Workflow Interoperability: The WfMC Reference Model and an Implementation

MASTER THESIS

Submitted by:
Atila Kaya
Matriculation Number: 13653
19.04.2001

TECHNICAL UNIVERSITY HAMBURG-HARBURG
GERMANY

Supervised by:

Prof. Dr. Joachim W. Schmidt
Software Systems Department
TECHNICAL UNIVERSITY HAMBURG-HARBURG

and

Prof. Dr. Wolfgang Meyer
Process Automation Techniques Department
TECHNICAL UNIVERSITY HAMBURG-HARBURG

I declare that:

this work has been prepared by myself,
all literally or content-related quotations from other sources are clearly pointed out,
and no other sources or aids than the ones that are declared are used.

Hamburg, 19.04.2001

Atila KAYA

Table of Contents

Bibliography

Chapter 1

Introduction

In the last few years, the Internet has become more than an information and communication medium. People are increasingly using the Internet for purposes such as online banking, travel booking or buying of goods.

Parallel to these developments, enterprises are obligated to participate in the resulting global competition while trying to fulfill the drastically increasing demands. Furthermore, these demands have also led to the definition of new forms of business, e.g., electronic commerce, business to business commerce, virtual enterprises, etc.

The workflow management technology has been used for management of business processes in the past. However, most workflow management systems follow a process-centric approach and are restricted to intra-organizational applications.

Nowadays, enterprises and organizations have to collaborate with other enterprises and organizations in order to fulfill the requirements of global competition. This development can be realized not only at companies employing electronic commerce but also at companies that are doing traditional business, e.g., car manufacturing.

Considering the fact that enterprises are already using different workflow management products, an appropriate way to allow different workflow management systems to interoperate is essential.

Therefore, “workflow interoperability” is the key issue when supporting the implementation of interorganizational workflows.

1.1 Goals

The main goal of this work is to investigate and evaluate an approach that enables interoperability of existing workflow management systems.

Special emphasis will be given to the approach defined by the Workflow Management Coalition [WfMC96]. Hence, the proof of the workflow interoperability concept of the Workflow Management Coalition is aimed.

In particular, considering the Interoperability Interface standardized by the Workflow Management Coalition [WfMC00], an interoperability framework will be designed and implemented for an existing workflow management system. The applicability of the defined interoperability framework will be tested and evaluated.

Furthermore, a different approach on workflow interoperability, which has been implemented in the MARIFlow project, will be presented briefly.

1.2 Structure

This work is structured as follows:

In the second chapter, a comprehensive presentation of workflow management systems will be made. The presentation will start with the history of workflow management representing the origins, evolution and generations of workflow management. Next, the significance of workflow management technology for manufacturing processes will be studied. The significance of workflow management for electronic commerce will be described in the third section.

After a brief introduction of the Workflow Management Coalition (WfMC) that was founded to standardize and improve the workflow management technology, the third chapter will present the Workflow Reference Model of the WfMC. An overview on workflow management systems will be given.

To introduce the workflow terminology, basic definitions and the relationship among them will be given in the first section. Next, the architectural aspects of workflow management systems including the components and interfaces will be presented. In the third section an implementation model of workflow management systems will be described. Finally the Workflow Reference Model, which is defined to identify the interfaces within the generic workflow product structure, will be studied in detail.

The rationale of the forth chapter is to present the Interoperability Interface defined by the Workflow Management Coalition.

The objectives of the Interoperability Interface will be represented first. Next, different levels of interoperability will be described. Subsequently, models of interoperability will be introduced. Afterwards, the interoperability level to be implemented will be studied in more detail. Accordingly, sample processes and interoperability scenarios contributing case studies will be defined. These case studies will be used to evaluate the functionality of the interoperability framework. The last section of chapter four will comprise future perspectives of workflow interoperability. Here, beside the future plans of the Workflow Management Coalition to improve workflow interoperability, a Petri net-based approach for workflow management and the role it may play for workflow interoperability will be studied.

Chapter five will focus on the WFMC’s Interoperability Interface’s technical details. An interoperability framework will be presented as an implementation of the standardized Interoperability Interface. Analysis, design, implementation of the framework and a test environment will be represented. Furthermore, the fifth chapter will evaluate workflow interoperability approaches implemented in two projects with regard to the consequences recognized during the implementation. Particularly, after the implementation and the testing of the interoperability framework, the workflow interoperability approach of the Workflow Management Coalition will be evaluated. Furthermore, the MARIFlow project, which implements a different approach on interoperability, will be briefly presented and evaluated. In the last section of this chapter problems that are identified in both projects will be described.

Presenting the concluding remarks of this work will be the rationale of chapter six. First, the entire work will briefly be summarized. In the second section, critics about the interoperability approach of the Workflow Management Coalition will be made. Finally, further areas of research on workflow interoperability will be described.

Acknowledgements:

My special thanks go to the Software Company T-Nova Innovationsgesellschaft mbH and Software Systems Department (STS) of the Technical University Hamburg-Harburg.

Particularly, I would like to thank Dipl.-Inf. Matthias Andersen and Dipl.-Inf. Michael Sager of T-Nova for their encouragement and guidance especially during the testing of the interoperability framework.

In addition, I would like to thank Dipl.-Inf. Patrick Hupe and Prof. Dr. Joachim W. Schmidt of STS for their guidance to this work.

Chapter 2

Workflow Management Systems

The main goal of this chapter is to introduce workflow management systems. In the first section, the history of workflow management representing the origins, evolution and generations of workflow management systems will be presented.

The significance of workflow management for manufacturing processes is the subject of the second section.

Subsequently, the third section will describe the current and future role of workflow management technology in the context of electronic commerce.

2.1 History of Workflow Management

To draw a first picture of the workflow management concept, analyzing the history of workflow management is an appropriate starting point. For this purpose, first information technology development that has influenced and impacted the development of workflow management will be discussed.

Next, solutions to the new requirements that contributed to the evolution of workflow management will be described. Finally, different stages in the development of workflow management technology will be defined.

2.1.1 Origins of Workflow Management

Two major developments in information technology may be considered as the triggers of the advent of workflow management. The first one is the enormous technological progress in hardware and software technologies in the last two decades. With the rapid improvement in hardware technology a revolution took place in office work. Personal computers and workstations with continuously increasing performance were produced for lower prices. As a result, computer technology has started to dominate the office work. Besides this, network technology enabled the interconnection of computers to build up networks. Furthermore, the connectivity of hardware systems enabled the development of integrated software systems.

The evolution in database systems enabled the integration of office work opening new dimensions for software development. As a consequence of these improvements in hardware technology and database systems, more sophisticated software such as sophisticated word processing, spreadsheet and imaging applications were developed.

However, the overall productivity in application areas did not increase proportionally to these improvements due to the lack of integration of the software products. In order to achieve better results, a change in the concept of application system development was necessary.

This necessity became the second trigger for the advent of workflow management. The existing goal of automating the pieces of application systems had been replaced by the broader new approach of the comprehensive development of integrated application systems. This new goal required a shift from task- or data-centric view to process- or work-centric view, in the application system development process.

The task- or data-centric view focuses on the integration of data to achieve the efficient implementation of distinct tasks with respect to the dependencies among them. In contrast, the process- or work-centric view focuses on the work as a whole. This approach concentrates on the work from a broader perspective by integrating all related programs, processes, functions, data, documents, persons, organizations etc. Hence, this enables the optimization of processes to minimize the costs and increase the efficiency [JaBu96].

2.1.2 Evolution of Workflow Management

The developments in information technology mentioned in Subsection 2.1.1 led to the development of new software technologies, which have supported some aspects of the workflow management for a period of time.

Next, these software technologies that contributed to the evolution workflow management will be elaborated:

Office Automation

Workflow management can be regarded as an outcome of the office automation concept that started in the 1970s. Both workflow management systems and office automation aim at the automation of the execution of work.

The difference is that office automation aims at automating individual tasks of the work, whereas workflow management aims at automating the control of tasks during the execution of a business process. Thus, workflow management systems assist human workers in performing work processes.

Despite the different concepts of reaching automation of work, some fundamental requirements of office information systems such as activity scheduling, function integration, personal assistance and task management can be applied to workflow management systems [BrPe84].

Image-Processing

Several business processes involve interactions with paper-based information, which may need to be captured as image data as part of an automation process.

Usually the captured image data is required to be passed to persons involved in a particular process. The persons that work interact with different software programs for different purposes. As a consequence, image-processing systems are required to have some workflow capability either built-in or supplied in conjunction with a workflow management product [WfMC95].

Document Management

The wide-spread of computer technology in office work resulted in the replacement of paper documents with electronic documents. The first generation of document management systems were called *passive document management systems* because they only reacted on direct user requests. For instance, a user could query for a certain document or he could lock a class of documents. Context-based retrieval and document management functions (e.g. lock, release, combine, destroy) were the major features of passive document management systems.

Active document management systems enhanced passive document management by incorporating service functions which are based on time management. For example, triggers caused a document to be presented for review after a certain period of time. Oversimplified, this was a first step towards workflow management. Active document management systems are considered as the ancestors of document–centric workflow management systems [JaBu96].

Electronic Mail

Enhanced electronic mail systems are used to distribute information among individuals considering their attributes such as organization roles. Thus, electronic mail systems and workflow management systems share some common characteristics.

Groupware Applications

Groupware is defined as computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to shared environment [EGR91].

Groupware is the technology designed to facilitate the work of groups. This technology may be used for communication, cooperation, coordination, problem solving, competition, or negotiation. While traditional communication technologies like the telephone also qualify as groupware, the term is ordinarily used to refer to a specific class of technologies relying on modern computer networks, such as email, newsgroups, videophones, or chat. Groupware technologies are typically categorized along two primary dimensions:

- Whether users of the group work/act at the same time (synchronously) or distributed over time (asynchronously)
- Whether users work/act in the same place or in different locations

CSCW (Computer-Supported Cooperative Work) is closely related and sometimes confused with groupware. CSCW refers to the field of study which examines the design, adoption, and use of groupware. Despite the name, this field of study is not restricted to issues of "cooperation" or "work" but also examines competition, socialization, and play.

Groupware is regarded as one possible and suitable superordinated application area for workflow management [Boc93].

However, groupware and workflow management systems differ in their rationales. Groupware provides support for unstructured, ad-hoc processes whereas workflow management systems primarily support structured, recurring processes.

Database Management

Conventional database management systems are passive, so that they only response to explicit requests from applications or users. Active database management systems enrich passive database management systems by implementing event-condition-action (ECA) rules [McDa89]: When an event occurs and if a condition holds, an action will be executed

A simple workflow can be considered as a multi-step activity consisting of miscellaneous activities. Active database management systems can support multi-step activities by implementing each step of a multi-step activity as a transaction. In detail, this is achieved by implementing the control flow of transactions by ECA rules, which are embedded into the database.

Software Development Management

Software development management coordinates the development of software processes. The management of a software development process comprises three major phases: modelling of the software process, analysis of the model and execution of the model [JaBu96].

Often a software process has to provide some workflow functionality to transfer development tasks and necessary information among related persons.

Business Process Reengineering

With the change of conditions in the business world and the resulting change in the structure of business, redesigning business processes, also called business process reengineering (BPR), became obligatory for an increasing number of companies. As a consequence, business processes had and have to be remodelled, reanalyzed and redefined to achieve better performance.

Business Process Reengineering handles this problem with tools that support activities in areas such as analyzing, modelling and defining business processes. Here, the focus of interest is on the execution of whole business processes, not on the execution of single tasks.

Moreover, business processes consist of a number of different views on a business: technological view, economical view, information-oriented view etc. Thus, Business Process Reengineering takes different aspects of a business into account.

The conceptual ideas of business process modelling, especially its broad, multi-aspect approach, triggered the development of workflow management systems [JaBu96].

2.1.3 Generations of Workflow Management

Like many other software technologies the development of workflow management systems started with academic and commercial prototypes. The next stage was the development of conceptual models and architectures.

McCarthy and Bluestein classified workflow management technology development into three stages: homegrown (1989-1992), rudimentary (1992-1995), and dynamic (1994-1999) workflow computing [McBl91].

Homegrown workflow computing does neither include a distinguishable workflow model nor a workflow description language. Thus, workflow related data have to be hard-coded into application programs. Homegrown workflow computing shows very little or no adaptability at all. That’s why some authors do not even classify this technology as part of workflow management system development.

Rudimentary workflow management technology differs from the homegrown technology in the availability of an autonomous workflow engine. A workflow engine is a workflow execution module that provides the necessary run-time execution environment for a workflow instance. Changes to existing workflows are allowed until a certain degree determined by the workflow model and the execution engine.

According to Jablonski and Bussler, the difference between rudimentary and dynamic workflow management systems is that dynamic systems can be adjusted dynamically to new application requirements [JaBu96]. This is achieved due to the adaptability of the workflow management system’s architecture to new hard- and software infrastructures and the extensibility for additional functionality.

2.2 Significance of Workflow Management for Manufacturing Processes

In the early 1900s with the implementation of first assembly lines in the automobile industry the automation of manufacturing had its start. In spite of the fact that this change became a revolution and completely changed the concept of manufacturing, the requirement for higher efficiency and productivity remained the same until today.

At present modern manufacturing systems are sophisticated systems, which are based on processes that require appropriate coordination of activities to achieve predefined business goals. These activities are based on interactions between humans and non-humans, such as computers and robots. Most of these activities are assisted and executed by specialized engineering software tools and programs such as CAD (computer-aided design), CAM (computer-aided manufacturing), CAPP (computer-aided process planing) or NC (numerical control) programming programs. Besides, standard office programs such as an e-mail program, a word processor or a spreadsheet application are involved in almost every administrative activity.

Workflow management, an approach having its ancestors in different fields of computer science and system engineering, has been realized as the key for optimizing manufacturing processes. A workflow management system is defined as a system, which provides procedural automation of a business process managing the sequence of work activities and the invocation of appropriate human and/or IT resources associated with the various activity steps [WfMC93]. Workflow management systems are not only well-suited for office environments (e.g. within an insurance company) but also comply with the requirements of engineering environments (e.g. the manufacturing environment) [JaBu96].

The following example demonstrates the implementation of a workflow management system in the manufacturing area. Figure 2.1 shows the fundamental parts of the manufacturing control process of a leading producer of turbo chargers in Europe.

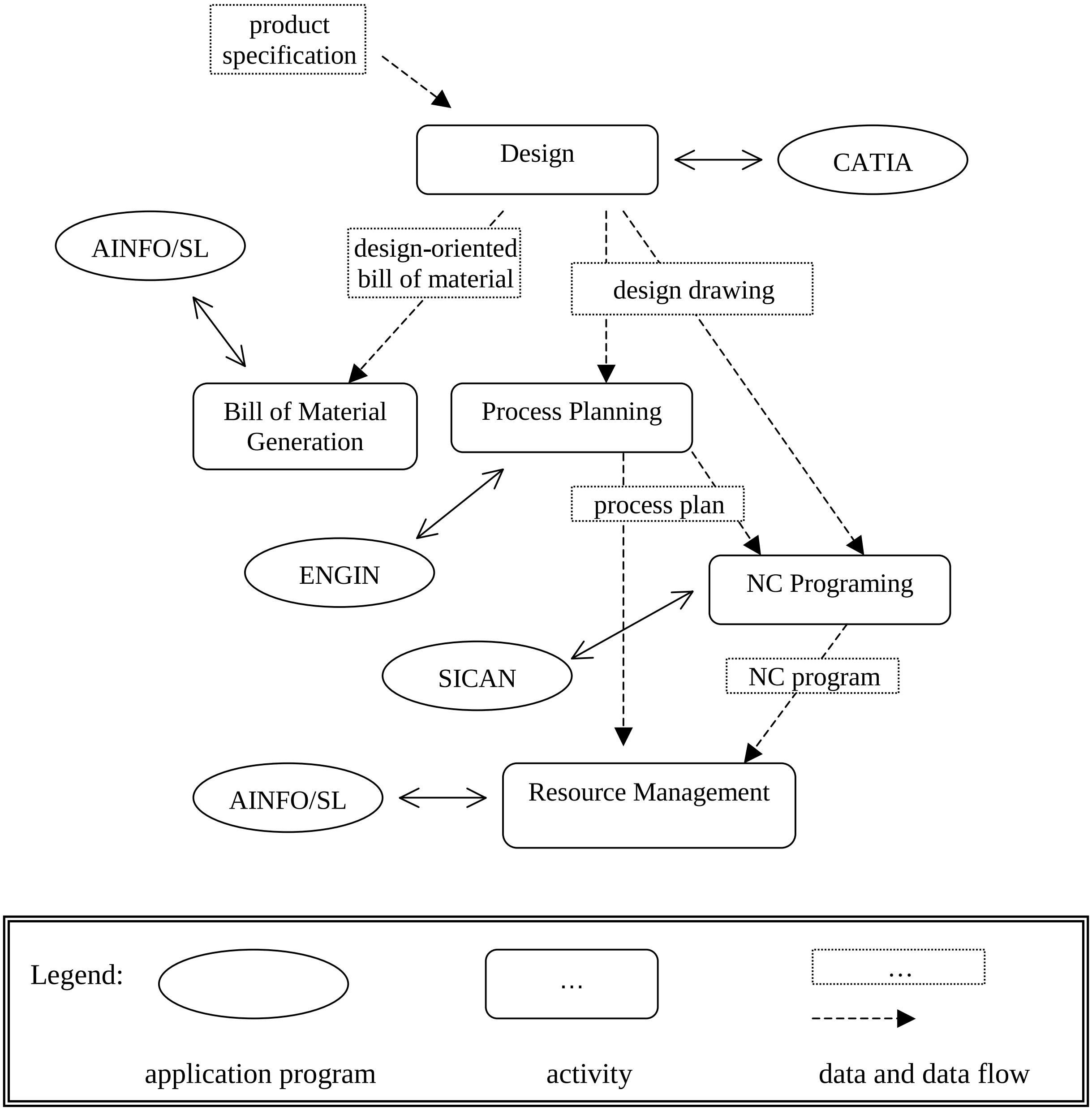

Figure 2.1: Structure of the Manufacturing Control Process Example

The manufacturing process analyzed in the following comprises the CAD and CAPP areas. The main task in the design phase is the generation of technical drawings for part manufacturing. The drawings are sent to the CAPP area, specifically to the two sub-areas “Bill of Material Generation” and “Process Planning”. At this point, bills of materials and process plans are defined. Afterwards, the “Numeric Control” (NC) programming department is called to deliver NC programs for those operations of the process plans which refer to NC machines. Finally, the NC programs are sent to the “Resource Management” department where the availability of reference NC machines, tools and fixtures is checked [JaBu96].

As seen in Figure 2.1, each step of the process is associated with an application program (AINFO/SL, CATIA, ENGIN, SICAN). Persons using the associated application program are responsible for the execution of a step.

Nowadays, the expanding usage of the Internet in business applications and globalization results in a global competition. As a result, this challenge leads to requirements becoming more difficult to achieve for the manufacturing area. Workflow management systems bring following major advantages to fulfill these new requirements:

- Enterprises are forced to outsource their production into low cost locations, usually into different countries, in order to reduce manufacturing costs. Workflow management systems enable the implementation of outsourcing.
- Another requirement is the necessity to manufacture more flexibly and adapt faster to the rapid changes in the market. Workflow management systems are appropriate tools for the redefinition of existing processes and their implementations.

  In the current stage of workflow computing, workflow management shows dynamics: The workflow model can be adjusted dynamically to new application requirements, the workflow management systems architecture can be adopted to new hard- and software infrastructures and it can be extended systematically with additional functionality [JaBu96].
- The changes in business conditions make the estimation of the required capacity for manufacturing processes more difficult. Thus, in several cases after a certain period of time, processes have to support more activities than initially planned and implemented. Workflow management systems enable distributed enactment of processes and therefore bring a major advantage considering scalability and load-balancing.
- The globalization and resulting high competition in the market requires companies to become more than a “typical” manufacturer. Moreover, they are forced to consider their manufacturing processes as part of a supply chain. Supply chain management covers whole activities starting from accepting an order to the delivery of the finished product. As a result, supply chain management requires the participation of wholesalers, manufacturers and customers. A supplier in a supply chain has to integrate four different business processes: planning, sourcing, manufacturing and delivery. With the usage of workflow management, a supply chain can be implemented and managed efficiently [AnAl01].

Workflow interoperability, which will be explained in Chapter 4 in more detail, opens up new dimensions for workflow management. Enterprises equipped with workflow management systems with interoperability features are capable of facing the requirements mentioned above.

2.3 Significance of Workflow Management for E-Commerce

With the rapid growth in the recent past, the Internet dominated two major areas, namely communication and information distribution. In a short period of time most enterprises started making use of the Internet as an information and communication platform.

Nowadays, most attention is focused on the transaction supporting facility of Internet enabling E-Commerce. E-Commerce can be defined as the direct electronic exchange of goods and services using computers and telecommunication to send and receive goods [WfMC01]. The Internet provides the worldwide platform for E-Commerce enabling global competition in the market. All services offered on the Internet are enabled through the execution of some back-end processes. For example, a company selling books online on its web site runs a process behind to manage and execute customer orders. After an order is received, some activities like searching in the appropriate stock to send the book, selecting a delivery agent, communicating with the stock and the agent, monitoring delivery, handling exceptions, e.g., in case the book is sent to a wrong address etc have to be executed. These activities define the steps of a process. Workflow management systems are ideal tools to manage such business processes:

The delivery of goods and services through E-Commerce requires the execution of business processes that span several organizations and/or enterprises (interorganizational processes). Workflow interoperability enables the execution of processes that may not be restricted by enterprise boundaries. Workflow interoperability will therefore equip enterprises for business to business (B2B) commerce.

Sometimes several units or companies of different enterprises work together on one process. With workflow interoperability each unit may have its own workflow engine responsible for the execution of its process part. The usage of existing processes and related knowledge is an important advantage for businesses. Therefore, workflow interoperability will open the way for virtual enterprises.

Moreover, workflow interoperability creates the opportunity for automated trading [WfMC01]. An organization selling a product may send out tenders to interested parties. The workflow engines of other organizations, when receiving these tenders, would start to negotiate offers. Finally, the workflow engine of the producer, which receives these offers, would accept the best offer and commit a transaction.

As a consequence, workflow interoperability will enable organizations to offer better products and services and adapt faster to market changes.

Chapter 3

The WfMC Reference Model

Workflow management (WFM) is a fast evolving technology that is used in a variety of industrial processes. The evolution of workflow management concentrates on the automation of business processes where interactions of human and machine-based activities play a major role. Within workflows, information or tasks are passed among participants in a way that is governed by rules or procedures.

The concept of workflow management has been researched and developed since the beginning of the 1980s with the efforts of several research groups. Thus, workflow software products evolved from several different origins. While some products have been developed as pure workflow software, many have evolved from image processing systems, document management systems, relational or object database systems, and electronic mail systems. As a consequence some vendors have invented new specific terminology and interfaces, whereas others adopted terminology and interfaces from other technologies.

This development resulted in the creation of a large number of different workflow products, which were focused on specific aspects of workflow management. This enabled the users to choose a product that met their specific application needs best. On the other hand, the definition of a common terminology to enable the standardization of workflow products became necessary. The definition of a common terminology is the only way to define new standards that enable different workflow products to interoperate.

Only a successful standardization could enable workflow management system users to choose the product that suits their requirements best in a specific application area and combine the strength of different products in one infrastructure.

