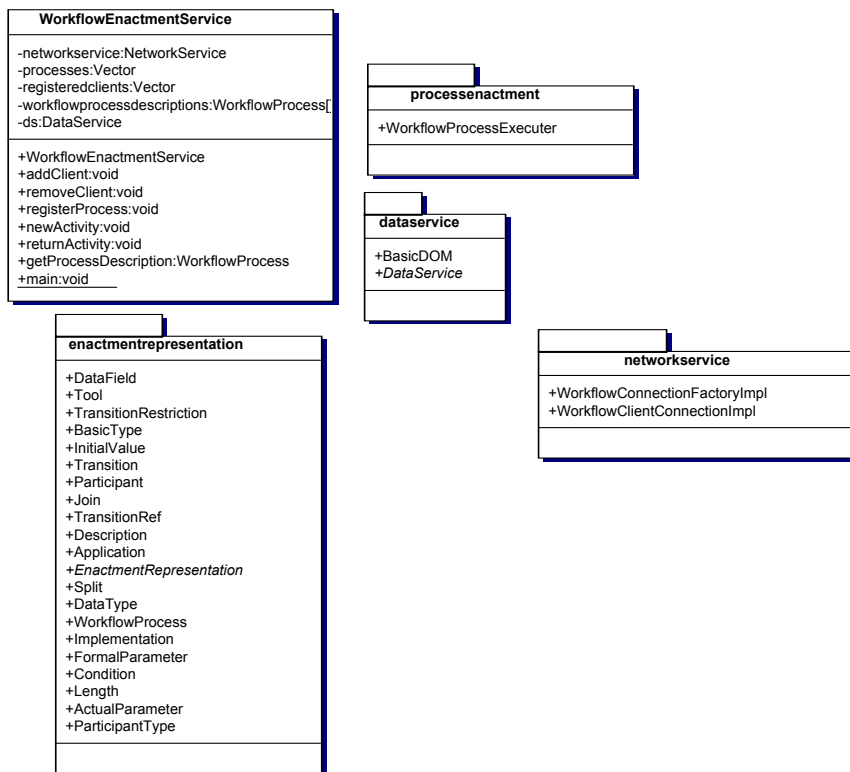
This package contains the implementation of a BooleanWidget context source.

The workflowclient package:



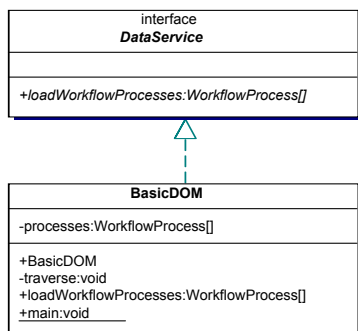
This package contains the implementation of a workflow client.

The workflowenactment package:



This package contains the implementation of the workflow enactment service. In addition the sub-packages of enactmentrepresentation, processenactment, dataservice and networkservice are also located within this package.

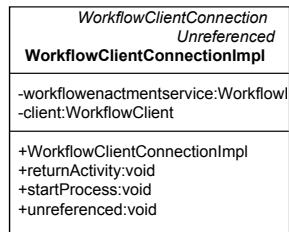
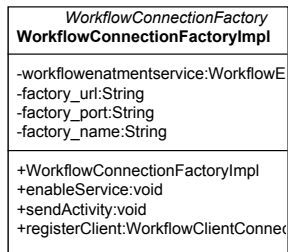
The workflowenactment.dataservice package:



This package contains the interface and implementation class of the “DataService” layer.

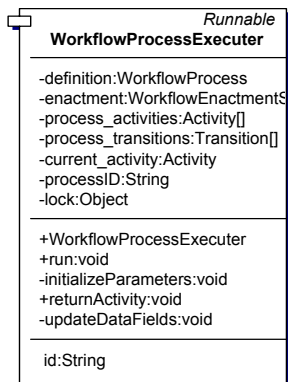
The `workflowenactment.enactmentrepresentation` contains all the classes necessary to represent the process definition within the workflow enactment service at run-time. The “EnactmentRepresentation” class is the base class for all other representation classes.

The `workflowenactment.networkservice` package:



This class contain the implementation classes of the “NetworkService” layer of the workflow enactment service.

The `workflowenactment.processenactment` package:



This package contains the implementation class, which is responsible for process enactment.

Appendix G: Workflow actions based on context changes process definition

```

<?xml version="1.0" encoding="us-ascii"?>
<Package xmlns="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xyz="http://www.xyzeorder.com/workflow"
xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0
http://wfmc.org/standards/docs/TC-1025_schema_10_xpdl.xsd" Id="0" Name="test context query
workflow process">
  <PackageHeader>
    <XPDLVersion>0.09</XPDLVersion>
    <Vendor>Man og Jon Ole</Vendor>
    <Created>3/04/2004 5:27:17 PM</Created>
  </PackageHeader>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <Script Type="text/javascript"/>
  <TypeDeclarations/>
  <Participants/>
  <Applications/>
  <DataFields/>
  <WorkflowProcesses>
    <WorkflowProcess Id="1" Name="ContextSourceSubscription" AccessLevel="PUBLIC">
      <ProcessHeader/>
      <FormalParameters/>
      <DataFields>
      </DataFields>
      <Participants>
        <Participant Id="CII">
          <ParticipantType Type="SYSTEM"/>
          <Description>Reference to Context Information Framework</Description>
        </Participant>
      </Participants>
      <Applications>
        <Application Id="subscribeContextSource">
          <FormalParameters>
            <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
              <DataType>
                <BasicType Type="STRING"/>
              </DataType>
            </FormalParameter>
            <FormalParameter Id="contextcondition" Index="2" Mode="IN">
              <DataType>
                <BasicType Type="BOOLEAN"/>
              </DataType>
            </FormalParameter>
            <FormalParameter Id="operation" Index="3" Mode="IN">
              <DataType>
                <BasicType Type="STRING"/>
              </DataType>
            </FormalParameter>
          </FormalParameters>
        </Application>
      </Applications>
    </WorkflowProcess>
  </WorkflowProcesses>
</Package>