Due to these requirements, several vendors developing workflow management systems founded the Workflow Management Coalition (WfMC) in 1993 as a nonprofit international organization. Although there had been a large number of products specialized on different aspects, all products had some common characteristics. This helped defining standards for various functions so that some level of interoperability among different products could be achieved.

The Workflow Management Coalition (WfMC) has been established to identify these functional areas and to develop appropriate specifications for implementation in workflow products. It is intended that such specifications will enable interoperability between heterogeneous workflow products and improved integration of workflow applications with other IT services such as electronic mail and document management, thereby improving the opportunities for the effective use of workflow technology, to the benefit of both vendors and users of such technology [WfMC95].

The WfMC states its missions as:

- To increase the value of customers investments with workflow technology
- To decrease the risk of using workflow products
- To expand the workflow market through increasing the awareness of workflow

After the definition of a common terminology of workflow management, the WfMC focused on the standardization of interfaces between modules of a workflow management system but also between workflow management systems and their clients. Several working groups were created to discuss and propose these interfaces in order to standardize them. Figure 2.2 illustrates briefly the components and interfaces standardized by WfMC:

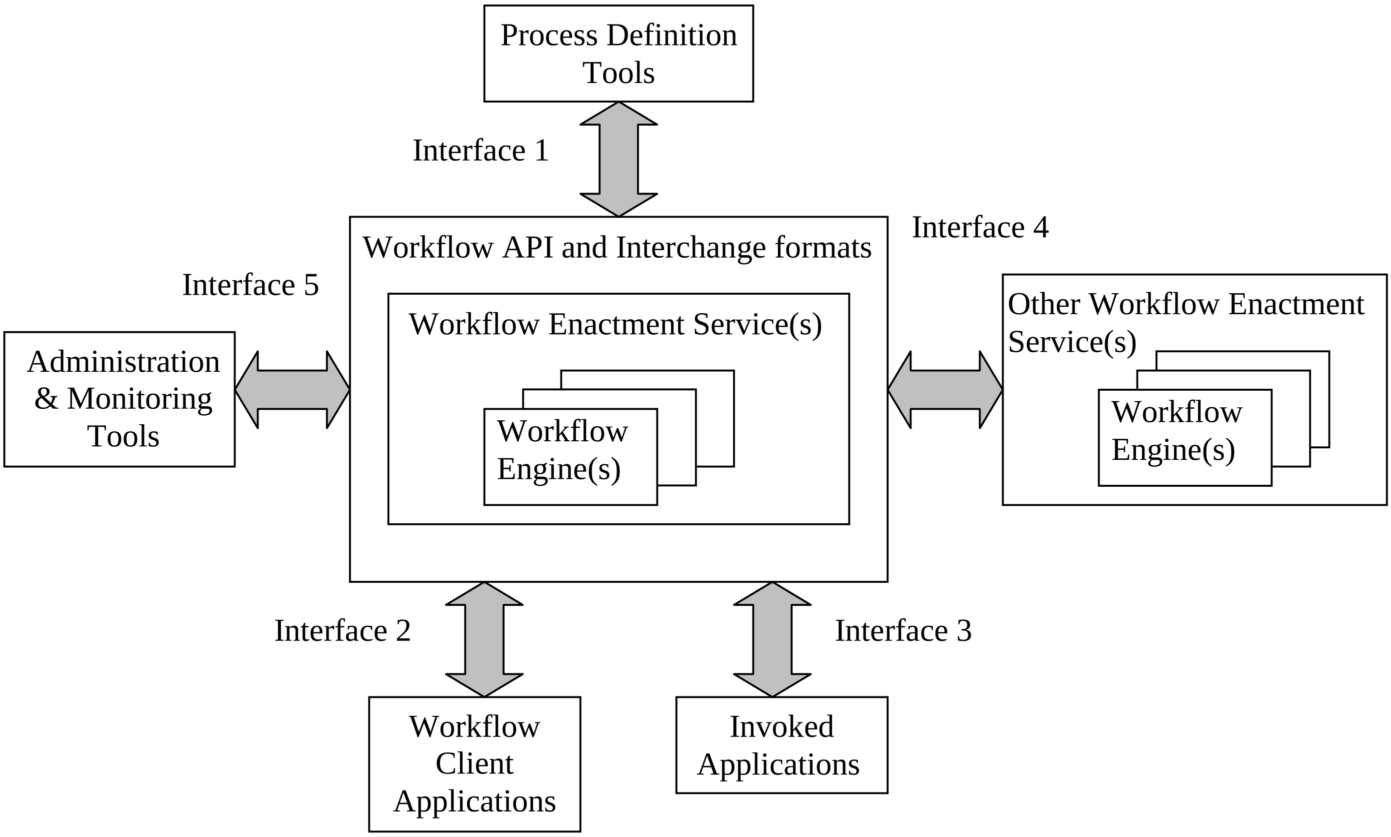
Process Definition Tools
Interface 1
Workflow API and Interchange formats
Interface 4
Interface 5
Workflow Enactment Service(s)
Other Workflow Enactment Service(s)
Administration & Monitoring Tools
Workflow Engine(s)
Workflow Engine(s)
Interface 2
Interface 3
Workflow Client Applications
Invoked Applications

Figure 3.1: The WfMC Workflow Reference Model – Components & Interfaces

The Workflow Management Coalition has defined a common reference model for workflow management products by identifying their characteristics, terminology and concepts. The goal of this chapter is to study this workflow reference model. Here, the standards and publications of the Workflow Management Coalition will be used as the main reference.

In the first section, basic definitions of the workflow management technology and the relationships among them will be given.
Components and interfaces contributing the architecture of workflow management systems will be introduced in the second section.
Next, the implementation model of a workflow system that suits the majority of the workflow management products will be presented.
The workflow reference model developed by the Workflow Management Coalition to identify the interfaces within the generic workflow product structure will be presented in the last section.

3.1 Basic Definitions

The basic terms of workflow terminology and the relationships among them are illustrated in Figure 3.2 according to the Workflow Management Coalition [WfMC99a]. Next, these basic terms will be defined:

Workflow

A workflow is concerned with the automation of a business process, in whole or partly, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [WfMC99a].
The automation of a business process is done within the process definition with respect to several process activities, procedural rules and associated control data used to manage the workflow execution.

Workflow Management System

A Workflow Management System (WFMS) is a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications [WfMC99a].
A workflow management system consists of software components to store and interpret process definitions, create and manage workflow instances as they are executed, and control their interaction with workflow participants and applications [WfMC99a].

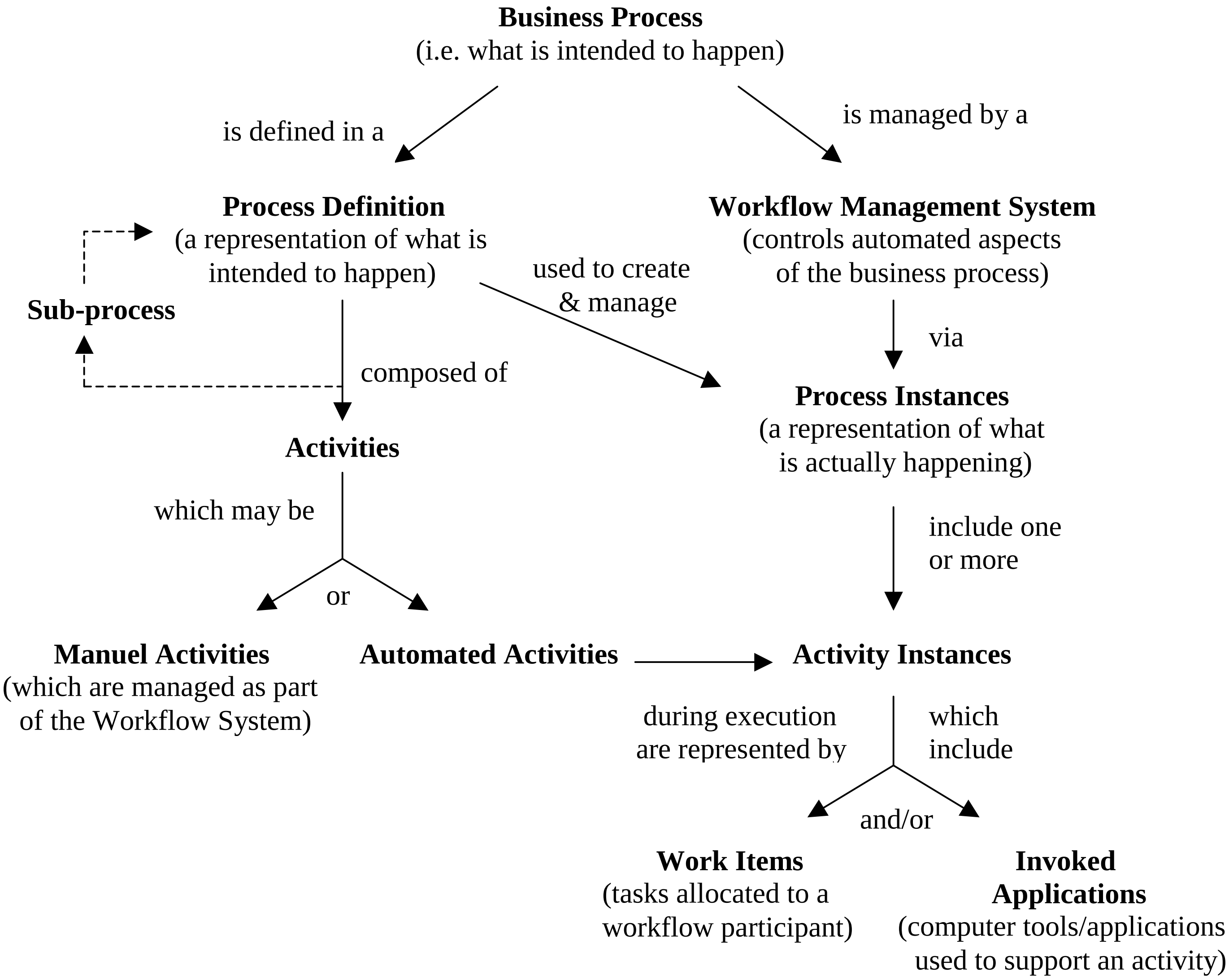
Business Process
(i.e. what is intended to happen)
is defined in a
is managed by a
Process Definition
(a representation of what is intended to happen)
Workflow Management System
(controls automated aspects of the business process)
used to create & manage
Sub-process
via
composed of
Process Instances
(a representation of what is actually happening)
Activities
which may be
include one or more
or
Manuel Activities
(which are managed as part of the Workflow System)
Automated Activities
Activity Instances
during execution are represented by
which include
and/or
Work Items
(tasks allocated to a workflow participant)
Invoked Applications
(computer tools/applications used to support an activity)

Figure 3.2: Basic Terms and their Relationships

Business Process

A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships [WfMC99a].

Process Definition

The representation of a business process in a form, which 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 [WfMC99a].

Activity

A description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resource(s) to support process execution: where human resource is required, an activity is allocated to a workflow participant [WfMC99a].

Automated Activity

An activity which is capable of computer automation using a workflow management system to manage the activity during execution of the business process of which it forms a part [WfMC99a].

Manual Activity

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 [WfMC99a].

Process Instance

A process instance is the representation of a single enactment of a process including its associated data [WfMC99a].
It therefore represents an instance of a process definition that includes manual and automated aspects.

Activity Instance

The representation of an activity within a (single) enactment of a process, i.e., within a process instance including its associated data [WfMC99a].

Work Item

The representation of the work to be processed (by a workflow participant) in the context of an activity within a process instance [WfMC99a]. Usually each activity generates one or more work items, which may be presented to a user via a work list.

Invoked Application

An invoked application is a workflow application that is invoked by the workflow management system to automate an activity, fully or partly, or to support a workflow participant during the processing a work item.

3.2 Architecture of Workflow Management Systems

Workflow management systems are typically used in areas such as administration, banking and insurance. However, they have many other application areas including industrial or manufacturing applications.
As a result of multiple application areas, workflow management products vary in their architecture, implementation techniques and specification areas. Despite this variety, all workflow systems show some common characteristics.
Therefore, the Workflow Management Coalition defined the workflow reference model to define a common model for the architecture including components and interfaces.
At the highest level, all workflow management systems may be characterized as providing support in three functional areas:

- The build-time functions: these are used for the definition, the modelling and the analysis of workflow processes and related activities,

- The run-time process control functions: these are concerned with managing the execution of workflow processes,
- The run-time activity interactions: these enable the cooperation with human users and IT applications during the execution of various steps of workflow processes.

Figure 3.3 shows the basic characteristics of workflow management systems and the relationships between these main functions [WfMC95].

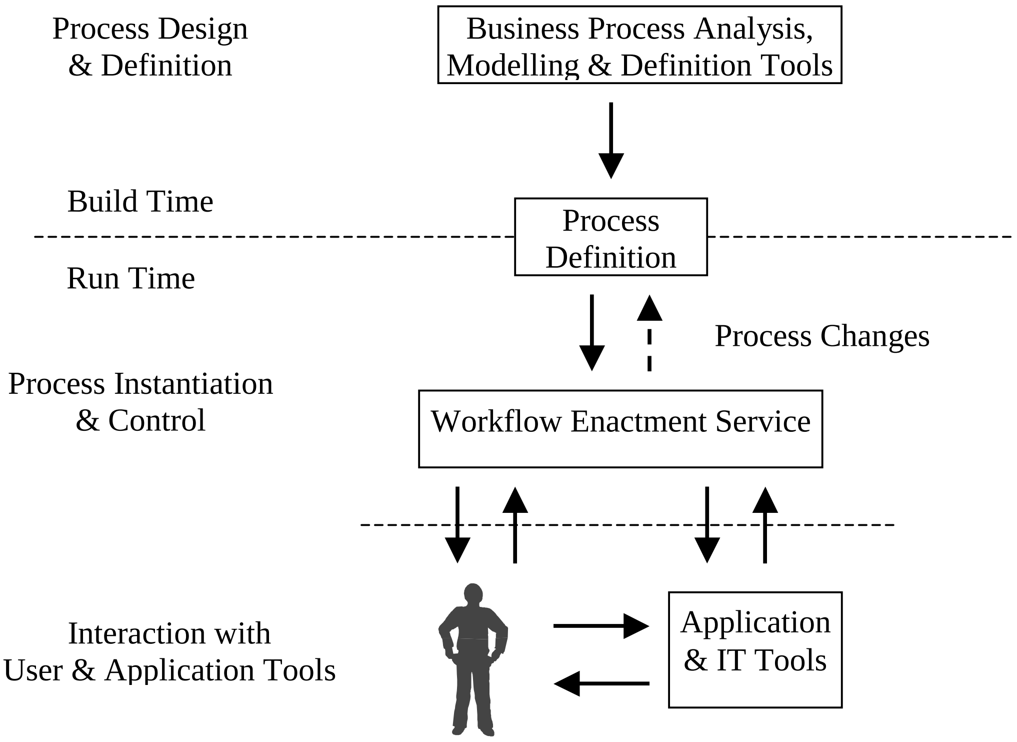

Figure 3.3: Workflow System Characteristics

Build-time Functions

Build-time functions are used to define business processes in a way that can be interpreted and executed by the workflow management system later. Therefore a mapping of a real world process into a definition that can be managed by the workflow enactment service is necessary. A *process definition* is also known as a process model, a process template, or a process metadata. The workflow management coalition prefers the term process definition. A process definition is defined as the computerized representation of a process that includes the manual definition and workflow definition [WfMC95].
The process definition has to comprise all required information for the execution of a process. This includes starting and finishing conditions, constituent activities, rules defining the execution, etc.
The process definition may be given in textual or graphical form or in a formal process definition language depending on the implementation of the definition tool of a specific product. Some workflow management systems have the ability to allow dynamic changes to process definitions at run-time, as shown in Figure 3.3 with the dashed arrow.

It is clear that the definition of a process is a prerequisite for the management of the execution of a process. Thus, build-time functions determine the expressiveness, comprehensiveness and functionality of a workflow management system.

Run-time process control functions

The process definition is interpreted at run-time by the enactment software, which instantiates a process instance and performs the enactment of the functional components in order to execute the process instance. The run-time process control functions are concerned with the mapping of the process as modelled in the process definition and the process in the real world. Therefore run-time control functions have to manage the interactions of users and IT application tools.

The core component of run-time control mechanism is the basic workflow management software, which is also called as workflow engine, responsible for process creation and deletion, control of the activity scheduling within an operational process and interaction with application tools or humans [WfMC95].

Run-time activity interactions

Individual activities, defined as steps of a process, are human- or computer-oriented and usually require the use of some IT tools or programs (e.g. word-processor). For the management of process execution, interactions with the process control software at run-time are unavoidable. These interactions are necessary to transfer control between activities, activate necessary application tools, change data etc…

Therefore, the workflow management coalition is concerned in standardizing interfaces that manage the necessary interactions. Detailed information about these interfaces will be given later in this work.

3.3 Implementation Model of Workflow Management Systems

Despite the differences, workflow management products have some basic implementation structure in common. Thus, the Workflow Management Coalition defined an implementation model of a workflow system that suits the majority of the workflow management products.

The defined implementation model is an abstract model, which defines major functional components of a workflow system and interfaces between them. Consequently, there are several implementation variations of this model. Some vendors may prefer not to implement all the defined interfaces between the functional components. Therefore the Workflow Management Coalition defines different conformance levels to identify supported functions of the implementation model. The main functional components of a generic workflow system are illustrated in Figure 3.4.

The generic model has three types of components [WfMC95]:

- Software components which provide support for various functions within the workflow system (shown in dark fill)
- Various types of process definitions and control data (shown unfilled) which are used by one or more software components
- Applications and application databases (shown in light fill), which are apart of the workflow product but may be invoked by it as a part of the total workflow system

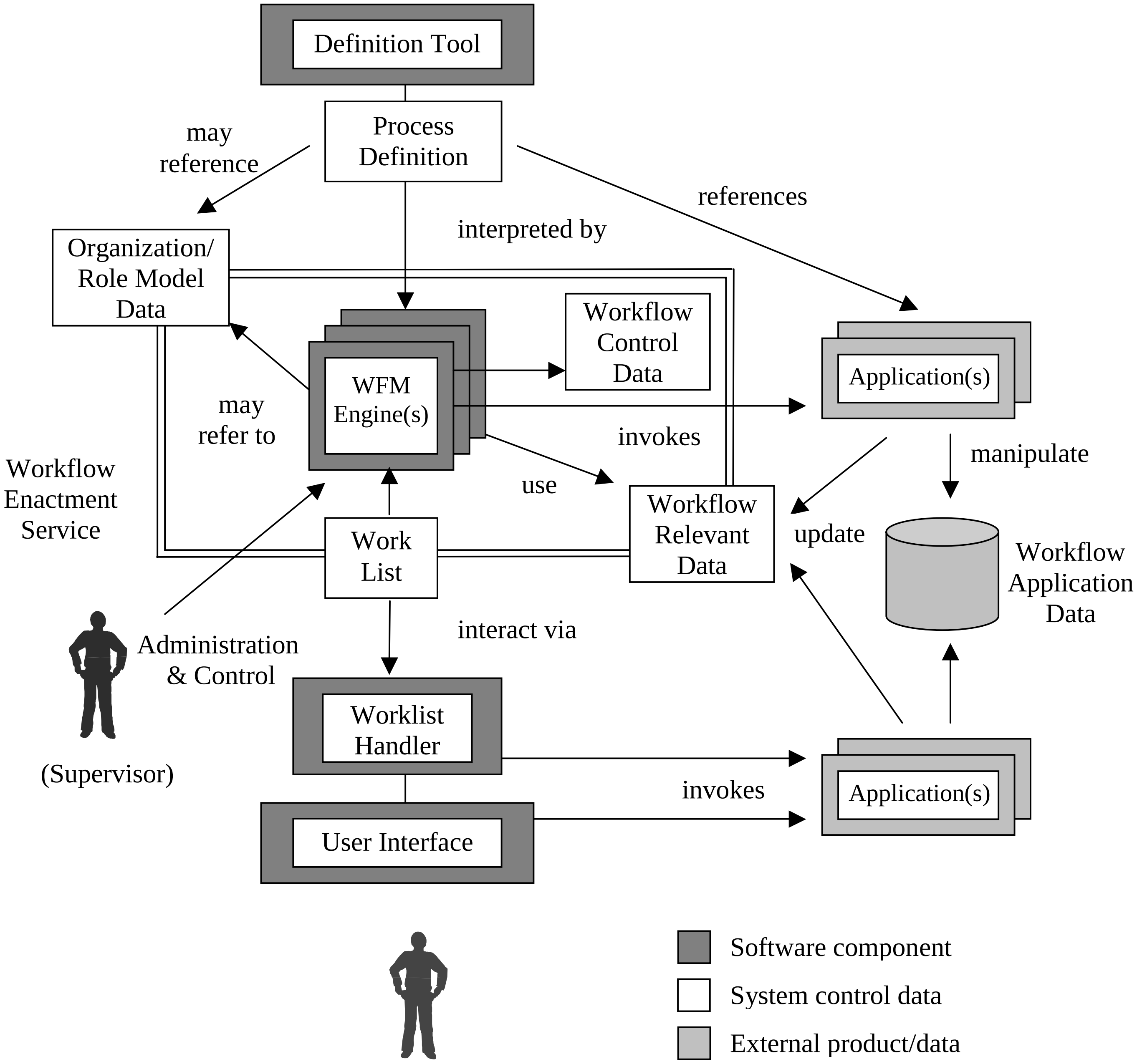

Figure 3.4: Generic Workflow Product Structure

Now some important functional components of the generic model will be discussed in more detail:

Process Definition Tool

The process definition tool enables the transformation of real world processes into a representation form that can be handled by the workflow engine. The representation can be in textual or graphical form.
Here, Petri nets are a well-founded process modelling technique invented by Carl Adam Petri in the sixties [Pet62, Pet81]. Following elements contribute the basic concept of the classical Petri net:

- Places denote the passive part of a process and are represented by circles

- Transitions depict the active part of a process and are represented by rectangles.
- Places and transitions are connected via arcs that are represented as arrows.
- Tokens represent the dynamic part of a process and are represented by points.

Petri nets enable the graphical representation and the analysis of the process. Consequently, Petri nets and advanced Petri nets are implemented by several workflow products for process definition and modelling.

The process definition tool may be part of a workflow management product, part of a business process analysis product or a standalone application from another vendor. If the definition tool is not part of the workflow management product it has to work with, a compatible interchange format must be implemented in both systems.

Workflow Enactment Service

The workflow enactment service consists of one or more workflow engines in order to create, manage and execute particular workflow instances [WfMC99a]. Hereby, each workflow engine is responsible for the execution of a specific process instance.

The task of the workflow enactment service consists of interpreting the process definition, controlling the instantiation of processes, sequencing the related activities, adding work items to the related user work lists and invoking auxiliary application tools. The capability of invoking auxiliary applications is an important facility of workflow engines. Some products are limited to a number of tools whereas others can be extended to invoke a wider range of tools, which may be local or remote to a workflow engine.

The workflow enactment service maintains internal control data that includes state information about processes and activity instances. The workflow control data may also include further information, e.g., information about checkpointing and recovery/restart that are used by the workflow engines to coordinate and recover from failure conditions [WfMC95].

The workflow enactment service gets all necessary information for the execution of a process instance from the process definition and other relevant data. For example, the process definition may refer to an organization-role model that contains information concerning the organizational structure and roles within this structure. The workflow enactment service is responsible for the linking of this information to the participants included in the execution of a process.

Worklists

Where user interactions are necessary within the process execution, the workflow engine(s) places items on worklists for attention by the worklist handler, which manages the interactions with the workflow participants [WfMC95]. In some systems this process is invisible for users, so that users only get the next task provided by the worklist handler. In other systems users can see the whole worklist and choose individual items of work from the list.

Worklist Handler & User Interface (Workflow Client Application)

The workflow handler is a software component that is responsible for the interaction of the workflow enactment service with workflow participants. It provides support for the execution of processes by managing the activities requiring user attention.

The complexity of the worklist handler depends on the specific product. In some systems it may support sophisticated functions such as controlling the allocation of the work between a set of users to provide load balancing and work assignment.

In addition to these worklist handling functions, workflow engines typically support a wider range of client applications, including sign-on and –off of workflow participants, requesting the commencement of an instance of particular process types, requesting work items queued for particular participants, etc… [WfMC95] This wide aspect is the reason why the term workflow client application is preferred and used instead of workflow handler in the reference model.

In Figure 3.4 user interfaces are shown as separated pieces of software. In some systems the worklist handler and the user interface may be integrated into a single software service. In several companies, especially in enterprises working with several software products, standardized interfaces for participants are required. Therefore, it is an important facility for workflow management system products to have user interfaces as separated software components.

Both worklist handler and user interface may be able to invoke appropriate supporting applications. This facility is necessary to support a user in particular tasks to be undertaken. For example, it may be required that every time an employee finishes a required activity, a confirmation message should be sent automatically to his supervisor.

Besides this implementation model, as described by the Workflow Management Coalition, there are also other implementation scenarios for a workflow management system: The structural model of a generic workflow product identifies a series of software components and interfaces. In a concrete product implementation this structure may be realized in a variety of different ways. This is an important area of product differentiation [WfMC95].

Workflow Control Data, Workflow Relevant Data and Workflow Application Data

Information about the internal state of a workflow system with its processes and activity instances is termed as workflow control data. This information is workflow enactment service-specific and not accessible or interchangeable via workflow application programming interface (WAPI) commands. Nevertheless, some of this information may be provided by the workflow enactment service in response to specific requests.

The data, which is generated and updated by workflow application programs, are called workflow relevant data. Contrary to workflow control data, workflow relevant data may be manipulated by both workflow enactment services and workflow applications. Workflow enactment services can access these data in order to make navigation decisions or control operations.

Workflow relevant data is also defined as data that is used by a workflow enactment system to determine the state transitions of a workflow instance, for example within pre- and post-conditions, transition conditions or workflow participant assignment [WfMC99a]. Additionally, workflow relevant data may be transferred between activities by the workflow enactment service.

Workflow application data is application-specific and not accessible by the workflow enactment software. Thus, it can be accessed and manipulated only by the applications.

In case of heterogeneous workflow enactment services, workflow relevant data and workflow application data may be transferred or transformed between workflow enactment services.

Figure 3.5 illustrates the data types and their usage within a workflow management system.

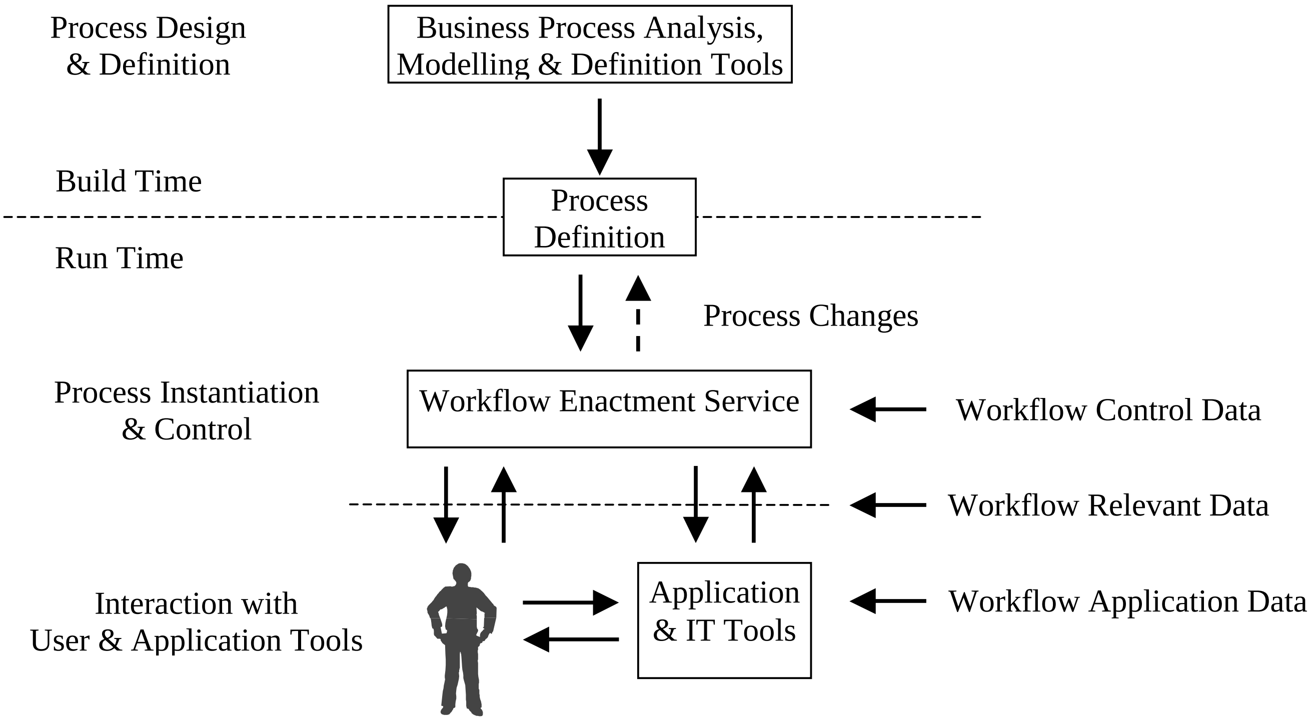

Figure 3.5: Types of Data in Workflow Management Systems

3.4 The Workflow Reference Model

The workflow reference model has been developed by the Workflow Management Coalition to identify the interfaces within the generic workflow product structure. It is clear that different products in the market will have different levels of conformance to these models due to different positioning in the market, especially concerning interoperability.

To enable the cooperation of different workflow management products the definition of interfaces and data interchange formats is of great importance. Figure 3.6 illustrates the workflow reference model that defines the architecture consisting of components and related interfaces among them. These components and interfaces are explained in detail.

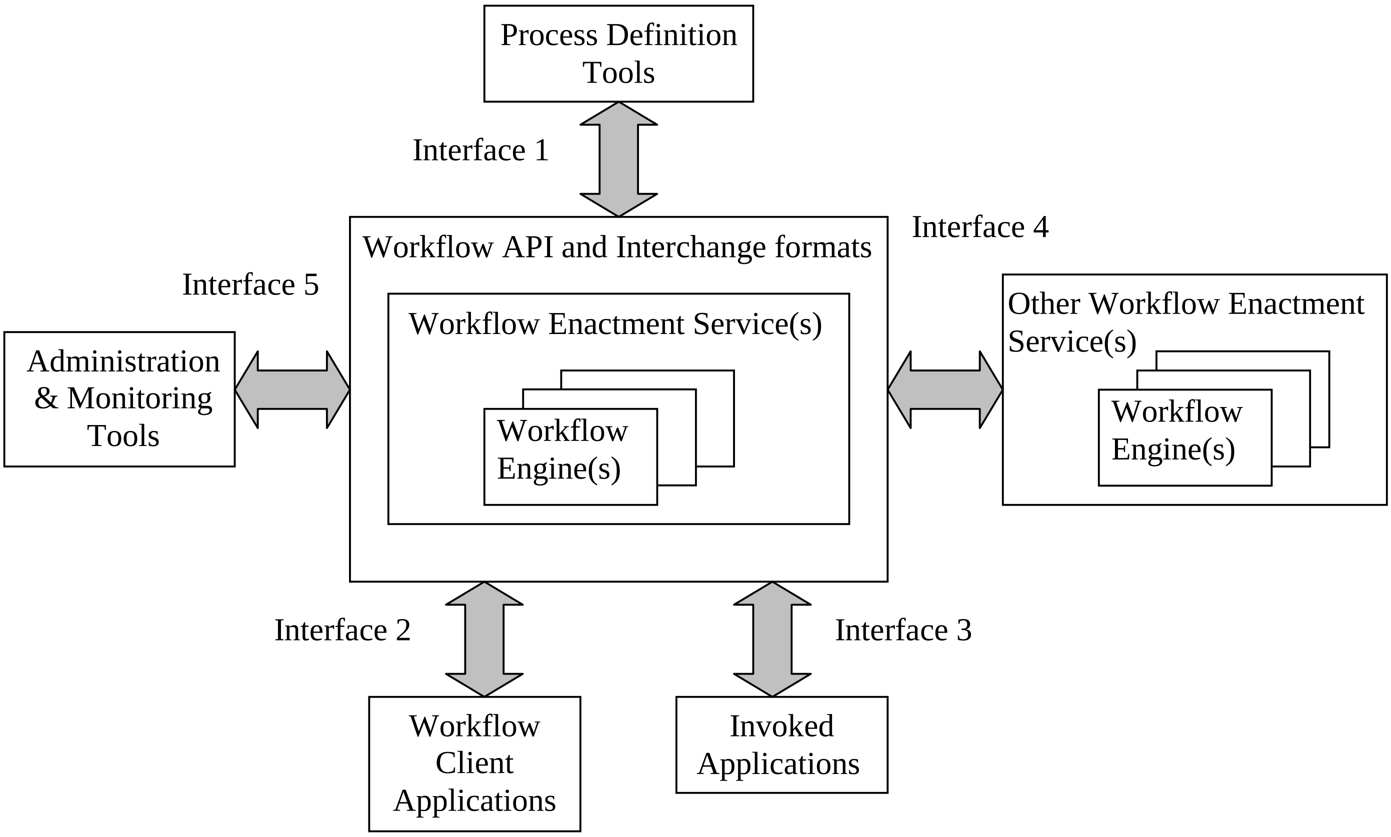
Process Definition Tools
Interface 1
Workflow API and Interchange formats
Interface 4
Interface 5
Workflow Enactment Service(s)
Other Workflow Enactment Service(s)
Administration & Monitoring Tools
Workflow Engine(s)
Workflow Engine(s)
Interface 2
Interface 3
Workflow Client Applications
Invoked Applications

Figure 3.6: Workflow Reference Model – Components & Interfaces

Workflow Enactment Service

A workflow enactment service is a software service that consists of one or more workflow engines in order to create, manage and execute particular workflow instances. Applications may interface to this service via the workflow application programming interface (WAPI) [WfMC99a].
In the reference model, as illustrated in Figure 3.6, there is a separation between the process and activity control service and the application tools and end user tasks, which are required for the execution of the activities. As a consequence interactions between the workflow enactment service and external resources occur with the help of two interfaces:

- The Client Application Interface, also termed Interface 2, which is responsible for the cooperation of workflow enactment service with the worklist handler.
- The Invoked Applications Interface, also termed Interface 3, which supports the workflow enactment service to activate necessary tools to execute an activity.

The workflow management service may be centralized or functionally distributed. In a distributed workflow enactment service several 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 (product-specific) process definition attributes [WfMC95].
A heterogeneous workflow enactment service comprises two or more homogenous services, which follow common standards for interoperability at a defined conformance level [WfMC95].

When heterogeneous products are involved, a standardized interchange is necessary between workflow engines. Via interface 4, the enactment service may transfer activities or sub-processes to other enactment services for execution [WfMC95]. Interface 4 enables the delegation of a particular work, typically a sub-process, to other workflow engines. Transfer of process definition is not part of interface 4's responsibility. The activities and sub-processes, which have been delegated to a remote workflow enactment service for the execution, are pre-defined on that workflow enactment service.

Workflow Engine

A workflow engine is a software service or "engine" that provides the run-time execution environment for a process instance [WfMC99a].
A workflow engine typically executes following tasks [WfMC95]:

- Interpretation of the process definition
- Control of process instances: creation, activation, suspension, termination etc.
- Navigation between process activities, which may involve sequential or parallel operations, deadline scheduling, interpretation of workflow relevant data etc.
- Sign-in and sign-off of specific participants
- Identification of work items for user attention and an interface to support user interactions
- Maintenance of workflow control data and workflow relevant data, passing workflow relevant data to/from applications to users
- An interface to invoke external applications and link any workflow relevant data
- Supervisory actions for control, administration and audit purposes

A workflow engine may be responsible for the whole run-time environment but also for only a part of it. In the latter case several workflow engines constitute the workflow enactment service, where a workflow engine may be responsible only for a specific type of processes.

Workflow Application Programming Interface (WAPI) & Interchange Formats

WAPI is an abbreviation for Workflow API's and Interchange Formats, published by the Workflow Management Coalition, and incorporating specifications to enable interoperability between different components of workflow management systems and applications [WfMC99a].
WAPI may be regarded as a set of API calls and interchange functions supported by a workflow enactment service at its boundary for interaction with other resources and applications. Although this architecture refers to the five interfaces within WAPI, a number of functions within each of these interfaces are common (for example process status calls may be issued from the client application interface or the administration interface) [WfMC95].
Most of the WAPI are APIs with defined parameter and result sets. Furthermore WAPI is usually required to define data interchange formats, e.g., for the interchange of workflow relevant and application data among the interfaces, for the exchange of process definitions, etc…

Now the five interfaces within the WAPI will be described (see Figure 3.6):

Process Definition Tools & Workflow Definition Interchange (Interface 1)

Process definition tools are 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.

Process definition tools have different levels of sophistication. Besides modelling they may include services for the definition and analysis of process models. Another important aspect is the ability to handle organizational data. Organizational data enables the representation of the organization model within workflows. As an example this data may include information about the roles of participants in an organization, hierarchical relationships among them, etc…

Most of the products in the market provide definition tools that can represent the real world processes in a form that can be interpreted only by the specific workflow management product they belong to.

The process definition import/export interface, Interface 1, enables the interchange between a process definition tool and a run-time workflow management software. This interface helps to separate process modelling and process execution responsibilities.

Consequently, implementation of the process definition import/export interface brings two important advantages. First of all, separating of the build-time and run-time environments a process definition can be executed by an arbitrary workflow product implementing this interface at run-time. Thus, it provides the independence of modeling tools and workflow run-time products.

The second advantage of implementing this interface is that 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.

The specification for this interface [WfMC99b] defines a common meta-model for describing the process definition and also a textual grammar for the interchange of process definitions (Workflow Process Definition Language – WPDL) and APIs for the manipulation of process definition data.

Workflow Client Applications & Workflow Client Application Interface (Interface 2)

A workflow client application is an application, which interacts with a workflow engine, requesting facilities and services from the engine. Client applications may interact with a workflow engine for a variety of reasons. Client applications may perform some common functions:

- Worklist handling
- Process instance initiation and process state control functions (e.g. suspend, resume, etc.)
- Retrieval and manipulation of process definition data
- Various system administration functions (e.g., suspending the use of certain process definitions [WfMC 99a])

The worklist handler is a software service that enables interactions for human-involved activities. It may be implemented as part of the workflow management product or as a separate application. Furthermore, it supports the adaptation of user applications to the specific requirements of an enterprise. This enables the enterprise to use different workflow management products with the same enterprise-specific end-user applications.

The workflow client application interface, which is also called Interface 2 in the workflow reference model, enables the access from a workflow client application to the workflow engine and worklist in a product independent manner.

The intended APIs for workflow client application use are grouped into various functional areas as follows [WfMC98]:

- Session establishment
- Workflow definition operations
- Process control functions
- Process status functions
- Worklist/work-item handling functions
- Process supervisory functions
- Data handling functions
- Administration functions
- Application invocation

Implementation of these APIs supports workflow client applications to operate with different workflow engines.

Invoked Applications & Invoked Applications Interface (Interface 3)

An invoked application is a workflow application that is invoked by the workflow management system to carry out an activity by an application, fully or partly, or to support a workflow participant in processing a work-item [WfMC99a].

Application invocation is not a workflow specific functionality but it is of great importance for a workflow management system. 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 OSI TP protocol.

Some workflow products use so-called “Tool Agents” that can handle the application control and information exchange. These tool agents represent at least one specific invocation technology: for instance, some tool agents support MS Windows DDE commands, others can communicate based on protocols like MS OLE or CORBA [WfMC98].

Another possibility of application invocation is the implementation of workflow-enabled applications. Workflow-enabled applications are applications that can interact with the workflow enactment service by using standardized APIs.

The invoked applications interface is defined as a set of APIs to enable the cooperation of workflow enactment service and tool agents or workflow enabled applications. Figure 3.7 illustrates the approach for this interface:

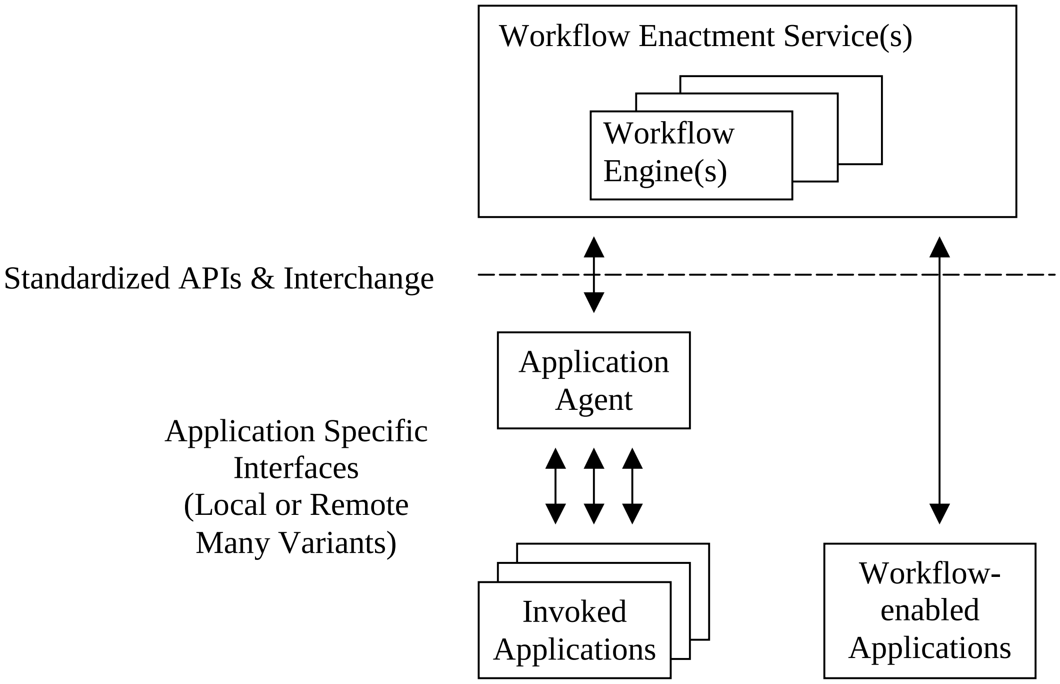
Workflow Enactment Service(s)
Workflow Engine(s)
Standardized APIs & Interchange
Application Agent
Application Specific Interfaces (Local or Remote Many Variants)
Invoked Applications
Workflow-enabled Applications

Figure 3.7: Invoked Application Interface

Application invocation is not limited to local applications as depicted in Figure 3.7. The application to be invoked may be located on the same system as the workflow engine or even on a remote, network accessible system.

As defined in the Workflow Management Coalition's specification on WAPI [WfMC98], the API operates as "calls" at run-time. These API calls should enable the usage of enterprise-specific single end-user interfaces regardless of the number of workflow management products. WAPI calls may be implemented in several languages.

They may be used by workflow applications, such as worklist handlers or other applications, or workflow engines that require interaction with another workflow management product within the context of API functions. The interoperability of different workflow management products will be discussed next.

Workflow Interoperability & Interoperability Interface (Interface 4)

Workflow interoperability is defined 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 [WfMC96].

There is a wide range of workflow management products specialized on different aspects ranging from ad-hoc routing of tasks to regularized production processes in the market. Thus, users of workflow management technology, companies or other organizations planning to introduce this technology, have the chance to choose the product that suits their requirements best.

However, this product specialization can only become an advantage for the user if the workflow management products are able to work together. Therefore, the Workflow Management Coalition defined an interoperability interface supporting simple interoperability scenarios such as the instantiation of a process on a remote workflow engine.

More complex interoperability scenarios, such as the cooperation of different vendor's workflow engines to provide a single workflow enactment service, may be realized in the future.

Different levels of interoperability are defined by workflow management coalition in the interoperability abstract specification [WfMC96]. After the definition of interoperability abstract specification a specification defining an XML-based language, which is designed to model the data transfer, is released [WfMC00].

In Chapter 4 workflow interoperability and the interoperability interface will be discussed in more detail.

Administration and Monitoring Tools & Interface (Interface 5)

Persons responsible for the management of workflow are usually called workflow administrators or system administrators. They use administration and monitoring tools for workflow administration and monitoring purposes.

An administration and monitoring tool may exist as an independent management application interacting with different workflow engines. Besides, they can also be implemented as an integral part of the workflow enactment service with the additional functionality to manage other workflow engines.

The Administration and Monitoring Interface, Interface 5, enables several workflow services to share a range of common administration and monitoring functions. Thus, it includes specific commands within the WAPI set to manipulate designated administration and monitoring functions. This interface is proposed to allow a complete view on the status of the work flowing through the organization, regardless of which system it is in [WfMC95].

The Administration and Monitoring Interface may support the following types of operations (some of which are common to other interface areas) [WfMC95]:

- User management operations
  - establish / delete / suspend / amend privileges of users of workgroups
- Role management operations
  - define / delete / amend role-participant relationships
  - set or unset role attributes
- Audit management operations
  - query / print / start new / delete audit trail or event log, etc.
- Resource management operations
  - set / unset / modify process or activity concurrency levels
  - interrogate resource control data (counts, thresholds, usage parameters, etc.)
- Process supervisory functions
  - change the operational status of a workflow process definition and/or its existent process instances
  - enable or disable particular versions of a process definition
  - change the state of all process or activity instances of a specified type
  - assign attribute(s) to all process or activity instances of a specified type
  - terminate all process instances
- Process status functions
  - open / close a process or activity instances
  - query, set optional filter criteria

- fetch details of process instances or activity instances, filtered as specified
- fetch details of a specific (individual) process or activity instance

At present, most of the workflow management products in the market do not implement all interfaces of the workflow reference model. Usually, they implement some of these interfaces and only some part of the functionality that is defined in the specification.

Chapter 4

The WfMC Interoperability Interface

The rationale of this chapter is to represent the interoperability interface defined by the Workflow Management Coalition in detail.

In the first section the objectives stated for the standardization of the interoperability interface will be presented. The next section will explain different levels of interoperability varying from level one representing no interoperability to level six representing the highest level, where workflow products will provide the same standard user interfaces or at least a common look-and-feel of them. Different models of interoperability is the subject of the third section.

The fourth section will go into more detail and present the operations to be implemented in order to reach the level of interoperability that is currently standardized. Furthermore, sample interoperability scenarios will be introduced to briefly describe the implementation of these operations.

Some case studies, which are essential to evaluate the functionality of the interoperability interface, are subject of the fifth section. First, three processes will be described. In the second part of the fifth section, two interoperability scenarios among these processes will be described.