```

```

    </FormalParameter>
  </FormalParameters>
</Application>
</Applications>
<Activities>
  <Activity Id="1" Name="Activity1">
    <Implementation>
      <Tool Id="subscribeContextSource" Type="APPLICATION">
        <ActualParameters>
          <ActualParameter>BOOLEAN</ActualParameter>
          <ActualParameter>TRUE</ActualParameter>
          <ActualParameter>startProcess(2)</ActualParameter>
        </ActualParameters>
      </Tool>
    </Implementation>
    <Performer>CII</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Split Type="XOR">
          <TransitionRefs>
            </TransitionRefs>
          </Split>
        </TransitionRestriction>
      </TransitionRestrictions>
      <ExtendedAttributes/>
    </Activity>
  </Activities>
</Transitions>
</Transitions>
</WorkflowProcess>
<WorkflowProcess Id="2" Name="Adhocprocess" AccessLevel="PUBLIC">
  <ProcessHeader/>
  <FormalParameters/>
  <DataFields>
  </DataFields>
  <Participants>
    <Participant Id="P1">
      <ParticipantType Type="HUMAN"/>
      <Description>Human client</Description>
    </Participant>
  </Participants>
  <Applications>
  </Applications>
  <Activities>
    <Activity Id="1" Name="Activity1">
      <Performer>P1</Performer>
      <TransitionRestrictions>
        <TransitionRestriction>
          <Split Type="XOR">
            <TransitionRefs>
              </TransitionRefs>
            </Split>
          </TransitionRestriction>
        </TransitionRestrictions>
        <ExtendedAttributes/>
      </Activity>
    </Activities>
  </Transitions>
</Transitions>
</WorkflowProcess>

```

```
</WorkflowProcesses>  
</Package>
```

Appendix H: Revised process for exception condition

```

<?xml version="1.0" encoding="UTF-8"?>
<Package Id="0" Name="test context query workflow process"
  xmlns="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xyz="http://www.xyzeorder.com/workflow"
  xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-
1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>0.09</XPDLVersion>
    <Vendor>Man og Jon Ole</Vendor>
    <Created>1/21/2004 5:27:17 PM</Created>
  </PackageHeader>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <Script Type="text/javascript"/>
  <TypeDeclarations/>
  <Participants/>
  <Applications/>
  <DataFields/>
  <WorkflowProcesses>
    <WorkflowProcess AccessLevel="PUBLIC" Id="1" Name="ContextValueQuery">
      <ProcessHeader/>
      <FormalParameters/>
      <DataFields>
        <DataField Id="contextvalue1" IsArray="FALSE">
          <DataType>
            <BasicType Type="BOOLEAN"/>
          </DataType>
          <InitialValue>TRUE</InitialValue>
          <Length>0</Length>
        </DataField>
      </DataFields>
      <Participants>
        <Participant Id="CII">
          <ParticipantType Type="SYSTEM"/>
          <Description>Reference to Context Information Framework</Description>
        </Participant>
        <Participant Id="P1">
          <ParticipantType Type="HUMAN"/>
          <Description>Human client</Description>
        </Participant>
        <Participant Id="P2">
          <ParticipantType Type="HUMAN"/>
          <Description>Human client</Description>
        </Participant>
      </Participants>
      <Applications>
        <Application Id="pollContextSource">
          <FormalParameters>
            <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
              <DataType>

```



```

    <BasicType Type="STRING"/>
  </DataType>
</FormalParameter>
<FormalParameter Id="contextvalue1" Index="2" Mode="OUT">
  <DataType>
    <BasicType Type="BOOLEAN"/>
  </DataType>
</FormalParameter>
</FormalParameters>
</Application>
</Applications>
<Activities>
  <Activity Id="1" Name="Activity1">
    <Implementation>
      <Tool Id="pollContextSource" Type="APPLICATION">
        <ActualParameters>
          <ActualParameter>BOOLEAN</ActualParameter>
          <ActualParameter/>
        </ActualParameters>
      </Tool>
    </Implementation>
    <Performer>CII</Performer>
    <TransitionRestrictions/>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="2" Name="Activity2">
    <Route/>
    <Performer>P1</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="XOR"/>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="3" Name="Activity3">
    <Route/>
    <Performer>P2</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="XOR"/>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="4" Name="verifyState">
    <Description>Verify contextual state</Description>
    <Implementation>
      <Tool Id="pollContextSource" Type="0">
        <ActualParameter>BOOLEAN</ActualParameter>
        <ActualParameter>UNDEFINED</ActualParameter>
      </Tool>
    </Implementation>
    <Performer>P1</Performer>
    <TransitionRestrictions/>
  </Activity>
</Activities>
<Transitions>
  <Transition From="1" Id="1" To="2">
    <Condition>contextvalue1 == &quot;TRUE&quot;</Condition>