The last section of this chapter will concentrate on plans to improve workflow interoperability. In this section, beside the future plans of Workflow Management Coalition a further approach to workflow interoperability and its role will be discussed.

4.1 Interoperability Interface

The interoperability interface, also called Interface 4 in the Workflow Reference Model, is designed for the communication and interoperation between workflow engines, which may be instances of the same workflow product or instances of different workflow products.

A key objective of the Workflow Management Coalition is to define standards that will allow workflow systems, henceforth called workflow products from different vendors to pass work items seamlessly between one another [WfMC95].

Today there are several workflow products that focus on different areas of implementation ranging from those used for more ad-hoc routing of tasks or data to those used for more standardized production processes. The implementation of the interoperability interface should allow the workflow product's vendors to continue focusing on particular application areas. This is the first requirement for the interoperability interface in order to enable a workflow product user to benefit from the variety of the products focused on different aspects. Due to the variety of the products in the market, the interoperability interface is standardized to support only simple interoperability operations such as instantiation of a known process definition, querying and changing of the process state and returning the results. The Workflow Management Coalition is working on the improvement of the interoperability interface in order to support more complex interoperability scenarios in the future.

Different interoperability models (scenarios), which can operate at a number of levels, are defined by the Workflow Management Coalition. Next, these levels and models will be described in more detail.

4.2 Levels of Interoperability

Eight levels of interoperability are defined by the Workflow Management Coalition in the Interoperability Abstract Specification [WfMC96]. The levels are distinguished by the architectural and consequential operational characteristics of implementations of workflow engines.

Level 1 – No interoperability

Products that do not support any way of communication with other products have no potential for interoperability and therefore belong to this category.

Level 2 – Coexistence

Products that can coexist on the same platform characterize this level. At this level products share the same run-time environment (hardware, operating system, network). However, there are no direct interactions between different workflow products at this level. Different workflow products may interoperate using WAPI interfaces with the help of human actors.

Gateways

A gateway is a mechanism that allows specific workflow products to move work between each other.

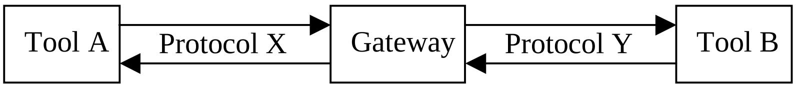

Figure 4.1: Software tools interacting via a gateway that performs protocol transliteration

A gateway may be part of a workflow product or may be a separate product. Gateways are concerned with the transfer of workflow control data and, where necessary, application data between different process instances. If more than two workflow instances are involved, the gateway will also have to perform routing operations. Two levels of gateways are defined:

Level 3 – Unique Gateways

This level is characterized by workflow products working together using some bridging mechanism that performs:

- Routing of operations between workflow engines and instances
- Transliteration and delivery of workflow relevant data

- Transliteration and delivery of workflow application data

Level 3a – Common Gateway API

The level is characterized by workflow products working together using gateways that share a common (standard) API. This level carries the implication that the operations supported by different gateway mechanisms have been normalized to produce a common subset that can be supported by a standard, but does not exclude the possibility of supersets [WfMC96].

Level 4 – Limited Common API Subset

This level is characterized by workflow products that share a common (standard) API that allows them to interact (interoperate) with each other directly in order to move and manage the work between them.

The implementation of this level of interoperability requires that a core set of API function calls are defined in a published standard and that most/all workflow engines can implement that API. The implementation models for this level are quite simple and are based on the use of APIs or encapsulations. (See Figure 4.2)

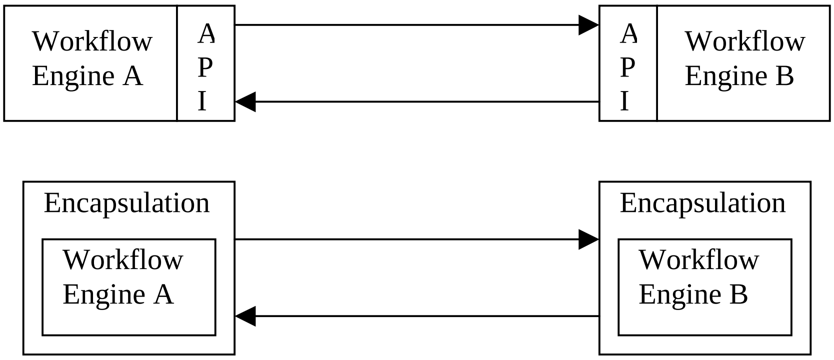

Figure 4.2: Workflow engines interoperating via API calls.
Encapsulated interoperating workflow engines.

The implementation of the encapsulations or APIs will need to handle any necessary data transformations. In order to avoid the need to implement multiple APIs for a given workflow product so that interoperation with different workflow products will be supported, it may be necessary to define neutral information formats to handle the transport of workflow relevant and workflow application data. Each implemented API would then be required to convert to/from the neutral information format [WfMC96].

The Workflow Management Coalition has standardized the neutral information format in a further specification [WfMC00]. However, the semantics of specific elements are not given here. Thus, the appropriate implementation of a neutral information format as defined in the specification [WfMC00] is not enough to achieve workflow interoperability. For this reason, the Workflow Management Coalition is recommending that an interoperability contract to be established among vendors participating in interoperable workflows. Each interoperating vendor must ensure that all factors impacting their implementation is defined clearly and completely in the interoperability contract. Some of the topics that should be considered in the interoperability contract can be: Data requirements, data constraints, error handling, transport protocol limitations, security consideration etc [WfMC00].

Level 5 - Complete Workflow API

This level is characterized by all workflow products sharing a single standard API that gives access to the full range of possible operations by any workflow management system. This excludes any domain specific functionality that might be offered by workflow products developed to address the needs of particular market segments [WfMC96].

In case of a complete workflow API all operations of a workflow management system can be directly accessed by another system. In order to create a complete API set, all workflow products on the market have to be investigated so that an intersecting set of operations that can be supported by all products can be defined. It is clear that this level can only be reached through continuous development at level four.

First, a set of common operations is defined at an abstract level. Afterwards, each workflow product's vendor, supporting this level of interoperability, must map these operations to his specific products operations.

Level 6 – Shared Definition Formats

This level is characterized by different workflow products having a shared format for process definitions that covers routing decisions; user access rights and the maintenance of workflow system resources. The consequence of this is that an organization can produce a single definition for each process that is to be supported on a workflow system, and can guarantee the behavior of the process whatever the workflow engine used to enact it [WfMC96].

There are several products on the market that are specialized in different application areas. Thus, it is likely that all workflow products will not be able to support all possible operations. The most probable way to achieve this level of interoperability is that generic functions will be supported by all workflow products whereas specialized functions, which depend on the application area they belong to, will be supported only by certain workflow products. Considering the solution to support this level of interoperability, there are two steps to define a standard:

1. The definition of the generic set of functionality
2. The definition of operational profiles for different classes of workflow products in order to identify those that can be used to provide specific functionality

The Workflow Management Coalition is currently defining a vendor-independent process definition language (WPDL) in order to take the first step. Later additional functionalities could be offered using an extended language definition.

The implementation of this level will not only enable a single process definition to be enacted on a variety of workflow engines, but also modular parts of this definition to be enacted on different workflow engines. This ability of processing the work on different workflow products will result in more flexibility and higher efficiency.

The workflow definition language will enable the definition of processes in a standardized format, which will be mapped or transliterated to the specific process definition of an individual workflow product. Thus workflow products conforming to this level of interoperability will implement import and export interfaces, which have to enable the translations between the standard and product-specific process definitions.

Level 7 – Protocol Compatibility

This level assumes that all API client/server communication including the transmission of definitions, workflow transactions and recovery is standardized. To achieve this level of interoperability, vendors may be required to support a number of different mechanisms through which such interoperation can be effected [WfMC96].

The implementation of this level would enable to secure the consistency of the communication and therefore of the interoperability scenario.

At this point, it must be mentioned that the Workflow Management Coalition's has defined a further level of interoperability, in which all workflow products may present a standard user interface or at least a common look-and-feel, in the specification [WfMC96]. However, it has been appreciated by the Workflow Management Coalition that most probably this level will never be reached.

4.3 Models of Interoperability

Three interoperability models, covering defined interoperability levels, are identified by the Workflow Management Coalition's Specification [WfMC96]:

1. Chained Process Model

This model of interoperability assumes that a process instance enacted on workflow engine A triggers the creation and enactment of a sub-process instance on workflow engine B [WfMC96]. Thus it supports the transfer of a single item of work (a process instance or activity) in a distributed environment.

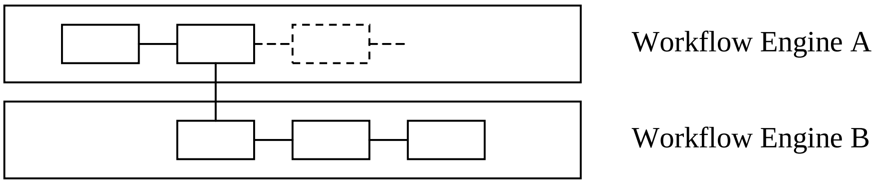

Figure 4.3: The chained model of interoperability

The newly created sub-process instance on workflow engine B operates independently in its own environment without any synchronization. After the enactment of the sub-process on workflow engine B, the process instance on workflow engine A may terminate or continue. Therefore, the process instance on workflow engine A and the sub-process instance on B may be executed concurrently.

2. Nested Sub-Process Model

The nested sub-process model of interoperability assumes that a process instance enacted on a workflow engine causes the creation and enactment of a sub-process instance on a second engine and is blocked until the termination of its sub-process. After the termination of its sub-process it can carry on with its enactment [WfMC96].

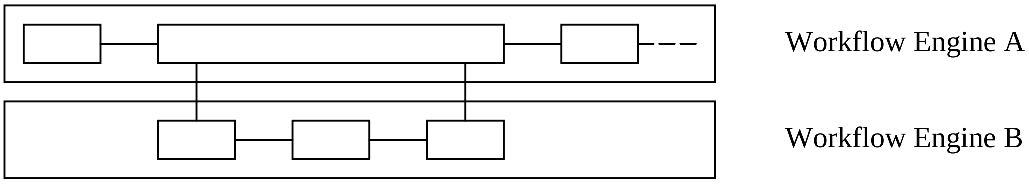

Figure 4.4: The nested sub-process model of interoperability

This model allows a process executed in a particular workflow domain to be completely encapsulated as a single task within a superior process executed in a different workflow environment.

3. Parallel-Synchronized Model

The parallel-synchronized model of interoperability assumes that two workflow engines simultaneously enact process instances and that at some point in the definition of each of the process instances a rendezvous has been specified. Having achieved the rendezvous point, the first workflow engine (which engine is first is not specified) waits for the other to reach its rendezvous point. Once the enactment of both process instances has attained the respective rendezvous points, there is some (unspecified) interchange between the workflow engines, before both continue with the enactment of their respective process instances [WfMC96].

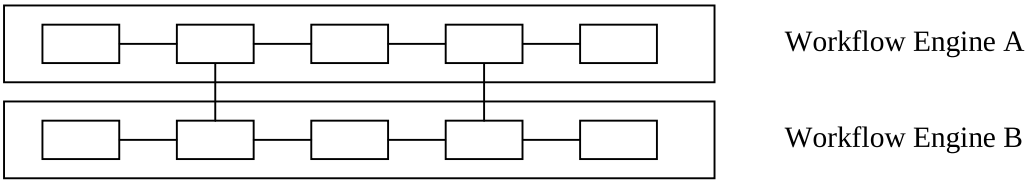

Figure 4.5: The parallel synchronized model of interoperability

The parallel-synchronized model of interoperability is outside the scope of what the Workflow Management Coalition’s standard is currently trying to achieve [WfMC00].
Thus, it is not possible to implement more sophisticated interoperability scenarios, where two processes can exchange information and be executed in parallel. For several cases in practice, chained process and sub-process models are sufficient, however there are also situations where the parallel-synchronized model is the only way of implementation.

4.4 Interoperability Level to be Implemented

The Workflow Management Coalition has released a specification (Wf-XML) to support the interoperability level four, Limited Common API Set, as defined in Section 4.2. It specifies an XML document definition designed to model the data transfer requirements [WfMC00]. Moreover, the specification aims at utilizing the extensible markup language (XML) to define a language, using tags as vocabulary, with which workflow systems will interoperate. Two basic models of interoperability, namely simple chained process model and nested sub-process model are supported both synchronously and asynchronously.
The specification describes a language that is independent of any particular implementation mechanism, such as programming language, data transport mechanism, platform, hardware etc… However, because of the fact that HTTP is considered as the most important data transport mechanism for Wf-XML, the specification provides a description of how Wf-XML interchanges are transferred using this protocol [WfMC00].
In general, security considerations are out of the scope of the specification because they are largely dependent upon the transport mechanism used by an implementation. This applies to user identification and authorization, encryption, and data/functional access control. Often, security mechanisms such as SSL (Secure Socket Layer) and LDAP (Lightweight Directory Access Protocol) are considered to be sufficient for some applications but insufficient for some others. Therefore, the security mechanism used by two or more interoperating services should be identified in the interoperability contract between them [WfMC00].

An individual interoperable function is termed as an *operation*. Operations are capable of receiving a set of request parameters and returning a set of response parameters. Operations are divided into groups to identify their context better: The primary operation groups required for interoperability are named *ProcessDefinition*, *ProcessInstance* and *Observer*. A resource may implement one or more groups of operations by supporting the operations defined to exist within that group. The table in Figure 4.6 shows briefly the groups and their operation(s) in the specification:

	Process Definition	Process Instance	Observer
CreateProcessInstance	**X**		
ChangeProcessInstanceState		**X**	
GetProcessInstanceData		**X**	
ProcessInstanceStateChanged			**X**

Figure 4.6: Operation groups collecting operations

The *ProcessDefinition* group is the most fundamental group of operations representing the description of a service's most basic functions and is the resource from which instances of a service will be created. The process definition provides a unique resource identifier for an interoperating service or service requestor. The resource identifier can be used to reference the desired process to be executed.

To clarify the implementation of the operation groups we will use a practical example (the automobile manufacturing process): The automobile manufacturing process requires the collaboration of a car manufacturer and several supplier companies. This can be achieved through the interoperability of the related workflow engines.

To start the interoperability scenario, the car manufacturer has to instantiate a defined process on the supplier's workflow engine. Here, the car manufacturer's workflow engine makes use of the *CreateProcessInstance* operation that exists within the *ProcessDefinition* group.

The *ProcessInstance* group represents the actual enactment of a given process definition and will have its own identifier other than the process definition's. With the reference to a unique resource identifier, a requestor may create several instances of a process definition. Each process instance will have its own resource identifier. Therefore, the requestor may operate on a specific instance. At this point, the requestor may change the state of an instance or require its data.

Considering the automobile manufacturing process example, after the instantiation of a process definition on the supplier's workflow engine, the car manufacturer's workflow engine may request the process specific data at any time using the *GetProcessInstanceData* operation. In addition, the car manufacturer may use the *ChangeProcessInstanceState* operation to change the state of the supplier's process instance. All operations related to a unique process instance are collected in the *ProcessInstance* group.

The ability of a process instance to communicate with its parent process instance is in many interoperability scenarios of major importance. This ability of a process instance informing its parent process instance corresponds to the relationship between a source and its observers as defined in the observer design pattern in [GHJV95]. Therefore, the operations enabling a process instance to communicate with its parent process are collected in a group named *Observer*.

The *Observer* group provides a means by which a process instance may communicate its completion or termination. In the nested sub-process model, the requestor must be able to determine or be informed when a sub-process completes. The *Observer* group will provide this information by giving a process instance the resource identifier of the requestor.

Considering an automobile manufacturing process, and the car manufacturer would be interested in getting informed by the supplier when the process execution is completed. In order to do this, the supplier's workflow engine has to register the resource identifier of the car manufacturer's workflow engine. If the process instance on the supplier's engine is completed or terminated, it will inform the car manufacturer's workflow engine by using the registered resource identifier in a *ProcessInstanceStateChanged* operation, which exits within the *Observer* group.

The following figure indicates the relationship between each group of operations described above [WfMC00]:

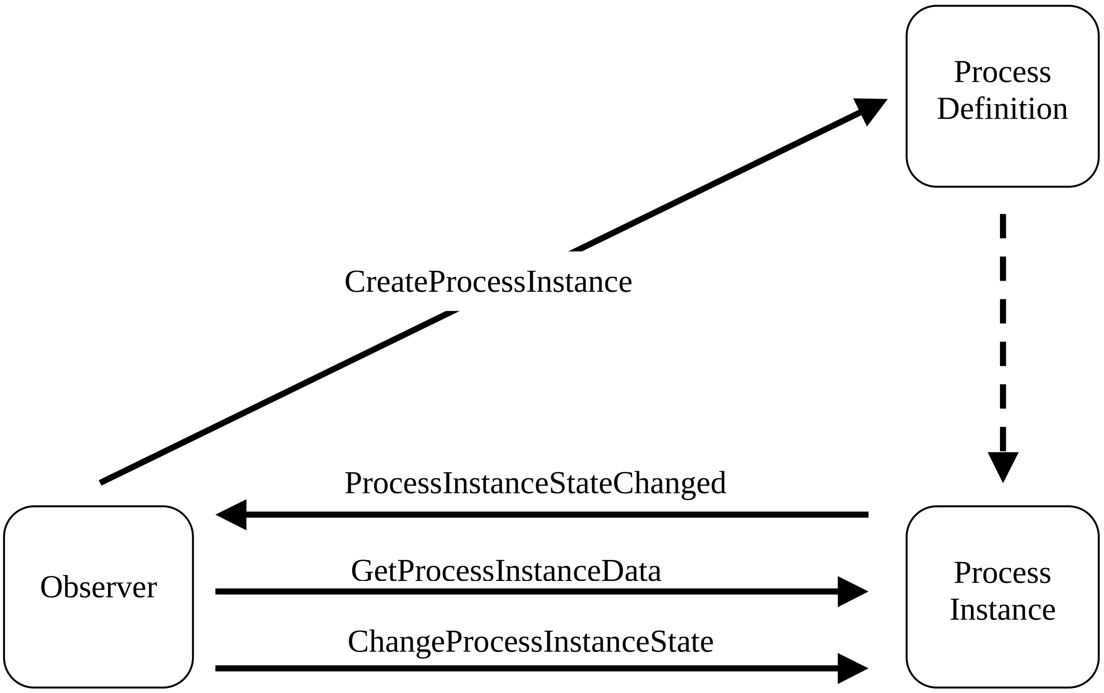

Figure 4.7: Interoperability Subjects and their respective Operations

General Structure of Messages

Every Wf-XML message defined in the specification is an XML document. In order to provide clarity and precision each Wf-XML message contains a XML declaration as follows: '<?xml version="1.0"?>'

- The root element of a Wf-XML message is named "WfMessage". This element carries a required attribute version, which indicates the particular version of the specification with which the message conforms. Each Wf-XML Message contains an optional section for transport-specific information (WfTransport), a single, required message header (WfMessageHeader), and a single, required message body (WfMessageBody).

- If necessary, the WfTransport section can be used to convey transport protocol-related information such as message security, batch or asynchronous processing, message identification, etc…
- The message header contains information relevant to routing and preprocessing of the message.
- The message body carries the operation specific information.

Consequently, the skeleton of a Wf-XML Message is as follows:

```
<?xml version=”1.0”?>
<WfMessage Version=”1.0”>
        <WfTransport/>
        <WfMessageHeader>
                …
        </WfMessageHeader>
        <WfMessageBody>
                …
        </WfMessageBody>
</WfMessage>
```

The specification contains a document type declaration (DTD) for the purposes of implementation reference and optional data validation by an XML processor [WfMC00]. Here, the specification does not require validity of all document instances, it only requires that all Wf-XML messages be well-formed.

Furthermore, the specification recommends the usage of namespaces for the interchange of process specific data. Using namespace declarations, applications will be able to distinguish elements defined by the specification from the other defined elsewhere, in order to achieve higher levels of interoperability without degrading conformance to the specification.

Representation of Process-Specific Data

A process is usually associated with data items, which may be workflow control data, workflow relevant data or application data. The process-specific data is called the *context of the process* in a request message and the *result of the process* in a response message. When a process is enacted, these data items must be specified and accessible.

The specification provides a place to identify these data items in the form of elements named *ContextData* and *ResultData*. These elements are placed inside the message body (WfMessageBody element). When a process is instantiated, the process instance is initialized with the contents of the *ContextData* element. When a process instance is completed, the resulting data is exchanged as the contents of the *ResultData* element.

Because of their process-specific nature, *ContextData* and *ResultData* must be defined for each implementation separately. As a placeholder for extensibility in this area, the WfMC specification defines a default content model of “ANY” for these elements. The two parties, which will participate in the interoperability scenario, should specify the content of these elements in the interoperability contract. Therefore, they have to extend the specification to meet their specific needs.

The Workflow Management Coalition is proposing three different ways to extend the specification, depending on the requirements of a given implementation:

- If full validity is required, the interoperating parties should agree on the necessary changes to the content models of the *ContextData* and *ResultData* elements and provide an extended DTD in their interoperability contract, against which their Wf-XML instances can be validated. If context and result data vary for each given process definition, separate DTDs may be required for each defined process in order to support validation. These changes will not affect conformance to the specification, as they are anticipated extensions.

- If well-formedness of the XML message is sufficient, interoperating parties can simply agree in their interoperability contract on the markup to be exchanged within the content elements, leaving the DTD declarations unchanged.
- Last, interoperating parties may exchange context information conform to an external DTD or schema using namespace declarations to reference that external source. This mechanism most easily allows for the utilization of industry standard markup within Wf-XML messages [WfMC00].

Sample Interoperability Scenarios

After the introduction of the basic terms in the Workflow Management Coalitions interoperability specification, two sample interoperability scenarios are used to illustrate the possible implementations of the standardized operations.
A typical interoperability scenario, which is an implementation of the nested sub-process model, may be achieved with the following sequence of synchronous and asynchronous Wf-XML messages (see Figure 4.8):

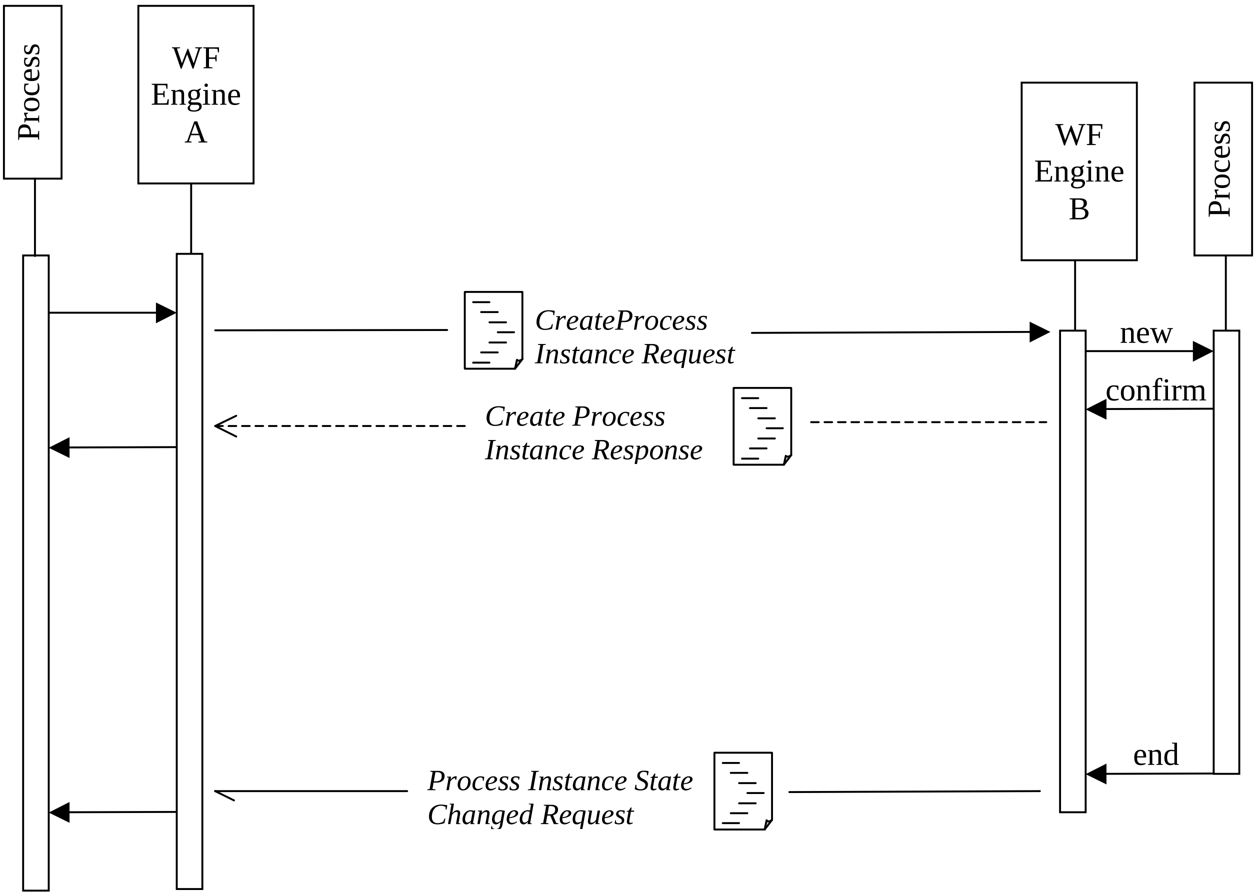
Process
WF Engine A
WF Engine B
Process
CreateProcess Instance Request
new
confirm
Create Process Instance Response
end
Process Instance State Changed Request

Figure 4.8: An interoperability scenario with synchronous and asynchronous Wf-XML messages

Create Process Instance Request

In this scenario the process instance on workflow engine A instantiates and starts a process on workflow engine B with a *create process instance request* message.
The header part of this message contains the information indicating that a response message is expected from workflow engine B after the instantiation. Besides this, the message body contains also an XML element named *Key* that indicates the URI (Uniform Resource Identifier) of the process definition on workflow engine B.
The presence of the XML element named *ObserverKey* in the message body indicates that the nested sub-process interoperability model will be used. The *ObserverKey* element contains the URI of the resource that is to be the observer of the instance on workflow engine B, in this case the URI of the process instance on workflow engine A. This observer resource is to be notified of the completion of the process instance on workflow engine B.
Apart from the *ObserverKey*, which must be present for this scenario, the message body includes the data, encapsulated in the XML element named *ContextData* in the specification, that are necessary to instantiate the process definition on workflow engine B.

Omitting the asynchronous *process instance state changed* message, the interoperability scenario in Figure 4.8 constitutes an example for the chained process model. Here, after the instantiation of the chained process on workflow engine B, the workflow engine A continues with the execution of its own process and does not wait for an answer message from workflow engine B.
Contrary to the nested sub-process interoperability model, the absence of the *ObserverKey* in the *create process instance request* message indicates that the chained process interoperability model is implemented.

Create Process Instance Response

After receiving the *create process instance request* message, the workflow engine B generates an instance of the process definition with the data encapsulated in the *ContextData* element and registers the URI of the observer process instance, in this case the URI of the process instance on workflow engine A. Then the workflow engine B generates and sends a *create process instance response* message and continues with the execution of the process instance.
Here, the message body includes an XML element named *ProcessInstanceKey,* which contains the URI of the newly created process instance on workflow engine B.
In case an exception or error occurs, the message body contains an exception element revealing information about the exception.
In case no exception has occurred, the *create process instance response* message received by the workflow engine A acknowledges that the create process instance operation has been successful. The workflow engine registers the URI of the newly created sub-process instance that exits on workflow engine B and waits until another message indicating the completion or termination of this instance arrives.

Process Instance State Changed Request

When the process instance is completed or terminated, the workflow engine B sends an asynchronous *process instance state changed* request message by using the URI of the process instance on workflow engine A, which has been registered before. The body of this message includes three important elements. The *ProcessInstanceKey* element indicates the URI of the process instance on the workflow engine B, so that workflow engine A can identify it. The *State* element includes information about the final state of the process instance, which may be either *completed* or *terminated*. The *ResultData* element encapsulates the resulting data of the process instance.

After receiving this message workflow engine A can use the result data and continue with the execution of its own process instance. The workflow engine A could generate a *process instance state changed* response message, which has no parameters, to acknowledge the successful receipt of the *process instance state changed* message.

Application of this Interoperability scenario in an automobile manufacturing process

The automobile manufacturing process requiring the collaboration of the automobile company and the supplier companies is an appropriate application of this interoperability scenario.

The process on workflow engine A represents the manufacturing process of a car manufacturer, whereas the process defined on workflow engine B represents the production process of a supplier company that produces certain automobile parts for this car manufacturer.

Considering the automobile manufacturing process example, the interoperability scenario in Figure 4.8, at a certain point of its manufacturing process the automobile company orders the start of the production process by the supplier company. The order request (*create process instance*) contains all necessary data about the production process. For example: names of the parts, types of the parts, amount of production, deadline of production etc…

After receiving the request, the supplier company starts the production process, registers the identification of the car manufacturer and sends an acknowledgement (*response*) including the unique identification of the production process.

The acknowledgement serves the car manufacturer as conformance for the successful start of the production process by the supplier company.

When the production process finishes, the supplier company informs the car manufacturer with a notification (*process instance state changed*) indicating the resulting data of the finished production process, which may include detailed information about the production process. For example: number of parts, production time, date of delivery etc…

The car manufacturer receiving this information may use it to make decisions on the production of other automobile parts and arrange its own production.

This scenario assumes that the cooperation of the car manufacturer and the supplier company has been successful without any problems. If the production processes in both companies are defined and implemented optimally, the scenario in Figure 4.8 would be used in most cases. However, regardless of good design and implementation, some expected or unexpected problems may still occur. For example: the car manufacturer may want to terminate or suspend the production of the supplier company due to external reasons such as change in the deadlines, delays or problems in its own production, etc… Then an alternative interoperability scenario is applied: Such an interoperability scenario, which is an application of the nested sub-process model, may be achieved with the following sequence of synchronous and asynchronous Wf-XML messages as illustrated in the next figure:

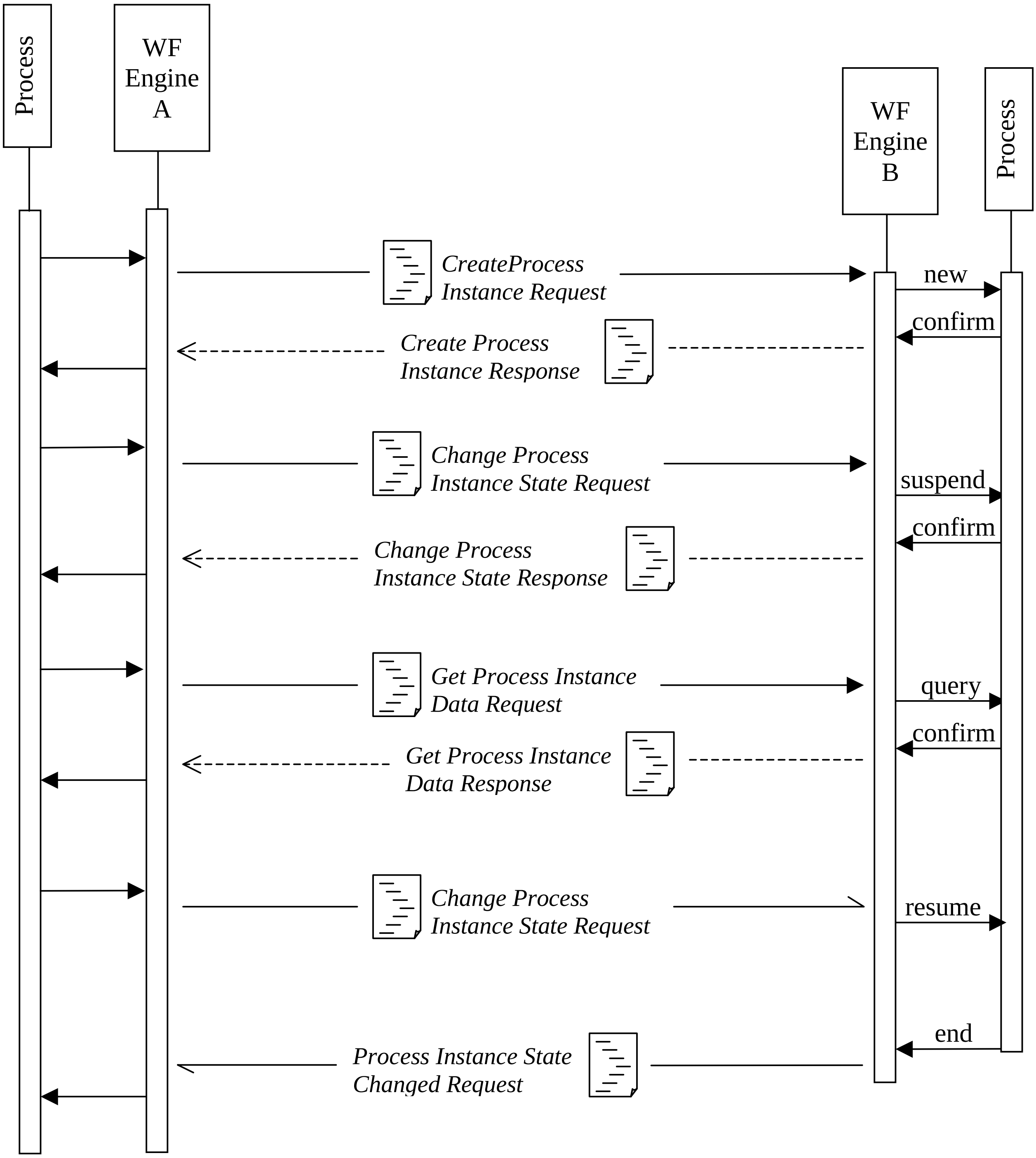
Process
WF Engine A
WF Engine B
Process
CreateProcess Instance Request
new
confirm
Create Process Instance Response
Change Process Instance State Request
suspend
confirm
Change Process Instance State Response
Get Process Instance Data Request
query
confirm
Get Process Instance Data Response
Change Process Instance State Request
resume
end
Process Instance State Changed Request

Figure 4.9: An alternative interoperability scenario with synchronous and asynchronous Wf-XML messages

Create Process Instance Request

Similar to the first scenario, the scenario in Figure 4.9 begins with a *create process instance request* message sent by workflow engine A to instantiate and start a process on workflow engine B.
Here, the *create process instance request* message involves three important elements: *Key* indicating the URI of the process definition on the workflow engine B, *ContextData*

encapsulating the necessary data for the instantiation, and *ObserverKey* including the URI of the process instance on workflow engine A.

Create Process Instance Response

After receiving the *create process instance request* message, the workflow engine B generates an instance of the process definition with the data encapsulated in the *ContextData* element and registers the URI of the process instance on workflow engine A.

To allow this, the message body includes an XML element named *ProcessInstanceKey*, which plays a major role for this scenario. The *ProcessInstanceKey* element contains the URI of the newly created process instance on workflow engine B.

The workflow engine A registers the URI of the newly created process instance and waits for an asynchronous message from the workflow engine B indicating the completion or the termination of the process instance and the resulting data.

Change Process Instance State Request

In this scenario, after a period of time the workflow engine A suspends the process instance. It sends a *change process instance state* message. This message's header contains the XML element *Key* that indicates the URI of the process instance on workflow engine B.

Besides, the message body contains an element named *State* indicating the desired state of the process instance, which is *open.notrunning.suspended* in this case.

Change Process Instance State Response

The workflow engine B, when receiving the *create process instance request* message, changes the current state of process instance to the required state. Then it sends a *create process instance state response* message indicating the new state resulting from the operation, in this case *open.notrunning.suspended*. Consequently, this message confirms that the *change process instance state* operation has been executed successfully.

Get Process Instance Data Request

After some time the superordinate process on workflow engine A needs to retrieve some data from the sub-process instance that is suspended. Depending on process-specific data, some of the required information may be incomplete or inconsistent at the time. The superordinate process on workflow engine A must be able to evaluate these data. An XML element named *ResultDataSet* is defined in the specification, which contains a list of data to be returned, where the list can contain all of the data or a subset of all data. If not specified, all data are returned.

The workflow engine A requests the necessary set of the data with a *get process instance data request* message. Here, the *Key* element in the message header indicates the URI of the process instance.

Get Process Instance Data Response

The workflow engine B receiving the *get process instance data* message retrieves the current set of result data, encapsulates it in the *ResultData* element of a *get process instance data response* message and sends the message to workflow engine A.

Change Process Instance State Request

Later in the scenario, the workflow engine wakes up the suspended process instance on workflow engine B. Therefore, it sends another *change process instance state request* message with the *State* element indicating the required state: *open.running*.

The workflow engine B receiving this message wakes up the suspended process instance.

This message could require a *change process instance state response* message for confirmation, which was left out in this scenario.

Process Instance State Changed Request

When the created sub-process instance is completed or terminated, the workflow engine B sends an asynchronous *process instance state changed* request message. Similar to the first scenario the *ResultData* element in the message body encapsulates the resulting data of the process instance.
After receiving this message, workflow engine A can use the resulting data and continue with the execution of its own process instance.

Application of the alternative scenario in the automobile manufacturing process

Considering the automobile manufacturing process described above, the alternative interoperability scenario in Figure 4.9 starts like the scenario in Figure 4.8: The car manufacturer starts the production process in the supplier company, that produces certain automobile parts for this car manufacturer. In particular, the car manufacturer's order request includes necessary data for the supplier company such as name and type of the parts, amount of order, deadline etc…
Receiving the order request, the supplier company starts with the production process, and sends a confirmation back to the car manufacturer.
After some time due to problems such as delays in the production or change of the deadlines the car manufacturer decides to temporarily stop the production of parts. Thus, it sends a request (*change process instance state*) to the supplier company.
Later the car manufacturer requests information about the result of the production (*get process instance data*). The supplier company sends this information.
The car manufacturer receiving the information can use it for modification of the manufacturing process. For example: The company could specify new deadlines for the production of other parts, redefine the amount of production, etc.
After the modifications, at some point of time the car manufacturer resumes the stopped production process (*change process instance state*).
When the production process finishes, the supplier company notifies the car manufacturer indicating the data of the finished production process.

4.5 Case Studies

In this work, the interoperability interface, as defined by the WfMC, will be implemented and evaluated. The evaluation will be based on three sample processes in two interoperability scenarios, similar to the interoperability scenarios defined in Section 4.4. The processes and interoperability scenarios will be described next.
Since the sequence of messages for the nested sub-process interoperability model includes those for the chained process interoperability model, a separate sample scenario for the chained process interoperability model will neither be defined nor tested.

4.5.1 Sample Processes

In this section, three sample processes from the marketing area will be defined. The main reason for choosing the test processes from this area is the availability of the process specific information. Moreover, the application area of a process determines the content of its process specific information, but has no effects on the functionality of the interoperability interface.

The Market Research Process I

When a company plans to launch a new product, the first step to take is most probably to make a comprehensive study of the related market segment. Only such a study enables a successful estimation of the related risks and chances of the planned investment.

Therefore, the research of the market should bring answers to questions such as: Which products do exist in the market, how are they presented in the market, how well-known are the present brands, what are the consumer opinions about these products, what is the degree of consumer satisfaction with these products etc...

Figure 4.10 illustrates the activities of a market research process distributed among different departments. First, the marketing department investigates the representation of the existing products in several media: Newspapers, magazines, radio, TV, Internet, etc…

Second, the operations department visits different stores to study the representation of the products in the market. These stores may be drugstores, supermarkets, pharmacies, etc. in different cities and/or in different countries.

Next, the marketing department checks the awareness of the consumers through questionnaires, workshops, telephone interviews, etc.

Finally, the administration department prepares a final report, which summarizes the results of the prior activities to represent the structure of the related market segment.

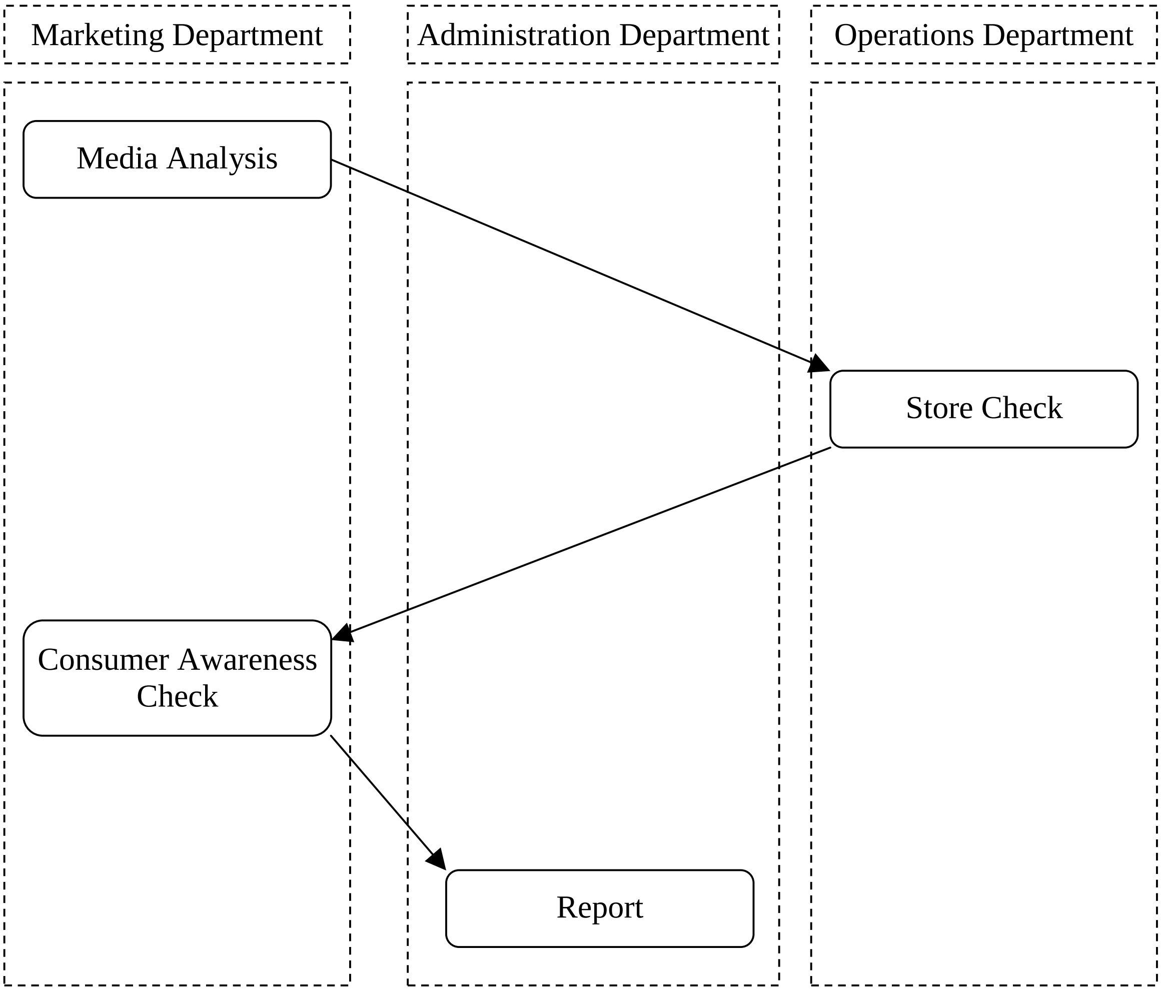

Figure 4.10: The Market Research Process I

The Market Research Process II

In some cases the consumer company ordering the market research may want to modify the content of the research depending on some factors, for example on the results of the media analysis. Figure 4.11 illustrates another market research process that is similar to the market research process I except its ability for modification.

The market research process II also starts with a media analysis, which is executed by the marketing department. Second, the operations department starts the store check operation.

Depending on the results of the media analysis, the customer company may have additional requirements. Thus, the administration department, which is responsible for the finance and management of the process, checks what additional requirements must be fulfilled. For example: the customer company may require the market research to be focused more on the name recognition of the existing products. In particular, the scope and the budget of the consumer awareness check operation must be extended. As a result, the partition of the overall budget for the market research process must be reconsidered and modified.

To rearrange the process cost, the administration department requests the operations department to report the costs so far.

Consequently, the operations department suspends the operation and informs the administration department about the costs. Considering the budget and the requirements, the administration department modifies the partition of the overall budget and orders the operations department to resume with the execution of the store check operation.

After ending the store check operation, the marketing department starts with the modified consumer awareness check operation.
Finally, the administration department prepares a final report representing the results of the market research.