```

```
</Transition>
<Transition From="1" Id="2" To="3">
  <Condition>contextvalue1 == &quot;FALSE&quot;</Condition>
</Transition>
<Transition From="4" Id="3" To="2">
  <Condition>contextvalue1 == &quot;TRUE&quot;</Condition>
</Transition>
<Transition From="4" Id="4" To="3">
  <Condition>contextvalue1 == &quot;FALSE&quot;</Condition>
</Transition>
<Transition From="1" Id="5" To="4">
  <Condition>contextvalue1 == &quot;UNDEFINED&quot;</Condition>
</Transition>
</Transitions>
</WorkflowProcess>
</WorkflowProcesses>
</Package>
```

Appendix I: XML schema for exception handling rules

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Action">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Generate" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="RevalidateProcessPath" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="Id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Condition">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Value"/>
      </xs:sequence>
      <xs:attribute name="Id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Event">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Type"/>
      </xs:sequence>
      <xs:attribute name="Id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Generate">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Activity" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="Transition" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="Id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Activity">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Name"/>
        <xs:element ref="Description"/>
        <xs:element ref="Implementation" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="Performer" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="UseCurrent" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="Id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Transition">
    <xs:complexType>
      <xs:sequence>

```

```

    <xs:element ref="UseCurrent" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="ConditionValue" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="To" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="From" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="Id" type="xs:string"/>
</xs:complexType>
</xs:element>
<xs:element name="Type" type="xs:string"/>
<xs:element name="Value" type="xs:string"/>
<xs:element name="Description" type="xs:string"/>
<xs:element name="Performer" type="xs:string"/>
<xs:element name="Implementation" type="xs:string"/>
<xs:element name="UseCurrent" type="xs:string"/>
<xs:element name="ConditionValue" type="xs:string"/>
<xs:element name="Name" type="xs:string"/>
<xs:element name="To" type="xs:string"/>
<xs:element name="From" type="xs:string"/>
<xs:element name="RevalidateProcessPath" type="xs:string"/>
<xs:element name="Events">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Event" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Conditions">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Condition" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Actions">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Action" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Rule">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Events" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Conditions" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Actions" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="Id" type="xs:string"/>
  </xs:complexType>
</xs:element>
<xs:element name="Rules">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Rule" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ExceptionHandling">
  <xs:complexType>
    <xs:sequence>

```

```
    <xs:element ref="Rules" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Appendix J: XML exception handling document for context state exception condition

```

<?xml version="1.0" encoding="UTF-8"?>
<ExceptionHandling xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="exceptionschema.xsd">
  <Rules>
    <Rule Id="1">
      <Events>
        <Event Id="1">
          <Type>TransitionException</Type>
        </Event>
      </Events>
      <Conditions>
        <Condition Id="1">
          <Value>transition value == "UNDEFINED"</Value>
        </Condition>
      </Conditions>
      <Actions>
        <Action Id="1">
          <Generate Id="1">
            <Activity Id="1">
              <Name>verifyState</Name>
              <Description>Verify contextual state</Description>
              <Implementation>current</Implementation>
              <Performer>P1</Performer>
              <UseCurrent>FALSE</UseCurrent>
            </Activity>
            <Transition Id="1">
              <UseCurrent>TRUE</UseCurrent>
              <From>1</From>
            </Transition>
            <Transition Id="2">
              <ConditionValue>transition value == "UNDEFINED"</ConditionValue>
              <To>1</To>
              <From>current</From>
            </Transition>
          </Generate>
        </Action>
      </Actions>
    </Rule>
  </Rules>
</ExceptionHandling>

```

Appendix K: Process path revalidation process specification

```

<?xml version="1.0" encoding="UTF-8"?>
<Package Id="0" Name="test context query workflow process"
  xmlns="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xyz="http://www.xyzeorder.com/workflow"
  xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-
1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>0.09</XPDLVersion>
    <Vendor>Man og Jon Ole</Vendor>
    <Created>1/21/2004 5:27:17 PM</Created>
  </PackageHeader>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <Script Type="text/javascript"/>
  <TypeDeclarations/>
  <Participants/>
  <Applications/>
  <DataFields/>
  <WorkflowProcesses>
    <WorkflowProcess AccessLevel="PUBLIC" Id="1" Name="ContextValueQuery">
      <ProcessHeader/>
      <FormalParameters/>
      <DataFields>
        <DataField Id="contextvalue1" IsArray="FALSE">
          <DataType>
            <BasicType Type="BOOLEAN"/>
          </DataType>
          <InitialValue>TRUE</InitialValue>
          <Length>0</Length>
        </DataField>
        <DataField Id="contextvalue2" IsArray="FALSE">
          <DataType>
            <BasicType Type="BOOLEAN"/>
          </DataType>
          <InitialValue>TRUE</InitialValue>
          <Length>0</Length>
        </DataField>
      </DataFields>
      <Participants>
        <Participant Id="CII">
          <ParticipantType Type="SYSTEM"/>
          <Description>Reference to Context Information Framework</Description>
        </Participant>
        <Participant Id="P1">
          <ParticipantType Type="HUMAN"/>
          <Description>Human client</Description>
        </Participant>
      </Participants>
    </WorkflowProcess>
  </WorkflowProcesses>
</Package>