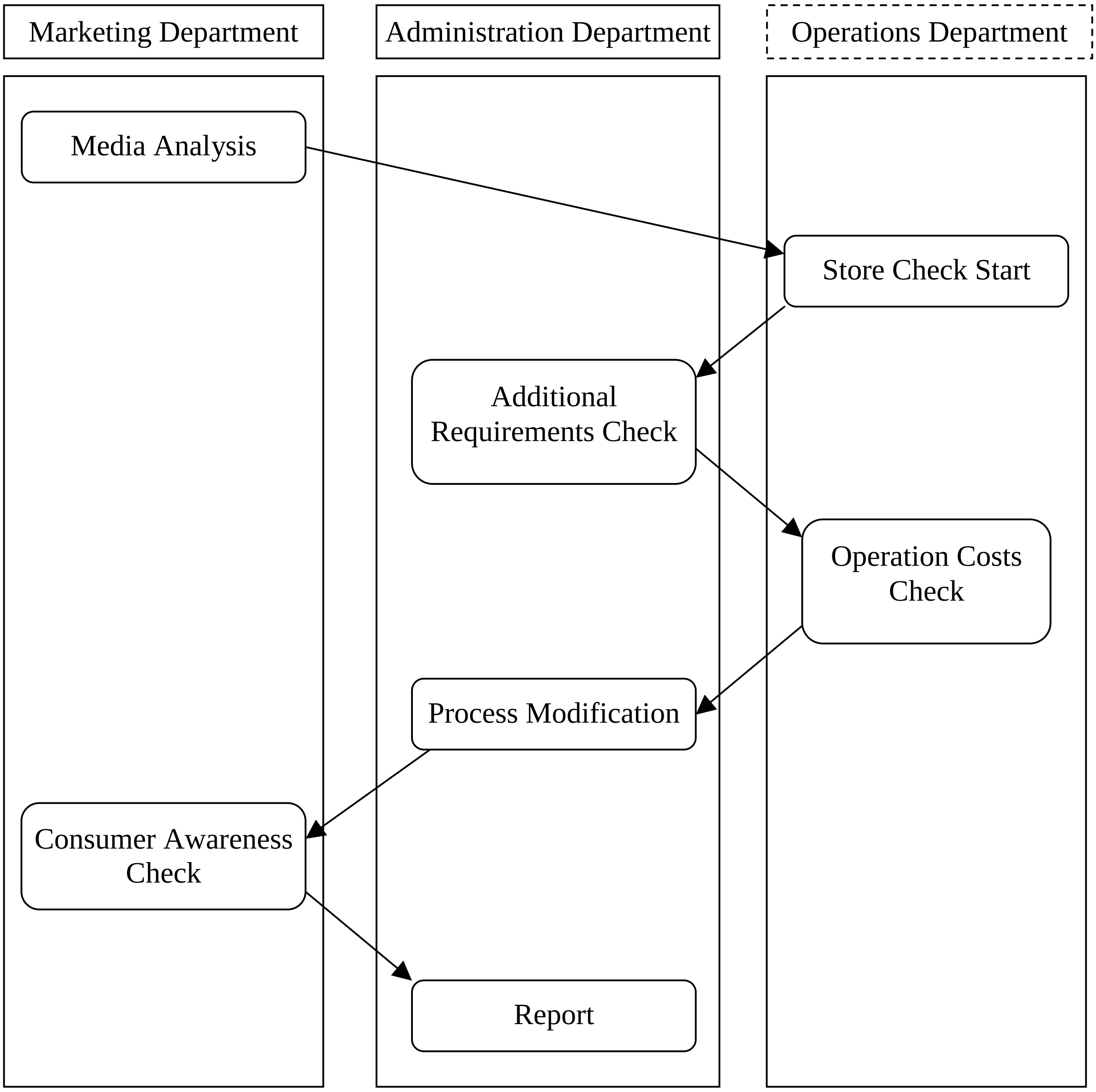

Figure 4.11: The Market Research Process II

The Store Check Process

For the execution of the store check process, an employee has to visit and investigate some stores. Thus, the store check process starts with the preparation of a travel application by an employee of the operations department. Afterwards, the travel application is sent to the management department, where a manager will approve or refuse it. In case the manager refuses the application, it will be sent back to the employee and he has to modify it. These steps will be repeated until the application is confirmed. After the manager confirms the travel application, the employee will travel and prepare a travel report at the end.

If the travel is done as planned within the travel application, the travel report will be sent directly to the personnel department that is responsible for the refund of the costs.

In case some deviations occurred, the travel report will be sent again to the management department for confirmation. If a manager refuses the travel report, the employee has to modify it, until it is confirmed. After the confirmation, the travel report will be sent to the personnel department.

The personnel department finally pays the travel costs and prepares a final report that indicates all travel related information.

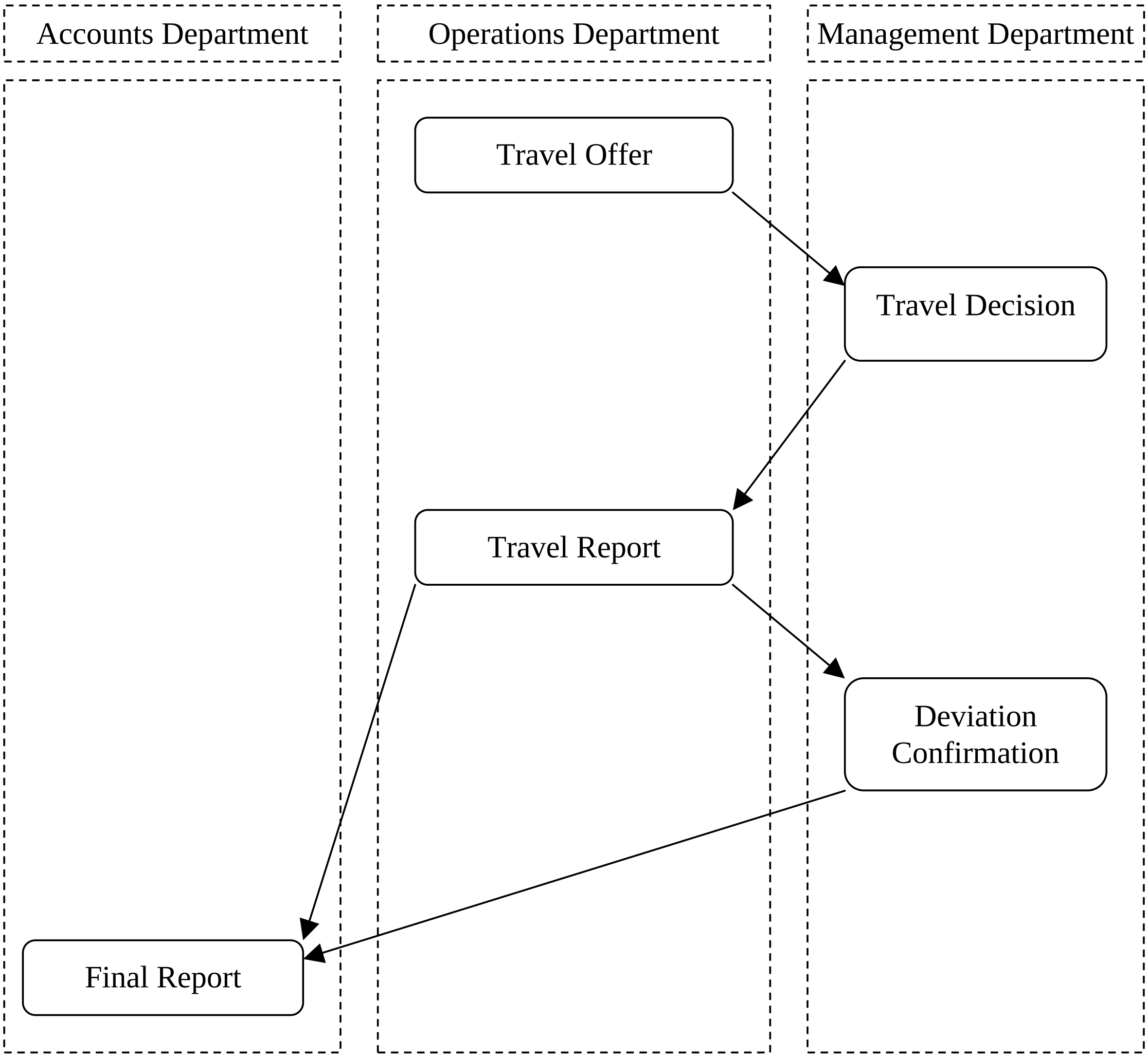

Figure 4.12: The Store Check Process

4.5.2 Interoperability Scenarios

In this subsection two interoperability scenarios will be defined and illustrated. These interoperability scenarios will later be used to evaluate the interoperability interface.

Interoperability Scenario I

The interoperability scenario in Figure 4.13 illustrates how the market research process I and the store check process can interoperate.

The market research process starts with a media analysis. The interoperability starts during the execution of the *StoreCheck* operation. In particular, for the execution of the store check operation the market research process assigns the execution of a store check process by another company.

After the start of the store check process, the market research process waits for the completion of it. When the store check process is completed, the market research process is notified via an information message indicating the results of the store check process.

At this point, the interoperation of the two companies is finished. Afterwards, the market research process handles these data and continues with the execution of itself.

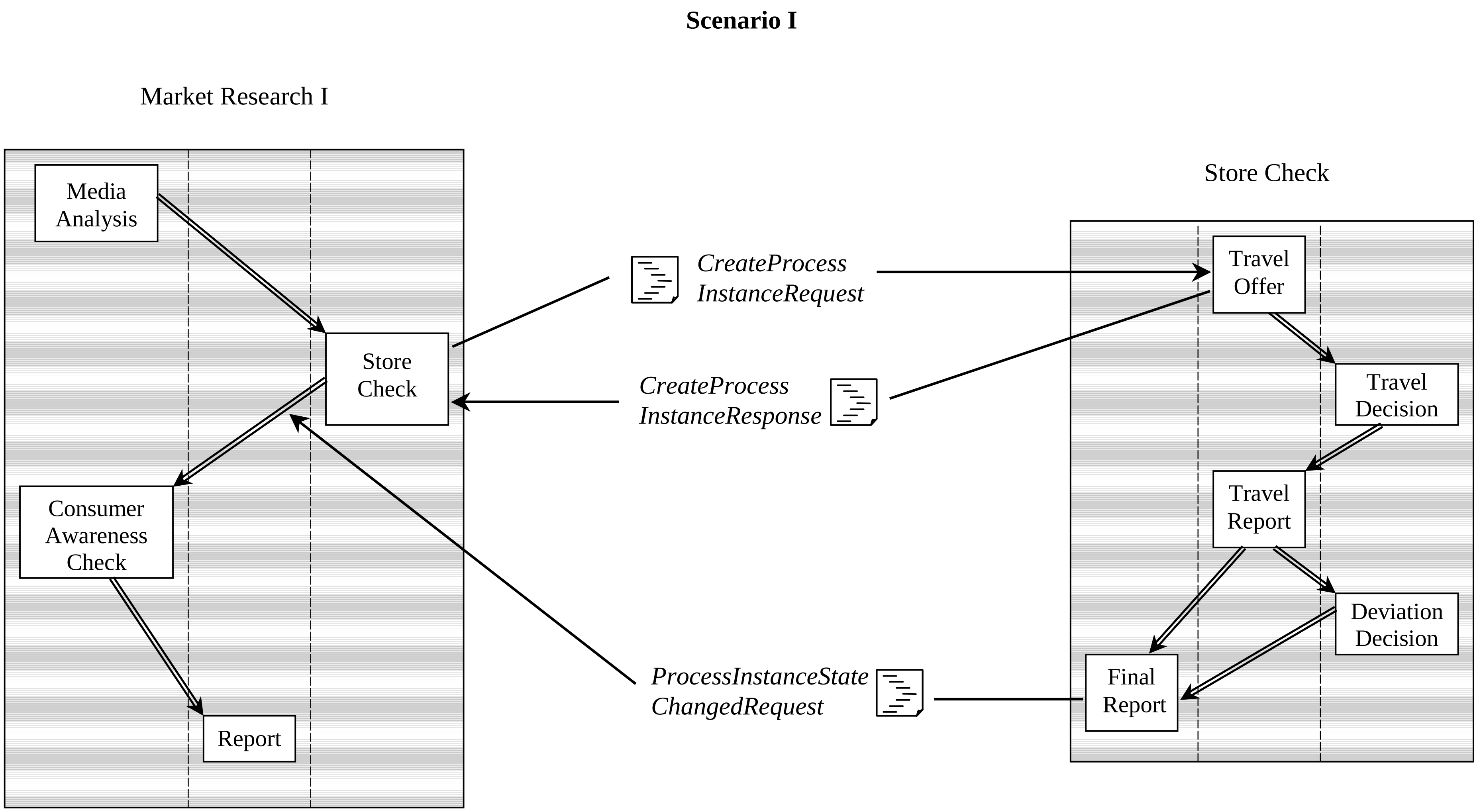

Figure 4.13: Interoperability Scenario I

Interoperability Scenario II

The interoperability scenario between the market research process II and the store check process is shown in Figure 4.14.

When the execution of the market research II process reaches the work item *StoreCheckStart*, it delegates another company for the execution of a store check process, and continues with the execution of the *AdditionalRequirementsCheck* work item.

After processing the additional requirements, the market research process II suspends the store check process.

In Figure 4.14 the store check process is at the *TravelReport* work item, at the time the suspend request arrives. However, it could have been at a different stage of the execution.

Next, the market research process requests the other company for information about the store check process. Handling of these data, which may be incomplete and inconsistent, is the issue of the market research process.

Afterwards, the market research process starts the execution of the *ProcessModification* work item. Depending on the results of the process modification, the market research process requests the other company to wake up the suspended store check process and waits for the completion of it.

Finally, the market research process receives a notification indicating the completion of the store check process and the resulting data. At this point the interoperability scenario is finished. The market research process handles these data and continues with the execution of itself.

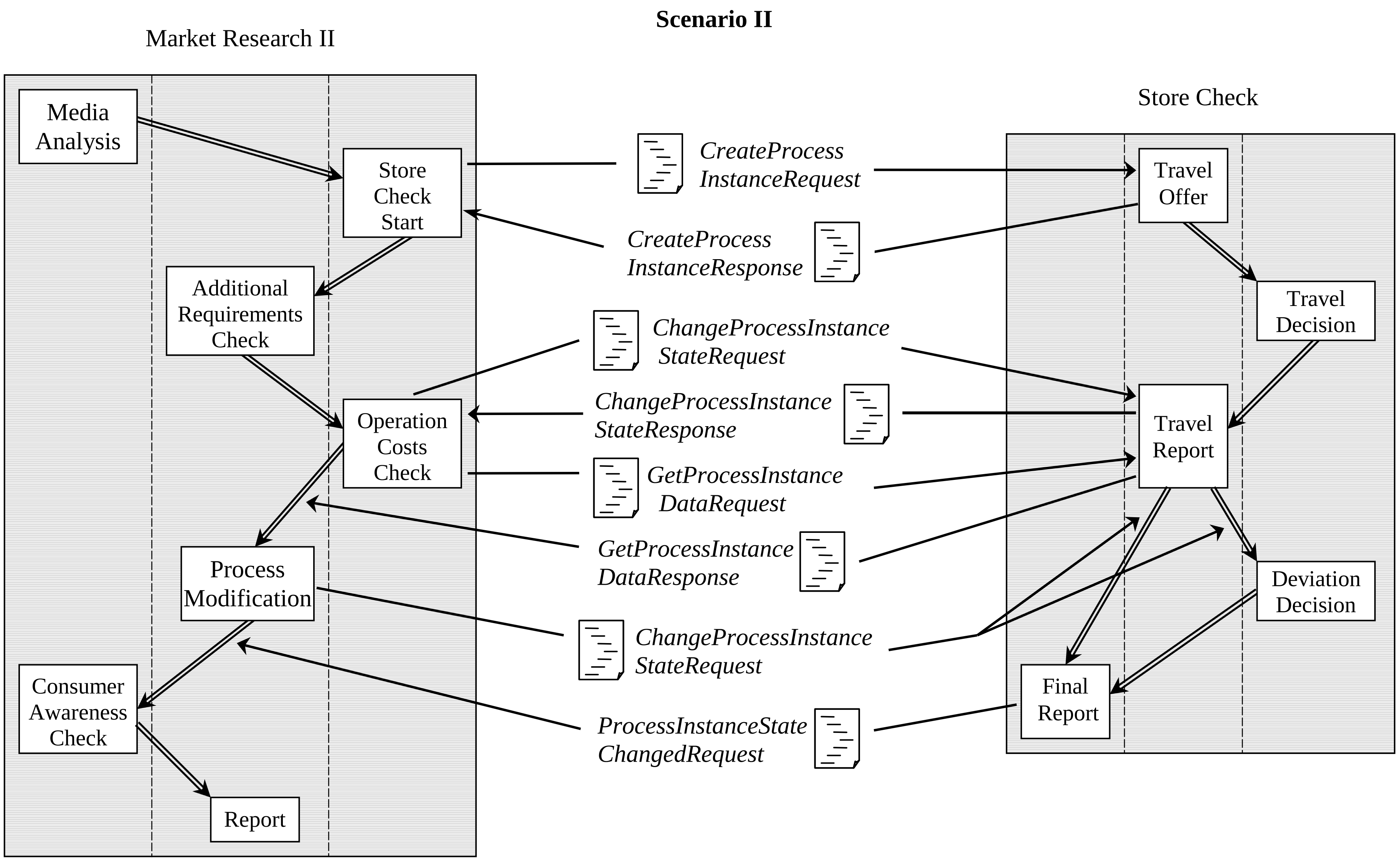
Scenario II
Market Research II
Store Check
Media Analysis
Store Check Start
Additional Requirements Check
Operation Costs Check
Process Modification
Consumer Awareness Check
Report
Travel Offer
Travel Decision
Travel Report
Deviation Decision
Final Report
CreateProcess InstanceRequest
CreateProcess InstanceResponse
ChangeProcessInstance StateRequest
ChangeProcessInstance StateResponse
GetProcessInstance DataRequest
GetProcessInstance DataResponse
ChangeProcessInstance StateRequest
ProcessInstanceState ChangedRequest

Figure 4.14: Interoperability Scenario I

4.6 Future of Workflow Interoperability

The workflow interoperability currently standardized by the Workflow Management Coalition allows only the implementation of simple interactions. In order to support more complex interactions, additional operations are planned for the future. These operations will be described briefly in Subsection 4.6.1.
The standardized workflow interoperability is currently limited to chained process and sub-process models. For several cases, they are sufficient practically, however there are also situations where the parallel-synchronized model is required. Due to the absence of support for the parallel-synchronized model, it is not possible to implement more sophisticated interoperability scenarios in which two process instances on different workflow engines can be executed simultaneously and can exchange data at certain points of the execution. This will be the next area of development of workflow interoperability.
Besides the Workflow Management Coalition's work there are also other significant efforts made for the interoperability of workflow management systems. One is the approach of Wil van der Aalst, who is proposing a concept for the interoperability of workflow management systems based on Petri nets. The pros and contras of van der Aalst's approach will be briefly summarized in Subsection 4.6.2.

4.6.1 Further Efforts of the Workflow Management Coalition

The Workflow Management Coalition has reserved additional operations for future use. The following table illustrates these operations [WfMC00]:

	General Operations	Process Definition	Process Instance	Observer
WfQueryInterface	**X**			
ListInstances		**X**		
SetData			**X**	
Subscribe			**X**	
Unsubscribe			**X**	
GetHistory			**X**	
Notify				**X**

Figure 4.15: Additional Interoperability Operations planned by WfMC

General Operations

WfQueryInterface

The *WfQueryInterface* operation will be used to query an implementation for various generic capabilities. In particular, it could be used to determine capabilities of an implementation such as support for various security requirements, conformance to the Wf-XML interoperability specification or XML processing features [WfMC00].

Process Definition Operations

ListInstances

The *ListInstances* operation will be used to retrieve a list of instances of the given process definition. Each instance in the returned list will be identified with its key, name and priority [WfMC00].

Process Instance Operations

SetData

This *SetData* operation will be used to set values of any number of properties of the given process instance resource. The settable properties of a resource will be given as the operation parameters, dependent on the interface in which it is invoked. At least one parameter must be provided, in order for the operation to have any effect, but all parameters will be optional. Current values of all the properties of the resource will be returned [WfMC00].

Subscribe

The *Subscribe* operation will be used to register a resource with another resource as a party, which is interested in status changes and events that occur. If this particular resource does not support other observers, an exception message will be returned to the caller [WfMC00].

Unsubscribe

The *Unsubscribe* operation will be used to remove a resource from the list of registered observers of a resource. The calling resource will no longer receive event notification after executing this operation [WfMC00].

GetHistory

The *GetHistory* operation will be used to retrieve a list of events that have occurred on this resource. If the service implementing this resource has not kept a transaction log, there may not be any history available. However, if there is, it will be returned by this method [WfMC00].

Observer Operations

Notify

The *Notify* operation will be used to send an event notification to a registered observer resource of a process instance [WfMC00].

4.6.2 A Petri Net-based Approach on Workflow Interoperability

Technologies such as Electronic Data Interchange, the Internet, and the World Wide Web (WWW) enable multiple organizations to participate in shared business processes. The rise of electronic commerce, virtual organizations and extended enterprises highlights the fact that more and more business processes are crossing organizational boundaries [KaWh96]. Thus, modern workflow management systems should be able to deal with workflow processes distributed over a number of organizations.

Considering interorganizational workflows, if each business partner has a private workflow process, which is connected to the workflow processes of some of the other partners, the workflow processes are termed as loosely coupled workflow processes. [Aal98]

Because processes are a dominant factor in workflow management, it is important to use an established framework for modelling and analyzing of workflow processes [HaLa91, Kou95, Law97]. For this purpose, Wil van der Aalst is proposing a framework based on Petri nets. Petri net is a well-founded process modelling technique invented by Carl Adam Petri [Pet62, Pet97]. Since then Petri nets have been used to model and analyze all kinds of processes with applications ranging from protocols, hardware and embedded systems to flexible manufacturing systems, user interaction, and business processes. There are several advantages of using Petri nets for workflow modelling: their formal semantics, graphical nature, expressiveness, analysis techniques and tools provide a framework for modelling and analyzing workflow processes [Aal98a, Aal98b, EKR95, EeNu93, MEM94, WoRe96].

Moreover, the correctness of interorganizational workflows can be verified using standard Petri net techniques. For example to prevent synchronization problems during workflow interoperability, workflow processes, which are modelled with Petri nets, can be checked for deadlocks or livelocks.

Message Sequence Charts (MSC) can be used to specify the communication between loosely coupled workflow processes. Message Sequence Charts are a widespread graphical language for the visualization of communications between systems/processes. In practice, there are numerous situations, where the organizations participating in a shared workflow process specify the coordination structure explicitly [Aal99].

According to Wil van der Aalst, workflow processes, which are modelled with a framework based on Petri nets, are examples of appropriate loosely coupled workflows. Furthermore, if message sequence charts are used to specify the communication structure of such workflows, using a technique defined by Wil van der Aalst, they can be checked for 1-consistency with the message sequence chart. However, the notion of 1-consistency described in [Aal99] is restricted to the situation where only one message sequence chart exists. Future work will aim on extending some of the results for the situation with multiple message sequence charts (e.g. n-consistency) [Aal99].

Moreover, van der Aalst approach is based on the assumption that all workflow products implement or will implement Petri nets for workflow modelling. Considering the fact that the workflow products have different origins and are highly specified, it is improbable that all products may implement Petri nets in the future. Therefore, the implementation of this approach is limited to the workflow products that use Petri nets for workflow modelling.

Furthermore, in many cases the coordination structure and the interaction between the business partners of an interoperability scenario are not specified explicitly [MLML96]. Consequently, message sequence charts cannot be implemented as proposed.

Chapter 5

Implementation of the WfMC Interoperability Interface

Beside the information given about workflow management systems and workflow interoperability so far, this work has also an implementation part, in which the interoperability interface as standardized by the Workflow Management Coalition will be implemented and used for two sample applications. For the implementation of the interoperability interface, an interoperability framework will be defined. Using a framework for implementation brings several advantages: reusability of the software, low development and maintenance costs, better software quality, etc [Pre97, ShGa96].

Thus, the rationale of this chapter is to represent different phases of the interoperability framework implementation and the evaluation of interoperability.

The first section comprises the analysis phase, where the application scenarios and the test and evaluation environment are outlined.

The second section will deal with the design of the interoperability framework. In particular, these issues include the design of the necessary classes and interfaces, design patterns and the generation of the complete XML DTD.

The third section will give information about the implementation of the interface and limitations and problems that were encountered during the implementation.

Afterwards, the environments used to test the interoperability framework will be described.

The last section of this chapter will focus on the evaluation of the results of different interoperability approaches. Here, beside the interoperability approach of the Workflow Management Coalition a different approach implemented in the MARIFlow project will be evaluated.

In the first subsection the interoperability approach used in this project will be evaluated with regard to the results noticed during the testing.

Subsequently, the MARIFlow project, which implements a different interoperability approach to enable interoperability for the maritime industry, will be described briefly. After the presentation of the MARIFlow project, critical aspects of the implementation and issues planned for the future will be depicted.

In the last subsection problems that are common for both interoperability approaches will be specified.

5.1 Analysis

The analysis phase of the interoperability framework conception and implementation consists of the inception and elaboration phases. In the inception phase an initial analysis is made to define the scope of the project and to plan the implementation.