```

```

<Participant Id="P2">
  <ParticipantType Type="HUMAN"/>
  <Description>Human client</Description>
</Participant>
</Participants>
<Applications>
  <Application Id="pollContextSource1">
    <FormalParameters>
      <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
        <DataType>
          <BasicType Type="STRING"/>
        </DataType>
      </FormalParameter>
      <FormalParameter Id="contextvalue1" Index="2" Mode="OUT">
        <DataType>
          <BasicType Type="BOOLEAN"/>
        </DataType>
      </FormalParameter>
    </FormalParameters>
  </Application>
  <Application Id="pollContextSource2">
    <FormalParameters>
      <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
        <DataType>
          <BasicType Type="STRING"/>
        </DataType>
      </FormalParameter>
      <FormalParameter Id="contextvalue2" Index="2" Mode="OUT">
        <DataType>
          <BasicType Type="BOOLEAN"/>
        </DataType>
      </FormalParameter>
    </FormalParameters>
  </Application>
</Applications>
<Activities>
  <Activity Id="1" Name="Activity1">
    <Implementation>
      <Tool Id="pollContextSource1" Type="APPLICATION">
        <ActualParameters>
          <ActualParameter>BOOLEAN1</ActualParameter>
          <ActualParameter/>
        </ActualParameters>
      </Tool>
    </Implementation>
    <Performer>CII</Performer>
    <TransitionRestrictions/>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="2" Name="Activity2">
    <Route/>
    <Performer>P1</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="XOR"/>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="3" Name="Activity3">

```



```

<Route/>
<Performer>P2</Performer>
<TransitionRestrictions>
  <TransitionRestriction>
    <Join Type="XOR"/>
  </TransitionRestriction>
</TransitionRestrictions>
<ExtendedAttributes/>
</Activity>
<Activity Id="4" Name="Activity4">
  <Implementation>
    <Tool Id="pollContextSource2" Type="APPLICATION">
      <ActualParameters>
        <ActualParameter>BOOLEAN2</ActualParameter>
      </ActualParameters>
    </Tool>
  </Implementation>
  <Performer>CII</Performer>
  <TransitionRestrictions/>
  <ExtendedAttributes/>
</Activity>
<Activity Id="5" Name="Activity5">
  <Implementation>
    <Tool Id="pollContextSource2" Type="APPLICATION">
      <ActualParameters>
        <ActualParameter>BOOLEAN2</ActualParameter>
      </ActualParameters>
    </Tool>
  </Implementation>
  <Performer>CII</Performer>
  <TransitionRestrictions/>
  <ExtendedAttributes/>
</Activity>
<Activity Id="6" Name="Activity6">
  <Route/>
  <Performer>P2</Performer>
  <TransitionRestrictions>
    <TransitionRestriction>
      <Join Type="XOR"/>
    </TransitionRestriction>
  </TransitionRestrictions>
  <ExtendedAttributes/>
</Activity>
</Activities>
<Transitions>
  <Transition Id="1" From="1" To="2">
    <Condition>contextvalue1 == "TRUE"</Condition>
  </Transition>
  <Transition Id="2" From="1" To="3">
    <Condition>contextvalue1 == "FALSE"</Condition>
  </Transition>
  <Transition Id="3" From="2" To="4">
  </Transition>
  <Transition Id="4" From="3" To="5">
  </Transition>
  <Transition Id="5" From="4" To="6">
    <Condition>contextvalue2 == "TRUE"</Condition>
  </Transition>

```

```
<Transition Id="6" From="5" To="6">  
  <Condition>contextvalue2 == "FALSE"</Condition>  
</Transition>  
</Transitions>  
</WorkflowProcess>  
</WorkflowProcesses>  
</Package>
```

Appendix L: Process path revalidation exception handler rules

```

<?xml version="1.0" encoding="UTF-8"?>
<ExceptionHandling xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="exceptionschema.xsd">
  <Rules>
    <Rule Id="1">
      <Events>
        <Event Id="1">
          <Type>TransitionException</Type>
        </Event>
      </Events>
      <Conditions>
        <Condition Id="1">
          <Value>transition value == "UNDEFINED"</Value>
        </Condition>
      </Conditions>
      <Actions>
        <Action Id="1">
          <Generate Id="1">
            <Activity Id="1">
              <Name>verifyState</Name>
              <Description>Verify contextual state</Description>
              <Implementation>current</Implementation>
              <Performer>P1</Performer>
              <UseCurrent>FALSE</UseCurrent>
            </Activity>
            <Transition Id="1">
              <UseCurrent>TRUE</UseCurrent>
              <From>1</From>
            </Transition>
            <Transition Id="2">
              <ConditionValue>transition value == "UNDEFINED"</ConditionValue>
              <To>1</To>
              <From>current</From>
            </Transition>
          </Generate>
        </Action>
      </Actions>
    </Rule>
    <Rule Id="2">
      <Events>
        <Event Id="1">
          <Type>TransitionException</Type>
        </Event>
      </Events>
      <Conditions/>
      <Actions>
        <Action Id="1">
          <RevalidateProcessPath>TRUE</RevalidateProcessPath>
        </Action>
      </Actions>
    </Rule>
  </Rules>
</ExceptionHandling>