In the elaboration phase the requirements from the interoperability interface are analyzed with the help of UML use cases. Moreover, further constraints that will effect the implementation are determined.

5.1.1 Inception

The interoperability framework will be programmed in the object-oriented programming language Java. Here, the framework must support the exchange of application-specific data. The workflow management systems will interact by exchanging Wf-XML messages, which will be parsed for well-formedness and validity using an XML Parser. The exchange of the Wf-XML messages will be realized as HTTP requests / responses.

The workflow management software EasyFlow of the company T-Nova is chosen as the workflow management system for which the framework is built in this work. The framework will be modularized so that only a workflow management system-specific part must be changed to connect to other workflow management systems. The three sample processes described in Subsection 4.5.1 will be modelled and realized using EasyFlow. The interoperability framework will be tested and evaluated using the two sample interoperability scenarios among these three sample processes as described in Subsection 4.5.2.

In order to test and evaluate the interoperability scenarios two separate EasyFlow workflow management systems and the interoperability interface are used.

5.1.2 Elaboration

In the elaboration phase the requirements from the interoperability interface are defined with the help of use cases. The use cases defined the user’s requirements for design and implementation.

As a framework is to be realized, the work’s focus is not on the details of the sample processes’s execution but on the interoperability modalities of the workflow engines in general. Consequently, the interoperability operations described in Section 4.4 define the use cases (see Figure 5.1).

For the use cases Create Process Instance, Get Process Instance Data and Change Process Instance State, workflow management system A is the initiating actor, whereas for the use case Process Instance State Changed, workflow management system B is the initiating actor.

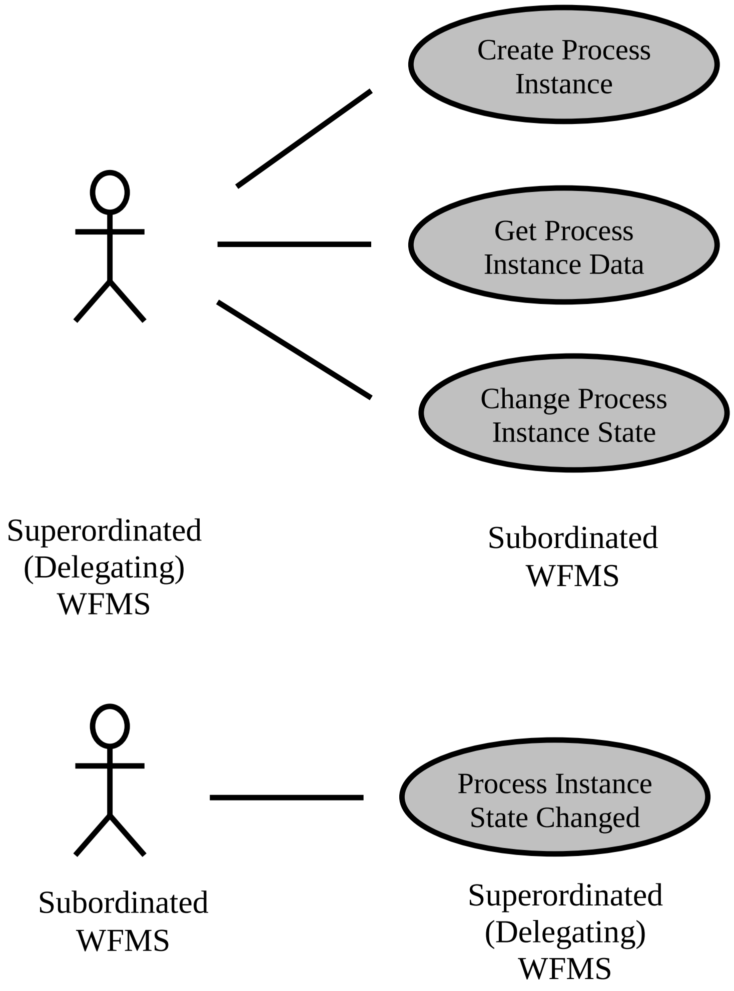
Create Process Instance
Get Process Instance Data
Change Process Instance State
Superordinated (Delegating) WFMS
Subordinated WFMS
Process Instance State Changed
Subordinated WFMS
Superordinated (Delegating) WFMS

Figure 5.1: Use Cases obtained from the WfMC's Interoperability Operations

EasyFlow Constraints

Some workflow management system-specific (EasyFlow-specific) constraints must be considered: In EasyFlow there are two major types of work items that are used for process modelling: A *manual work item* is a work-item that will be executed by a human. Contrary to this, an *automatic work item* is executed by the workflow enactment service. Consequently, the sample processes must be modeled in EasyFlow considering this workflow management system-specific implementation concept. The processes described in Subsection 4.5.1 therefore must be extended with automatic work items whenever they have to send a Wf-XML message.

Examples: Examples: An automatic work item must be added to the Market Research Process I illustrated in Figure 4.10 after the manual work item *StoreCheck*.

Similarly, an automatic work item must be added to the Store Check process described in Figure 4.12 after the manual work item *FinalReport*, so that the end of the process can be reported to the delegating workflow management system (see figure 4.13).

The model of Market Research Process II, which is illustrated in Figure 4.11, has to be extended with four additional automatic work items for interoperability operations (see Figure 4.14).

5.2 Design

In order to conceptualize and build the framework decisions considering the functionality have to be made in the design phase. In particular, the required functionality and an adequate partitioning of this functionality are used to realize the moduls and classes that contribute the interoperability framework.

Moreover, to accomplish the requirements for a good design, design patterns are applied [GHJV95].

As mentioned before, exchange of application-specific data must be supported. Technically this results in support for business-specific XML Data Type Definitions. Here an appropriate method that will allow easy implementation and modification for new interoperability scenarios has to be chosen.

5.2.1 The Interoperability Framework

The interoperability framework is divided into layered packages that are defined considering their functionality and degree of application dependency. Figure 5.2 illustrates the layered architecture of the interoperability framework:

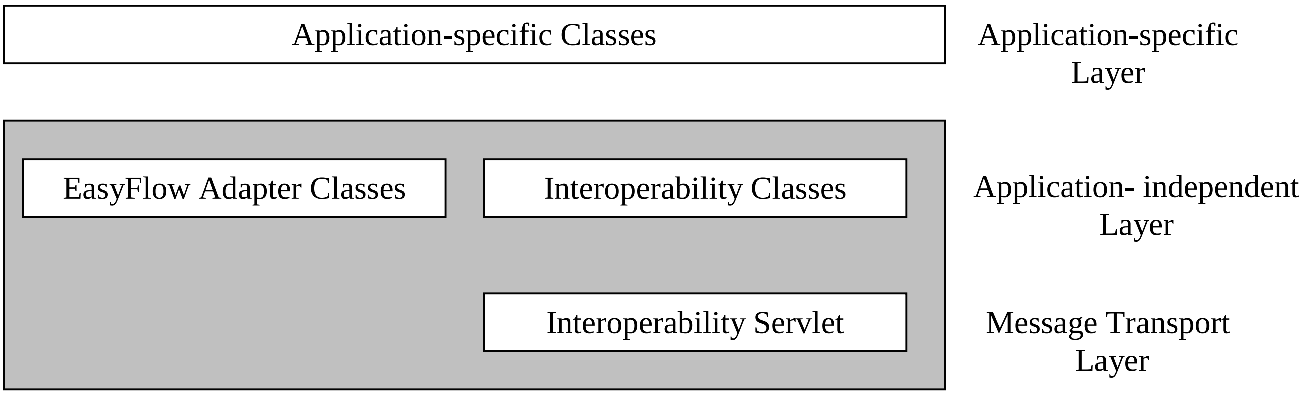

Figure 5.2: The Interoperability Framework

The message transport layer includes the functionality for the exchange of HTTP requests. The EasyFlow adapter classes and the interoperability classes contribute the application-independent layer. The application-specific layer contains the application-specific classes.

Each layer of the framework is realized by one or more package(s) that contain the related class(es). All workflow applications share the generic interoperability framework. However, each workflow application will have its unique application-specific layer.

5.2.1.1 The Application-independent Layer

Interoperability Classes

The interoperability classes (package *de.tnova.ezn.wapi.interoperability)* are illustrated in Figure 5.3. These classes will be briefly described:

InterOpException

Any exception that occurs during the execution of a Wf-XML operation would have to be returned to the caller. Various types of exceptions can be anticipated, including temporary and fatal error types. Thus, an *Exception* element is defined in the specification to carry this information [WfMC00]. Consequently, for the exceptions that may occur during the interoperability operations, a class named *InterOpException* is designed to encapsulate the required exception information for the *Exception* element.

Dispatcher

The *Dispatcher* class first creates a complete DTD by merging the standard Wf-XML DTD Version 1.0 with the application-specific DTD. Next, it parses incoming XML-requests for validity and well-formedness using the complete DTD. Then, it parses the contents of the header and body parts of the request to store the necessary information. Finally, it initiates an adequate sub-class of the *SourceManager* class with the parsing information.

SourceManager

The abstract class *SourceManager* is responsible for the handling of the request and the generation of the response, in case a response is required. However, the *SourceManager* class implements only the general, not application-specific tasks, for example the generation of the response message's transport and header parts, which do not contain any application specific data. The application-specific tasks are defined in the sub-classes, which will be described later in this section, of the class *SourceManager* with regard to the design pattern template method.

ObserverManager

The abstract class *ObserverManager* coordinates the tasks to be executed whenever the workflow management system wants to start a Wf-XML operation (see Figure 5.1). In particular, it generates an operation-specific request, builds up an URL connection, sends the request to the other workflow enactment service and evaluates the returned response.

Similar to the *SourceManager*, the *ObserverManager* deals only with the application independent tasks. Sub-classes of the ObserverManager that handle the process-specific tasks will be described later.

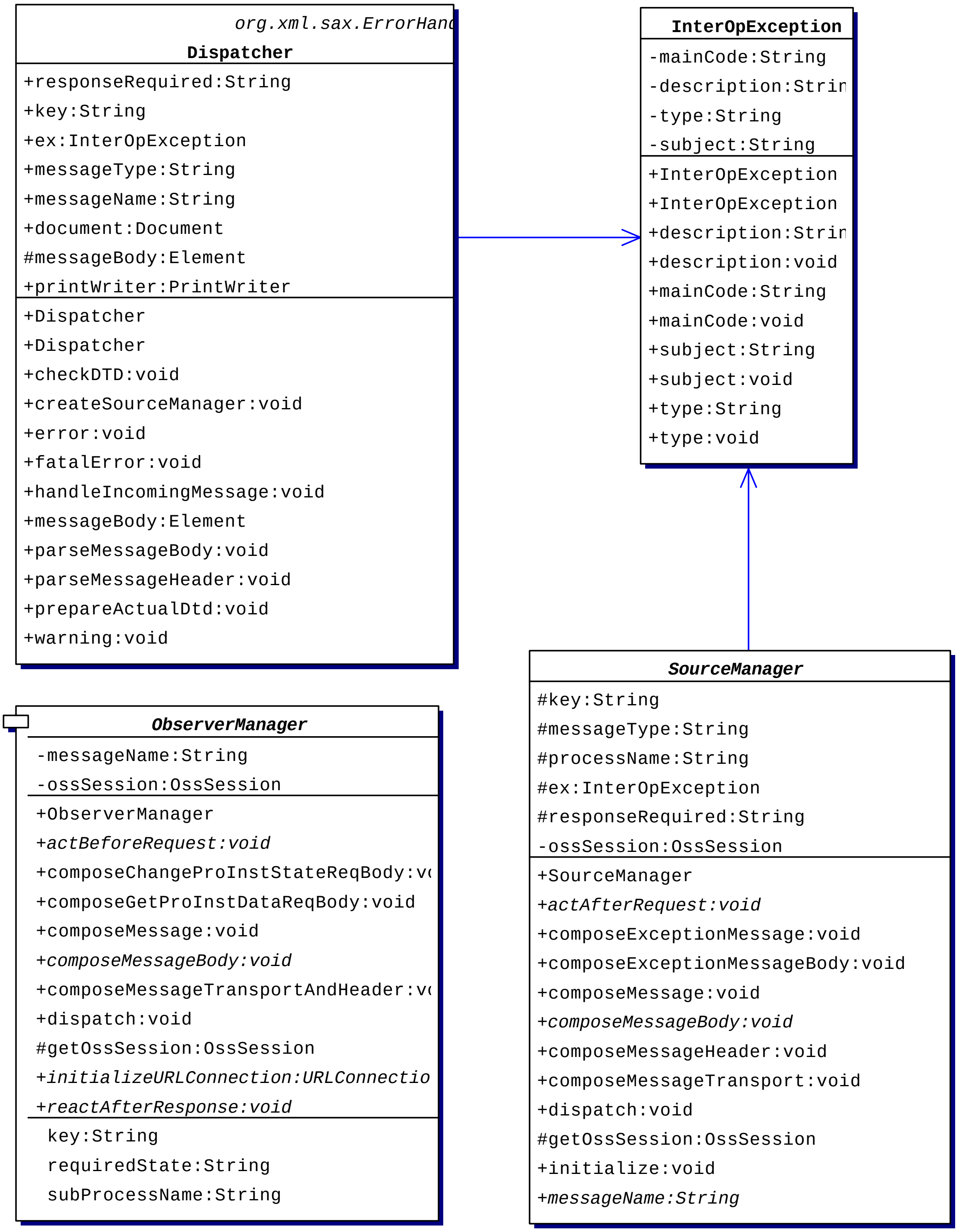

Figure 5.3: Interoperability Classes (package: *de.tnova.ezn.wapi.interoperability*)

EasyFlow Adapter Classes

The application-independent classes that are essential to implement any interoperability scenario in EasyFlow are named EasyFlow Adapter Classes.

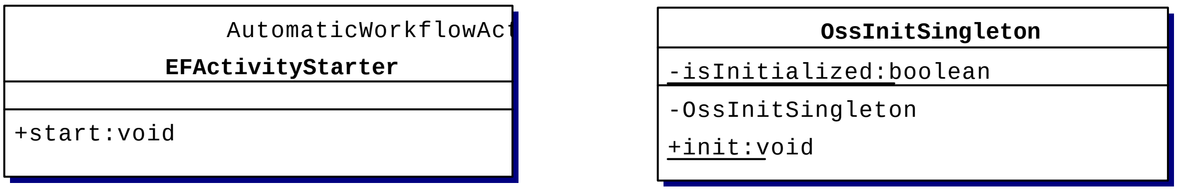

Figure 5.4: The EasyFlow Adapter Classes (package: *de.tnova.ezn.easyflow.wapi*)

EFActivityStarter

The abstract class *AutomaticWorkflowActivity* is implemented in EasyFlow WAPI to allow an activity to create a Java object and to method invocation on it. The class *EFActivityStarter* is defined as a sub-class of the abstract class *AutomaticWorkflowActivity*, so that it can initiate any adequate sub-class of the abstract class *ObserverManager* to start a Wf-XML operation.

OssInitSingleton

EasyFlow is a database-oriented workflow management system. Thus, anytime the interoperability interface has to execute some actions on a process definition or a process instance an *OssSession* (Object Storage System) is required. An *OssSession* object is also part of the EasyFlow WAPI and can be used several times after it has been initialized. The task of the class *OssInitSingleton* is to ensure that the *OssSession* is initialized only once.

5.2.1.2 The Application-specific Layer

Figure 5.4 illustrates the classes, which are responsible for the application-specific tasks of the of the Wf-XML operations. These classes are sub-classes of the abstract *SourceManager* and *ObserverManager* classes and take over the application-specific actions. Contrary to the packages of the application-independent layer, this package must not be implemented completely by both workflow management systems. For example, the workflow management system responsible for the execution of the *StoreCheck* process doesn't have to implement the whole package but only the classes *StoreCheckObserverManager* and *StoreCheckSourceManager*.

With the implementation of the application-specific sub-classes, the generic interoperability framework, which is described in Subsection 5.2.1, will be adapted to particular applications and business processes. Here the design pattern *Template Method* and the concept of a *Framework* are implemented [Pre97].

StoreCheckObserverManager

This class defines the StoreCheck application-specific methods that enable the execution of the automatic work item, which is responsible for reporting the end of the StoreCheck process.

MarketResearch1ObserverManager

The class *MarketResearch1ObserverManager* defines the Market Research Process I specific methods that enable the execution of the automatic work items of this process.

MarketResearch2ObserverManager

Similarly, the *MarketResearch2ObserverManager* class defines methods that enable the execution of the automatic work items of the Market Research Process II.

The StoreCheck-, MarketResearch1- and MarketResearch2ObserverManager classes encapsulate methods for sending request messages to start Wf-XML operations. These operations are defined as activities of automatic work items in the related process models.

StoreCheckSourceManager

This class defines the StoreCheck process specific methods that are necessary to execute the related Wf-XML operation requests.

MarketResearch1SourceManager

The class *MarketResearch1SourceManager* defines the Market Research Process I specific methods that enable the evaluation of the *ProcessInstanceStateChanged* operation request.

MarketResearch2SourceManager

Similarly the *MarketResearch2SourceManager* class contains methods that enable the evaluation of the *ProcessInstanceStateChanged* operation request.

The StoreCheck-, MarketResearch1- and MarketResearch2SourceManager classes encapsulate necessary methods to evaluate the incoming Wf-XML operation requests. The operations they execute are not modelled in the process definitions. They are invoked for each request message.

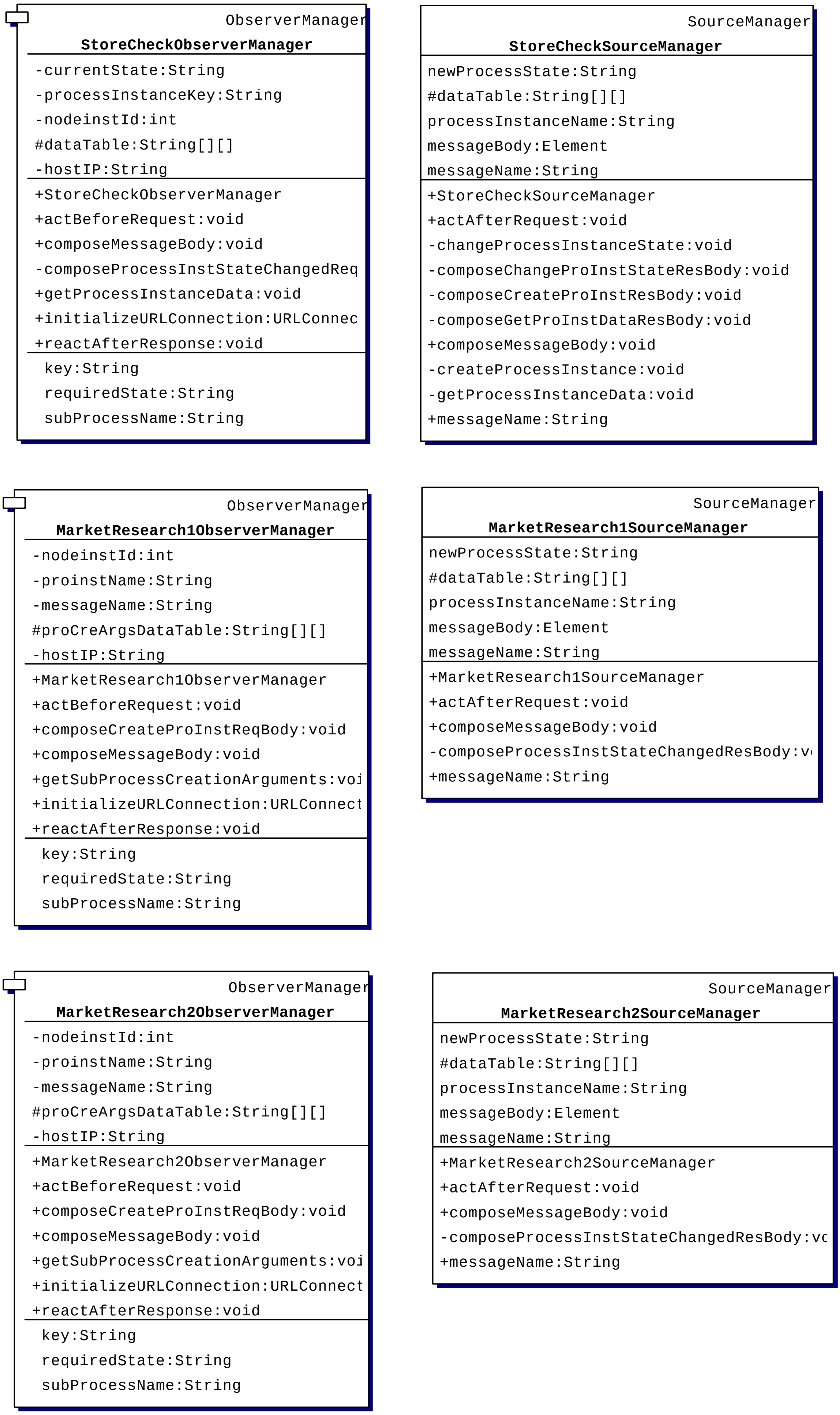

Figure 5.5: The Application-specific Classes (package: *de.tnova.ezn.manager*)

5.2.2 Design Patterns

Beside the functionality there are also further requirements for a good design: The application-specific parts and EasyFlow-specific parts must be separated from the generic functionality, so that the interoperability framework can be easily adapted to other interoperability scenarios and workflow management systems. Thus, *design patterns* are followed when designing the framework [GHJV95].

The *ProcessInstanceStateChanged* operation enables a process instance to inform its parent process about state changes (see Section 4.4). Thus, the *Observer* design pattern is applied here. It should be noted that another operation named *Notify* is reserved by the Workflow Management Coalition for future use (see Subsection 4.6.1).

For the design of the interoperability framework the behavioral design pattern *TemplateMethod* is used. The application-independent parts of operations are collected in *ObserverManager* and *SourceManager* classes, whereas the application-specific actions are implemented in sub-classes of these classes. Consequently, for an interoperability scenario sub-classes of *ObserverManager* and *SourceManager* implementing only the application-specific functionality must be defined. The resulting reusability of the code is one of the major advantages of object-oriented programming.

As explained in Subsection 5.2.1.1, each time an operation related to a process definition or a process instance has to be executed, an *OssSession* object is required by EasyFlow. The object instance can be used for several operations, however it must be ensured that it is initialized only once. Thus, with regard to the creational design pattern *Singleton* the class *OssInitSingleton* is defined as a Singleton (see Subsection 5.2.1.1). The class *OssInitSingleton* ensures that there is only one instance of this class, which initializes any *OssSession* object.

5.2.3 Generation of the complete DTD

To enable workflow interoperability, not only the functionality must be provided, but also exchange of business-specific data must be supported. Therefore, an appropriate method to support the application-specific data is obligatory. The Workflow Management Coalition proposed some procedures, which were described in Section 4.4 in detail.

In this work the procedure to extend the standard Wf-XML DTD Version 1.0 to include the application-specific data is chosen, because it can be easily implemented and allows for flexible support of any interoperability scenario. In order to generate the complete DTD, the standard Wf-XML DTD Version 1.0 is merged with the application-specific DTD.

Here, the *ContextData* and *ResultData* elements contribute the obligatory part of the application-specific data. Therefore the *ContextData* and *ResultData* elements, which are defined as placeholders with the attribute ANY in the standard WF-XML DTD Version 1.0, are replaced with the application-specific *ContextData* and *ResultData* elements.