```

```
</Action>  
</Actions>  
</Rule>  
</Rules>  
</ExceptionHandler>
```

Appendix M: Process description for invariant scenario

```

<?xml version="1.0" encoding="UTF-8"?>
<Package Id="0" Name="test context query workflow process"
  xmlns="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xyz="http://www.xyzeorder.com/workflow"
  xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-
1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>0.09</XPDLVersion>
    <Vendor>Man og Jon Ole</Vendor>
    <Created>1/21/2004 5:27:17 PM</Created>
  </PackageHeader>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <Script Type="text/javascript"/>
  <TypeDeclarations/>
  <Participants/>
  <Applications/>
  <DataFields/>
  <WorkflowProcesses>
    <WorkflowProcess AccessLevel="PUBLIC" Id="1" Name="ContextValueQuery">
      <ProcessHeader/>
      <FormalParameters/>
      <DataFields>
        <DataField Id="contextvalue1" IsArray="FALSE">
          <DataType>
            <BasicType Type="BOOLEAN"/>
          </DataType>
          <InitialValue>TRUE</InitialValue>
          <Length>0</Length>
        </DataField>
        <DataField Id="contextvalue2" IsArray="FALSE">
          <DataType>
            <BasicType Type="BOOLEAN"/>
          </DataType>
          <InitialValue>TRUE</InitialValue>
          <Length>0</Length>
        </DataField>
        <DataField Id="contextvalue3" IsArray="FALSE">
          <DataType>
            <BasicType Type="BOOLEAN"/>
          </DataType>
          <InitialValue>TRUE</InitialValue>
          <Length>0</Length>
        </DataField>
      </DataFields>
      <Participants>
        <Participant Id="CII">

```

```

    <ParticipantType Type="SYSTEM"/>
    <Description>Reference to Context Information Framework</Description>
  </Participant>
  <Participant Id="P1">
    <ParticipantType Type="HUMAN"/>
    <Description>Human client</Description>
  </Participant>
  <Participant Id="P2">
    <ParticipantType Type="HUMAN"/>
    <Description>Human client</Description>
  </Participant>
</Participants>
<Applications>
  <Application Id="subscribeContextSource">
    <FormalParameters>
      <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
        <DataType>
          <BasicType Type="STRING"/>
        </DataType>
      </FormalParameter>
      <FormalParameter Id="contextcondition" Index="2" Mode="IN">
        <DataType>
          <BasicType Type="STRING"/>
        </DataType>
      </FormalParameter>
      <FormalParameter Id="operation" Index="3" Mode="IN">
        <DataType>
          <BasicType Type="STRING"/>
        </DataType>
      </FormalParameter>
    </FormalParameters>
  </Application>
  <Application Id="pollContextSource1">
    <FormalParameters>
      <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
        <DataType>
          <BasicType Type="STRING"/>
        </DataType>
      </FormalParameter>
      <FormalParameter Id="contextvalue1" Index="2" Mode="OUT">
        <DataType>
          <BasicType Type="BOOLEAN"/>
        </DataType>
      </FormalParameter>
    </FormalParameters>
  </Application>
  <Application Id="pollContextSource2">
    <FormalParameters>
      <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
        <DataType>
          <BasicType Type="STRING"/>
        </DataType>
      </FormalParameter>
      <FormalParameter Id="contextvalue2" Index="2" Mode="OUT">
        <DataType>
          <BasicType Type="BOOLEAN"/>
        </DataType>
      </FormalParameter>
    </FormalParameters>
  </Application>

```

```

<Application Id="pollContextSource3">
  <FormalParameters>
    <FormalParameter Id="context_source_attributes" Index="1" Mode="IN">
      <DataType>
        <BasicType Type="STRING"/>
      </DataType>
    </FormalParameter>
    <FormalParameter Id="contextvalue3" Index="2" Mode="OUT">
      <DataType>
        <BasicType Type="BOOLEAN"/>
      </DataType>
    </FormalParameter>
  </FormalParameters>
</Application>
</Applications>
<Activities>
  <Activity Id="1" Name="Activity1">
    <Implementation>
      <Tool Id="pollContextSource1" Type="APPLICATION">
        <ActualParameters>
          <ActualParameter>BOOLEAN1</ActualParameter>
          <ActualParameter/>
        </ActualParameters>
      </Tool>
      <Tool Id="subscribeContextSource" Type="APPLICATION">
        <ActualParameters>
          <ActualParameter>BOOLEAN1</ActualParameter>
          <ActualParameter>[!BOOLEAN1]</ActualParameter>
          <ActualParameter>contextvaluechange</ActualParameter>
        </ActualParameters>
      </Tool>
    </Implementation>
    <Performer>CII</Performer>
    <TransitionRestrictions/>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="2" Name="Activity2">
    <Route/>
    <Performer>P1</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="XOR"/>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="3" Name="Activity3">
    <Route/>
    <Performer>P2</Performer>
    <TransitionRestrictions>
      <TransitionRestriction>
        <Join Type="XOR"/>
      </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes/>
  </Activity>
  <Activity Id="4" Name="Activity4">
    <Implementation>
      <Tool Id="pollContextSource2" Type="APPLICATION">
        <ActualParameters>

```

```

    <ActualParameter>BOOLEAN2</ActualParameter>
  <ActualParameter/>
</ActualParameters>
</Tool>
<Tool Id="subscribeContextSource" Type="APPLICATION">
  <ActualParameters>
    <ActualParameter>BOOLEAN2</ActualParameter>
    <ActualParameter>[!BOOLEAN2]</ActualParameter>
    <ActualParameter>contextvaluechange</ActualParameter>
  </ActualParameters>
</Tool>
</Implementation>
<Performer>CII</Performer>
<TransitionRestrictions/>
<ExtendedAttributes/>
</Activity>
<Activity Id="5" Name="Activity5">
  <Implementation>
    <Tool Id="pollContextSource3" Type="APPLICATION">
      <ActualParameters>
        <ActualParameter>BOOLEAN3</ActualParameter>
        <ActualParameter/>
      </ActualParameters>
    </Tool>
    <Tool Id="subscribeContextSource" Type="APPLICATION">
      <ActualParameters>
        <ActualParameter>BOOLEAN3</ActualParameter>
        <ActualParameter>[!BOOLEAN3]</ActualParameter>
        <ActualParameter>contextvaluechange</ActualParameter>
      </ActualParameters>
    </Tool>
  </Implementation>
  <Performer>CII</Performer>
  <TransitionRestrictions/>
  <ExtendedAttributes/>
</Activity>
<Activity Id="6" Name="Activity6">
  <Route/>
  <Performer>P1</Performer>
  <TransitionRestrictions>
    <TransitionRestriction>
      <Join Type="XOR"/>
    </TransitionRestriction>
  </TransitionRestrictions>
  <ExtendedAttributes/>
</Activity>
<Activity Id="7" Name="Activity7">
  <Route/>
  <Performer>P1</Performer>
  <TransitionRestrictions>
    <TransitionRestriction>
      <Join Type="XOR"/>
    </TransitionRestriction>
  </TransitionRestrictions>
  <ExtendedAttributes/>
</Activity>
<Activity Id="8" Name="Activity8">
  <Route/>
  <Performer>P2</Performer>
  <TransitionRestrictions>

```



```
<TransitionRestriction>
  <Join Type="XOR"/>
</TransitionRestriction>
</TransitionRestrictions>
<ExtendedAttributes/>
</Activity>
<Activity Id="9" Name="Activity9">
  <Route/>
  <Performer>P2</Performer>
  <TransitionRestrictions>
    <TransitionRestriction>
      <Join Type="XOR"/>
    </TransitionRestriction>
  </TransitionRestrictions>
  <ExtendedAttributes/>
</Activity>
</Activities>
<Transitions>
  <Transition Id="1" From="1" To="2">
    <Condition>contextvalue1 == "TRUE"</Condition>
  </Transition>
  <Transition Id="2" From="1" To="3">
    <Condition>contextvalue1 == "FALSE"</Condition>
  </Transition>
  <Transition Id="3" From="2" To="4">
  </Transition>
  <Transition Id="4" From="3" To="5">
  </Transition>
  <Transition Id="5" From="4" To="6">
    <Condition>contextvalue2 == "TRUE"</Condition>
  </Transition>
  <Transition Id="5" From="4" To="7">
    <Condition>contextvalue2 == "FALSE"</Condition>
  </Transition>
  <Transition Id="6" From="5" To="8">
    <Condition>contextvalue3 == "TRUE"</Condition>
  </Transition>
  <Transition Id="6" From="5" To="9">
    <Condition>contextvalue3 == "FALSE"</Condition>
  </Transition>
</Transitions>
</WorkflowProcess>
</WorkflowProcesses>
</Package>
```

Appendix N: Exception handler rules for invariant scenario

```

<?xml version="1.0" encoding="UTF-8"?>
<ExceptionHandling xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="exceptionschema.xsd">
  <Rules>
    <Rule Id="1">
      <Events>
        <Event Id="1">
          <Type>TransitionException</Type>
        </Event>
      </Events>
      <Conditions>
        <Condition Id="1">
          <Value>transition value == "UNDEFINED"</Value>
        </Condition>
      </Conditions>
      <Actions>
        <Action Id="1">
          <Generate Id="1">
            <Activity Id="1">
              <Name>verifyState</Name>
              <Description>Verify contextual state</Description>
              <Implementation>current</Implementation>
              <Performer>P1</Performer>
              <UseCurrent>FALSE</UseCurrent>
            </Activity>
            <Transition Id="1">
              <UseCurrent>TRUE</UseCurrent>
              <From>1</From>
            </Transition>
            <Transition Id="2">
              <ConditionValue>transition value == "UNDEFINED"</ConditionValue>
              <To>1</To>
              <From>current</From>
            </Transition>
          </Generate>
        </Action>
      </Actions>
    </Rule>
    <Rule Id="2">
      <Events>
        <Event Id="1">
          <Type>TransitionException</Type>
        </Event>
      </Events>
      <Conditions/>
      <Actions>
        <Action Id="1">
          <RevalidateProcessPath>TRUE</RevalidateProcessPath>
        </Action>
      </Actions>
    </Rule>
  </Rules>

```

```
<Rule Id="3">
  <Events>
    <Event Id="1">
      <Type>InvariantException</Type>
    </Event>
  </Events>
  <Conditions/>
  <Actions>
    <Action Id="1">
      <RevalidateProcessPath>TRUE</RevalidateProcessPath>
    </Action>
  </Actions>
</Rule>
</Rules>
</ExceptionHandler>
```

Appendix O: Workflow client with local process enactment and rule based building of the situated process – process description

```

<?xml version="1.0" encoding="UTF-8"?>
<Package Id="0" Name="test context query workflow process"
xmlns="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xyz="http://www.xyzeorder.com/workflow"
xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-
1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>0.09</XPDLVersion>
    <Vendor>Man og Jon Ole</Vendor>
    <Created>1/21/2004 5:27:17 PM</Created>
  </PackageHeader>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <Script Type="text/javascript"/>
  <TypeDeclarations/>
  <Participants/>
  <Applications/>
  <DataFields/>
  <WorkflowProcesses>
    <WorkflowProcess AccessLevel="PUBLIC" Id="1" Name="ContextAwareClient">
      <ProcessHeader/>
      <FormalParameters/>
      <DataFields/>
      <Participants>
        <Participant Id="P1">
          <ParticipantType Type="HUMAN"/>
          <Description>Human client</Description>
        </Participant>
      </Participants>
      <Applications/>
      <Activities>
        <Activity Id="1" Name="Activity1">
          <Implementation>
            <SubFlow Id="2" Execution="SYNCHR">
              <ActualParameters/>
            </SubFlow>
          </Implementation>
          <Performer>P1</Performer>
          <TransitionRestrictions/>
          <ExtendedAttributes/>
        </Activity>
      </Activities>
    </WorkflowProcess>
  </WorkflowProcesses>
</Package>

```

```

    <Transitions/>
  </WorkflowProcess>
  <WorkflowProcess AccessLevel="PUBLIC" Id="2" Name="ContextAwareClient">
    <ProcessHeader/>
    <FormalParameters/>
    <DataFields>
      <DataField Id="client_kb" IsArray="FALSE">
        <DataType>
          <SchemaType>
            <xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
              <xs:element name="action">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="attribute" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="value" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="activity" minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="actions">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="action" minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="activity" type="xs:string"/>
              <xs:element name="attribute" type="xs:string"/>
              <xs:element name="condition">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="attribute"/>
                    <xs:element ref="value"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="conditions">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="condition" minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="goal">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="attribute"/>
                    <xs:element ref="text" minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="knowledgebase">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="goal"/>
                    <xs:element ref="rules"/>
                    <xs:element ref="questions"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:schema>
          </SchemaType>
        </DataType>
      </DataField>
    </DataFields>
  </WorkflowProcess>

```

```

</xs:element>
<xs:element name="name" type="xs:string"/>
<xs:element name="poll-attrib" type="xs:string"/>
<xs:element name="question">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="attribute"/>
      <xs:element ref="poll-attrib"/>
      <xs:element ref="response" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="questions">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="question" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="response" type="xs:string"/>
<xs:element name="rule">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="conditions"/>
      <xs:element ref="actions"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="rules">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="rule" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="text" type="xs:string"/>
<xs:element name="value" type="xs:string"/>
</xs:schema>
</SchemaType>
</DataType>
<InitialValue>file://client_kb.xml</InitialValue>
<Length>0</Length>
</DataField>
</DataFields>
<Participants/>
<Applications>
  <Application Id="inferenceContext">
    <FormalParameters>
      <FormalParameter Id="client_kb" Index="1" Mode="IN">
        <DataType>
          <SchemaType>
            <xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
              <xs:element name="action">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="attribute" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="value" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="activity" minOccurs="0" maxOccurs="unbounded"/>

```

```

        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="actions">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="action" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="activity" type="xs:string"/>
<xs:element name="attribute" type="xs:string"/>
<xs:element name="condition">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="attribute"/>
            <xs:element ref="value"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="conditions">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="condition" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="goal">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="attribute"/>
            <xs:element ref="text" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="knowledgebase">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="goal"/>
            <xs:element ref="rules"/>
            <xs:element ref="questions"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="name" type="xs:string"/>
<xs:element name="poll-attrib" type="xs:string"/>
<xs:element name="question">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="attribute"/>
            <xs:element ref="poll-attrib"/>
            <xs:element ref="response" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="questions">
    <xs:complexType>
        <xs:sequence>

```

```

        <xs:element ref="question" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="response" type="xs:string"/>
<xs:element name="rule">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="name"/>
            <xs:element ref="conditions"/>
            <xs:element ref="actions"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="rules">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="rule" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="text" type="xs:string"/>
<xs:element name="value" type="xs:string"/>
</xs:schema>
</SchemaType>
</DataType>
</FormalParameter>
</FormalParameters>
</Application>
</Applications>
<Activities>
    <Activity Id="1" Name="inference">
        <Implementation>
            <Tool Id="inferenceContext" Type="APPLICATION">
                <ActualParameters>
                    <ActualParameter>client_kb</ActualParameter>
                </ActualParameters>
            </Tool>
        </Implementation>
        <TransitionRestrictions/>
        <ExtendedAttributes/>
    </Activity>
    <Activity Id="2" Name="Activity2">
        <Route/>
        <TransitionRestrictions/>
        <ExtendedAttributes/>
    </Activity>
    <Activity Id="3" Name="Activity3">
        <Route/>
        <TransitionRestrictions/>
        <ExtendedAttributes/>
    </Activity>
    <Activity Id="4" Name="Activity4">
        <Route/>
        <TransitionRestrictions/>
        <ExtendedAttributes/>
    </Activity>
    <Activity Id="5" Name="Activity5">
        <Route/>
        <TransitionRestrictions/>
    </Activity>

```

```
<ExtendedAttributes/>
</Activity>
<Activity Id="6" Name="Activity6">
  <Route/>
  <TransitionRestrictions/>
  <ExtendedAttributes/>
</Activity>
</Activities>
<Transitions/>
</WorkflowProcess>
</WorkflowProcesses>
</Package>
```

Appendix P: Workflow client with local process enactment and rule based building of the situated process – client knowledge base

```

<?xml version="1.0" encoding="utf-8" ?>
<knowledgebase xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="sie.xsd">
  <goal>
    <attribute>processfinished</attribute>
  </goal>
  <rules>
    <rule>
      <name>1</name>
      <conditions>
        <condition>
          <attribute>contextvalue1</attribute>
          <value>FALSE</value>
        </condition>
      </conditions>
      <actions>
        <action>
          <activity>2</activity>
        </action>
      </actions>
    </rule>
    <rule>
      <name>2</name>
      <conditions>
        <condition>
          <attribute>contextvalue1</attribute>
          <value>TRUE</value>
        </condition>
      </conditions>
      <actions>
        <action>
          <activity>3</activity>
        </action>
      </actions>
    </rule>
    <rule>
      <name>3</name>
      <conditions>
        <condition>
          <attribute>contextvalue1</attribute>
          <value>TRUE</value>
        </condition>
      </conditions>
    </rule>
  </rules>
</knowledgebase>

```

```
        <attribute>contextvalue2</attribute>
        <value>TRUE</value>
    </condition>
</conditions>
<actions>
    <action>
        <attribute>processfinished</attribute>
        <value>TRUE</value>
    </action>
    <action>
        <activity>4</activity>
    </action>
</actions>
</rule>
<rule>
    <name>4</name>
    <conditions>
        <condition>
            <attribute>contextvalue1</attribute>
            <value>TRUE</value>
        </condition>
        <condition>
            <attribute>contextvalue2</attribute>
            <value>FALSE</value>
        </condition>
    </conditions>
    <actions>
        <action>
            <attribute>processfinished</attribute>
            <value>TRUE</value>
        </action>
        <action>
            <activity>5</activity>
        </action>
    </actions>
</rule>
<rule>
    <name>5</name>
    <conditions>
        <condition>
            <attribute>contextvalue1</attribute>
            <value>FALSE</value>
        </condition>
        <condition>
            <attribute>contextvalue2</attribute>
            <value>TRUE</value>
        </condition>
    </conditions>
    <actions>
        <action>
            <attribute>processfinished</attribute>
            <value>TRUE</value>
        </action>
        <action>
            <activity>6</activity>
        </action>
    </actions>
</rule>
<rule>
    <name>6</name>
```

```
<conditions>
  <condition>
    <attribute>contextvalue1</attribute>
    <value>FALSE</value>
  </condition>
  <condition>
    <attribute>contextvalue2</attribute>
    <value>FALSE</value>
  </condition>
</conditions>
<actions>
  <action>
    <attribute>processfinished</attribute>
    <value>TRUE</value>
  </action>
  <action>
    <activity>6</activity>
  </action>
</actions>
</rule>
</rules>
<questions>
  <question>
    <attribute>contextvalue1</attribute>
    <poll-attrib>BOOLEAN.V1</poll-attrib>
    <response>TRUE</response>
    <response>FALSE</response>
  </question>
  <question>
    <attribute>contextvalue2</attribute>
    <poll-attrib>BOOLEAN.V2</poll-attrib>
    <response>TRUE</response>
    <response>FALSE</response>
  </question>
</questions>
</knowledgebase>
```

Appendix Q: Workflow client with activity contextual post conditions process description

```

<?xml version="1.0" encoding="UTF-8"?>
<Package Id="0" Name="test context query workflow process"
xmlns="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xyz="http://www.xyzeorder.com/workflow"
xsi:schemaLocation="http://www.wfmc.org/2002/XPDL1.0 http://wfmc.org/standards/docs/TC-
1025_schema_10_xpdl.xsd">
  <PackageHeader>
    <XPDLVersion>0.09</XPDLVersion>
    <Vendor>Man og Jon Ole</Vendor>
    <Created>6/2/2004 5:27:17 PM</Created>
  </PackageHeader>
  <ConformanceClass GraphConformance="NON_BLOCKED"/>
  <Script Type="text/javascript"/>
  <TypeDeclarations/>
  <Participants/>
  <Applications/>
  <DataFields/>
  <WorkflowProcesses>
    <WorkflowProcess AccessLevel="PUBLIC" Id="1" Name="ContextAwareClient">
      <ProcessHeader/>
      <FormalParameters/>
      <DataFields/>
      <Participants>
        <Participant Id="P1">
          <ParticipantType Type="HUMAN"/>
          <Description>Human client</Description>
        </Participant>
      </Participants>
      <Applications>
        <Application Id="contextual_actuator">
          <FormalParameters>
            <FormalParameter Id="postcondition_attrib" Index="1" Mode="IN">
              <DataType>
                <BasicType Type="STRING"/>
              </DataType>
            </FormalParameter>
            <FormalParameter Id="postcondition_value" Index="2" Mode="IN">
              <DataType>
                <BasicType Type="BOOLEAN"/>
              </DataType>
            </FormalParameter>
          </FormalParameters>
        </Application>
      </Applications>
      <Activities>
        <Activity Id="1" Name="Activity1">

```

```
<Implementation>
  <Tool Id="contextual_actuator" Type="APPLICATION">
    <ActualParameters>
      <ActualParameter>BOOLEAN1</ActualParameter>
      <ActualParameter>TRUE</ActualParameter>
    </ActualParameters>
  </Tool>
</Implementation>
<Performer>P1</Performer>
<TransitionRestrictions/>
<ExtendedAttributes/>
</Activity>
</Activities>
<Transitions/>
</WorkflowProcess>
</WorkflowProcesses>
</Package>
```