The generated complete DTD is used to parse the messages for validity and well-formedness. The main advantage of this method is its reusability and the ease of maintenance: For a new interoperability scenario the definition of the new application-specific data is sufficient, because the framework is able to generate the new complete DTD without any modifications.

5.3 Implementation

In order to implement the interoperability interface, some development tools are made use of. Not the superiority of these tools rather than the functionality and the availability of them played the major role for their choice.

This section will not only name the tools used in the development but also describe limitations and problems encountered during the implementation.

5.3.1 Development Tools

For the implementation of the interoperability framework in Java, the Integrated Development Environment *Visual Age® for Java™, Professional Edition for Windows®, Version 3.0* of the *IBM Corporation* is used.

Furthermore, in the design and implementation phases the tool *Together/J Version 4.2* of the company *TogetherSoft* was used.

For the coordination of the communication between the workflow management systems, the Servlet engine *Tomcat Version 3.1.1* is used. Tomcat is the reference implementation for the complementary Java Servlet 2.2 and JavaServer Pages 1.1 technologies. Beside the framework classes presented in Subsection 5.2.1, a HTTP servlet named *InteroperabilityServlet* was implemented in Java. This class coordinates the exchange of messages via HTTP protocol and completes the interoperability interface.

The Wf-XML messages are parsed with the *Xerces (Version 1.2.0)* Parser for well-formedness and validity. The *Xerces* parser is furthermore used to build the complete DTD, which is generated through the merging of the standard WF-XML DTD Version 1.0 and the application-specific DTD.

5.3.2 Limitations of Workflow Interoperation

Finding appropriate processes and defining interoperability scenarios between them has been the major difficulty during the implementation. The reason for this is that companies are considering processes and process-related information as know-how and are not ready and willing to publish this information. Moreover, the specification is supporting simple interoperability scenarios in its current state. Therefore, the processes and interoperability scenarios to be chosen are limited to low complexity.

Due to these limitations the business processes and interoperability scenarios used to test and evaluate the implementation are chosen from the marketing area. However, the application area of the examples does not play a considerable role when testing the functionality of the framework.

There are further constraints related to the workflow management product used for the modelling of the processes. Considering a process modeled in EasyFlow and its work items, only a work item of the type automatic can contain an activity that can initiate a class in Java and cause a Wf-XML message to be sent. Thus, process models must be modified to include automatic work items at steps where interoperability operations should be used. Moreover, for each automatic work item an activity including the Java class to initiate and the necessary parameters must be defined. In particular, each activity includes the Java class named *EFActivityStarter*, the name of the interoperability operation and the parameters of the operation, if any. The class *EFActivityStarter* initiates any adequate sub-class of the abstract class *ObserverManager* to start an interoperability operation.

5.4 The Test Environment

In order to test and evaluate the interoperability framework, the sample interoperability scenarios illustrated in Figures 4.13 and 4.14 are used.

For testing two separate EasyFlow workflow environments are used. Both environments include:

- Tomcat (Version 3.1.1)
- Xerces Parser (Version 1.2.0)

- The standard WF-XML DTD Version 1.0 and the application-specific DTD
- A servlet named *InteroperabilityServlet*
- The packages *de.tnova.ezn.wapi.interoperability* and *de.tnova.ezn.easyflow.wapi*

In addition, one of the EasyFlow workflow environments contains the Market Research Project I, the Market Research Project II and all process-related data, such as users, activities etc… Furthermore, this workflow environment includes the following classes in the package *de.tnova.ezn.manager*:

- MarketResearch1ObserverManager
- MarketResearch2ObserverManager
- MarketResearch1SourceManager
- MarketResearch2SourceManager

The second EasyFlow workflow environment includes the Store Check process and its process-related data. Moreover, it contains the following classes in the package *de.tnova.ezn.manager*:

- StoreCheckObserverManager
- StoreCheckSourceManager

The interoperability scenarios illustrated in Figure 4.13 and 4.14 have been tested successfully. Consequently, the interoperability approach implemented in this project will be evaluated in the next section.

5.5 Evaluation

The main goal of this section is to evaluate the results of different interoperability approaches implemented in two projects.
First, the interoperability approach followed in this project will be evaluated with regard to the results noticed during the testing.
Afterwards, the MARIFlow project, which implements a different interoperability approach to enable interoperability for the maritime industry, will be described briefly. After the presentation of the MARIFlow project, critical aspects of the implementation and issues planned for the future will be depicted.
In the last subsection problems that are common for both interoperability approaches will be specified.

5.5.1 Evaluation of Interoperability in this Project

The implementation of the interoperability interface, as defined by the Workflow Management Coalition, has been tested using the two interoperability scenarios described in Subsection 4.5.2 of this work. Both scenarios have been enacted successfully using the interoperability framework.
The processes that implement the interoperability scenarios are defined with the lowest detail necessary to test the functionality of the interface. However, they gave important clues about the most critical decisions to be made before implementing an interoperability scenario:

- Conformance
- An interoperability contract
- Definition of an interoperability scenario

The first subsection will describe how conformance to workflow interoperability standard should be expressed by the vendors of workflow management systems.

The next subsection will present the requirements for an interoperability contract that has to be established between the parties participating in an interoperability scenario.

The significance of a well-defined interoperability scenario will be the subject of the third subsection.

5.5.1.1 Conformance

When companies are planning to implement an interoperability scenario using the interoperability interface as defined in the Workflow Standard – Interoperability Wf-XML Binding, it is a prerequisite that their workflow management systems conform to this specification [WfMC00]. Thus, a method for ascertaining a system's conformance to the specification is essential.

The Interoperability Abstract Specification describes an approach for determining the ability of a workflow management system to implement the functionality described in this specification [WfMC96]. This approach is declared to be equally applicable to the Workflow Standard – Interoperability Wf-XML Binding.

In particular, two important evaluation criteria are determined in [WfMC96]: conformance statements and the capacity matrix.

Conformance Statements

To enable a purchaser to match compatible workflow products from different vendors, each vendor should publish the interoperability capabilities of their product giving clear indication of [WfMC96]:

- The transport mechanism(s) it uses to effect interoperability with other workflow engines
- The style(s) of interoperability dialogue it can support (*atomic, batched* or both):

  In *atomic* transmission a dialogue between two workflow engines is achieved by exchanging messages one by one. An alternative approach that may be employed for conducting dialogues between workflow engines is named *batched* transmission. Workflow engines that communicate asynchronously may batch up groups of request messages and send them as a session to the target workflow engine for processing [WfMC96].
- The mode(s) of interoperability dialogue it employs (*half-duplex* or *full duplex*)

Capabilities

The vendor of each product should produce a capability matrix that shows whether their workflow engine can initiate the message associated with that operation and whether it can respond to it [WfMC96].

Consequently, after the implementation of the interoperability interface, the capability matrix of EasyFlow is illustrated in the next figure:

Operation	**Initiate**	**Respond**
CreateProcessInstance	Yes	Yes
GetProcessInstanceData	Yes	Yes
ChangeProcessInstanceState	Yes	Yes
*ProcessInstanceStateChan*ged	Yes	Yes

Figure 5.6: Capability matrix of EasyFlow

5.5.1.2 Interoperability Contract

It has been realized that the requirements and definitions described in the Workflow Standard – Interoperability Wf-XML Binding are not enough for the implementation of an interoperability scenario. Thus, the Workflow Management Coalition recommends the establishment of an *interoperability contract* between the parties participating in an interoperability scenario [WfMC00].

In practice, the establishment of an interoperability contract is essential for an interoperability scenario. For the implementation of the interoperability scenarios evaluated in this work, the following assumptions are made:

- Both parties agreed on the application-specific data that will appear in the *ContextData* and *ResultData* elements.
- Both parties agreed on merging the standard WF-XML DTD Version 1.0 and the application-specific DTD in order to obtain the complete DTD.
- Both parties also agreed that, Wf-XML messages will be parsed not only for well-formedness but also for validity with regard to the complete DTD. Therefore, Wf-XML messages that are not fully compliant with the complete DTD are considered to be invalid and will not be processed.

Besides these assumptions made for the evaluation, there are further issues that should be addressed in an interoperability contract in many cases. Important issues to be described in the interoperability contract are:

- Security considerations:
  - Encryption of HTTP requets (SSL) / encryption of email (PGP)
  - Non-repudiation through contracts with digital signatures, messages with digital signatures
  - Firewall configuration requirements
- Error handling:

– Definition of the element *Sub-code*: A *Sub-code* is a three digit positive integer. It details the main code, e.g., when the main code represent the exception “Invalid Key”, the sub-code may specify the exception by saying that the format of the key is wrong [WfMC00].

- Definition of the element *Description*: A *Description* element includes detailed information about the error. Thereby it details the element *Subject*.
- Required recovery actions
- Data constraints:
- Allowable characters
- Character set encoding
- Field lengths
- Maximum message size
- Transport protocol limitations [WfMC00]:
- Required header data
- Time-out values
- Buffer size etc.

5.5.1.3 Definition of an Interoperability Scenario

After the interoperability contact is established between the two companies, the next step is the detailed design of an interoperability scenario.

Here, appropriate modelling of both processes is obligatory. In case these processes are already modeled, they have to be modified to enable the execution of the required Wf-XML operations. The necessary modifications may differ depending on the workflow management products used by the companies.

During the execution of a Wf-XML operation exceptions or errors may occur. A suitable approach to define the procedure to handle such situations is necessary. In particular, if an operation could not be processed due to an error, a response message that includes information related to the error is returned. A well-defined interoperability scenario should include a recovery procedure to be followed depending on the exception information. For example, in a process that is modeled with regard to possible exceptions, a person may be informed about the exception via email notification.

Modifications in an interoperability scenario will lead to modifications in the processes and in the process-specific parts of the interface. Consequently, this will require additional efforts and time. Thus, detailed definition of an interoperability scenario in the design phase is of great importance. However, in several cases enterprises prefer to keep their process definitions and information related to the execution of process instances private.

Consequently, the detailed definition of the interoperability scenario is one of the issues that have to be agreed on by the companies before the implementation.

5.5.2 Evaluation of Interoperability in the MARIFlow Project

The MARIFlow Project

In maritime industry, materials used in shipbuilding or repairs need to be certified by a classification society. Here, the classification society checks the material sent from the production plant and issues a certificate if the material fulfills the requirements. Subsequently, the certificate is delivered both to the production plant and the customer. The certificate is checked at every production stage as well as at ship's handover and at each survey during the ship's life cycle.

Consequently, this process requires the flow of documents among several organizations, e.g., classification societies, shipyards, suppliers, legal/insurance/government establishments. The execution is currently based on paper documents, which makes the process slow, expensive, tedious and error-prone. The improvement of the efficiency and quality of this business process is within a worldwide community requires an appropriate coordination structure.

The MARIFlow project aims at providing a prototype of an architecture for automating and monitoring the flow of control and data over the Internet among different organizations participating in the maritime industry. Thus, in the MARIFlow project diverse technological facets such as communication, security, databases, transaction support and agents have to be addressed. In particular, the goal of the project is to develop an adaptable workflow engine through which the activities of the different participants in the maritime industry can be harmonized, combined, and expanded through better tracking of functional dependencies and documents, improved data access and handling, and lower administrative overheads.

In the MARIFlow system, the higher-order process is defined through a graphical user interface, which is then mapped to a textual language called FlowDL. FlowDL allows indicating the source of the documents, their control flow and the activities that make use of these documents. A process definition in FlowDL is executed through co-operating agents, called MARCAs (MARIFlow Co-Operating Agents), that are automatically initialized at each side that enacts the process. MARCAs are responsible for handling the activities at their sites, routing the electronic documents, according to the process description among other MARCAs, keeping track of process information by logging activities, and providing security and authentication of documents during communication.

Further details of the MARIFlow project can be found in [Dog01a, ALSS00, Dog01b, CiDo01].

Consequences

The interoperability approach implemented in the MARIFlow project is based on the routing of electronic documents. Documents are exchanged by storing them in a folder hierarchy at the destination organization. Storing a document in a folder by another organization's system is recognized by a polling process. Subsequently, this process initiates a local process that is responsible for handling of this document.

Thus, contrary to the interoperability approach of the Workflow Management Coalition no interoperability messages are exchanged between the participating organizations. Currently, documents are not packed in XML messages, however XML and PDF formats are planned to be used in the future.

There are some critical aspects of the workflow interoperability based on document passing:

- Uniqueness of file names must be provided. A solution to this problem is to use the process instance name as a prefix for the generation of file names. Thereby, the file names consist of the process instance name and the name of the document. Another solution is to provide a folder for each process instance.
- The maintenance of file names is of major importance. The initial name of a file must be maintained, so that the observer will be able to recognize it after it has been handled by other actor(s).
- Beside the documents being passed, metadata about these documents must be delivered: e.g., sender, date, MIME-type, document-Id, job account number.
- The metadata of a document is sent in a separate document whose name corresponds to the original document.

- Synchronization of processes distributed over different organizations is a complex issue. For a process working on a document, it is difficult to determine if other processes have completed their operation on the document.
- Automatic initiation of processes on a subordinated system is realized usually through the input of documents. Time triggers are used to a lesser extend.
- A mechanism to support garbage collection is also required.
- Organizations participating in the process are protected by firewalls. Thus, to enable the filing of documents into folders that reside on systems protected by firewalls two methods are used: The first method is to pass encryption emails that contain the documents and are unwrapped automatically. The second is communication through a predefined port using HTTP requests.

Moreover following issues are not handled in the current prototype but are planned to be addressed in the future:

- A compensation mechanism that should handle failures in a timely and efficient fashion is essential. In case an exception occurs, the system should be taken to a stable state before the detection of the failure. However, it is not acceptable to roll back all successfully terminated activities in case of a failure. A hierarchical approach to failure handling is required, which allows for partially rolling back to the nearest point in process history tree where it is possible to restart execution.
- For the handling of exceptions a flexible approach is proposed by Hagen and Alonso [HaAl98]. In this approach, the business logic is separated from the exception handling logic, in order to make it easier to keep track of each. An exception handler is a special process that is started when an exception has been signaled. However, the approach does not propose a solution for handling unpredictable events that may occur anytime during execution. All the exception cases and their handlers must be defined at compile time.

5.5.3 Problems identified in both Projects

Some of the problems identified in the interoperability framework project are also recognized in the MARIFlow project:

- Realistic scenarios are very complex and difficult to be defined. Specification of complex inter-company business processes requires experience.
- Enterprises prefer to keep their process definitions and process execution information private. Thus, they are not willing to allow others to monitor the execution of their processes.

Chapter 6

Concluding Remarks

At the final stage, the rationale of this chapter is to present the conclusion of the work. In the first section results of this work will be summarized.

Subsequently, with regard to the evaluation made in chapter five, critics about the workflow interoperability approach of the Workflow Management Coalition will be presented.

In the last section of this chapter identified areas of further research will be described.

6.1 Summary

In this work, concepts and approaches for interoperability of workflow management systems have been presented. Here, the emphasis has been put on the interoperability approach specified by the Workflow Management Coalition.

The interoperability interface as standardized by the Workflow Management Coalition has been implemented as an interoperability framework. The interoperability framework has been tested and evaluated using the two interoperability scenarios described in Subsection 4.5.2 of this work. Both scenarios have been enacted successfully using the interoperability framework.

With regard to the interoperability framework, the concept of workflow interoperability of the Workflow Management Coalition has been proven. However, the implementation of the interoperability scenarios has been realized under some limitations and assumptions (see Subsection 5.3.2).

Furthermore, a different approach on workflow interoperability, which has been implemented in the MARIFlow project, has been briefly presented and evaluated.

Common problems of two different interoperability approaches have been identified and evaluated.

6.2 Critics

As described in chapter five, the implemented interoperability framework has been successfully tested using the two sample interoperability scenarios. However, it has been recognized that the interoperability approach used in this project is yet not mature.

In order to enable many different workflow management products the conformance, the interoperability specification Wf-XML binding is defined by the Workflow Management Coalition very generally [WfMC00]. Thus, for workflow management products, conformance to the workflow interoperability specification is not difficult however this is not enough to interoperate with other products. This fact is recognized and figured also by the Workflow Management Coalition. Consequently, for the recognized critical issues possible solutions are proposed but not standardized:

- The specification in its current state supports chained and nested sub-process models but not the parallel synchronized model of interoperability. Nevertheless, the processes used in practice are usually very complicated and require the implementation of the parallel synchronized interoperability model.
- Essential details of the implementation, e.g. an exception handling mechanism, the procedure to extend of the standard Wf-XML DTD, the definition of an interoperability scenario, and the details of interoperability contracts are not specified yet. The propositions of the WfMC are not detailed enough.

6.3 Extensions

Therefore possible areas of research for the future to improve workflow interoperability can be identified:

The existing specification is rather in an early stage. It needs to be refined with further interoperability operations in order to allow implementation of more complex scenarios. Thus the realization of the Wf-XML operations planned for the future (see Subsection 4.6.1) is of great importance.

An appropriate procedure should be standardized to extend the standard DTD.

Moreover the definition of interoperability scenarios and the subjects, which must be declared in an interoperability contract, need to be clearly defined. In this work the sample interoperability scenarios used to evaluate the interoperability approach are defined with few details, so that the functionality of the interoperability approach could be tested. Here, the problem is that obtaining all the necessary information necessary for the implementation is very difficult.

Consequently, in future works suitable projects from the practice should be used to evaluate and improve the interoperability approach of the Workflow Management Coalition. The experience gained herein through such implementations can be used to clearly specify critical issues such as the definition of an interoperability scenario or details of an interoperability contract.

Appendix A

The standard Wf-XML DTD

The following Document Type Declaration (DTD) is standardized by the Workflow Management Coalition for the purposes of implementation reference and optional data validation by an XML processor [WfMC00].

```
<!-- Wf-XML DTD, Revision 1.0 Final - 11 April, 2000

        If a DOCTYPE declaration is required to parse this set of declarations, the following line
should be prepended to this file:

        <!DOCTYPE WfMessage PUBLIC "-//WfMC//DTD Wf-XML 1.0//EN"
"http://www.wfmc.org/standards/docs/Wf-XML-1.0.dtd" [

        and the following line appended:

        ]>
-->

<!-- ~~~~~~~~~~~~~~~~~~ Entity Declarations ~~~~~~~~~~~~~~~~~~~ -->

<!-- The ISOLangs entity provides the choices for the ResponseLang attribute of the Request
element. These language codes are taken from the ISO 639:1988 standard, which can be used
for further clarification of the names of each language and can be obtained from
http://www.iso.ch/cate/d4766.html. Additional information is also available at:
http://www.oasis-open.org/cover/iso639a.html. -->
<!ENTITY % ISOLangs
"(aa|ab|af|am|ar|as|ay|az|ba|be|bg|bh|bi|bn|bo|br|ca|co|cs|cy|da|de|dz|el|en|eo|es|et|eu|fa|fi|fj|fo|fr|f
y|ga|gd|gl|gn|gu|ha|hi|hr|hu|hy|ia|ie|ik|in|is|it|iw|ja|ji|jw|ka|kk|kl|km|kn|ko|ks|ku|ky|la|ln|lo|lt|lv|m
```

```
g|mi|mk|ml|mn|mo|mr|ms|mt|my|na|ne|nl|no|oc|om|or|pa|pl|ps|pt|qu|rm|rn|ro|ru|rw|sa|sd|sg|sh|si|s
k|sl|sm|sn|so|sq|sr|ss|st|su|sv|sw|ta|te|tg|th|ti|tk|tl|tn|to|tr|ts|tt|tw|uk|ur|uz|vi|vo|wo|xh|yo|zh|zu)">

<!-- The following two entities are used to define the request and response elements for each
operation. -->
<!ENTITY % OperationRequest "(CreateProcessInstance.Request |
GetProcessInstanceData.Request | ChangeProcessInstanceState.Request |
ProcessInstanceStateChanged.Request)">

<!ENTITY % OperationResponse "(CreateProcessInstance.Response |
GetProcessInstanceData.Response | ChangeProcessInstanceState.Response |
ProcessInstanceStateChanged.Response)">

<!-- The ProcessInstanceData entity defines the properties of a process instance that may be
obtained using the GetPrcoessInstanceData operation. -->
<!ENTITY % ProcessInstanceData "(Name | Subject | Description | State | ValidStates |
ObserverKey | ResultData | ProcessDefinitionKey | Priority | LastModified)+">

<!-- This is the list of valid states defined by the WfMC for version 1.0 of Wf-XML. -->
<!ENTITY % vstates "(open.notrunning | open.notrunning.suspended | open.running |
closed.completed | closed.abnormalCompleted | closed.abnormalCompleted.terminated |
closed.abnormalCompleted.aborted)*">


<!-- ~~~~~~~~~~~~~~~~~ Element Declarations ~~~~~~~~~~~~~~~~~~~ -->

<!-- Root element -->
<!ELEMENT WfMessage (WfTransport?, WfMessageHeader, WfMessageBody)>
<!ATTLIST WfMessage Version CDATA #REQUIRED>

<!-- ~~~~~~~~~~ WfTransport ~~~~~~~~~~~~ -->
<!-- Used for transport-specific information, such as special security or asynchronous
processing. -->
<!ELEMENT WfTransport (CorrelationData?)>

<!ELEMENT CorrelationData (#PCDATA)>

<!-- ~~~~~~~~ WfMessageHeader ~~~~~~~~~~ -->
<!-- Information generally used in all messages, helpful for preprocessing. -->
<!ELEMENT WfMessageHeader ((Request | Response), Key)>

<!ELEMENT Request EMPTY>
<!ATTLIST Request ResponseRequired (Yes | No | IfError) #REQUIRED
              ResponseLang %ISOLangs; #IMPLIED>

<!ELEMENT Response EMPTY>

<!-- The URI of the resource. -->
<!ELEMENT Key (#PCDATA)>

<!-- ~~~~~~~~~ WfMessageBody ~~~~~~~~~~~ -->
```

```
<!ELEMENT WfMessageBody (%OperationRequest; | %OperationResponse;)>

<!ELEMENT CreateProcessInstance.Request (ObserverKey?, Name?, Subject?, Description?,
ContextData)>
<!ATTLIST CreateProcessInstance.Request StartImmediately (true|false) #FIXED "true">

<!ELEMENT ObserverKey (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT Description (#PCDATA)>

<!ELEMENT ContextData ANY>

<!ELEMENT GetProcessInstanceData.Request (ResultDataSet?)>
<!ELEMENT ResultDataSet %ProcessInstanceData;>

<!ELEMENT State %vstates;>
<!ELEMENT ValidStates %vstates;>
<!ELEMENT open.notrunning EMPTY>
<!ELEMENT open.notrunning.suspended EMPTY>
<!ELEMENT open.running EMPTY>
<!ELEMENT closed.completed EMPTY>
<!ELEMENT closed.abnormalCompleted EMPTY>
<!ELEMENT closed.abnormalCompleted.terminated EMPTY>
<!ELEMENT closed.abnormalCompleted.aborted EMPTY>

<!ELEMENT ResultData ANY>

<!ELEMENT ProcessDefinitionKey (#PCDATA)>
<!ELEMENT Priority (#PCDATA)>
<!ELEMENT LastModified (#PCDATA)>

<!ELEMENT ChangeProcessInstanceState.Request (State)>

<!ELEMENT ProcessInstanceStateChanged.Request (ProcessInstanceKey, State,
ResultData?, LastModified?)>

<!ELEMENT ProcessInstanceKey (#PCDATA)>

<!ELEMENT CreateProcessInstance.Response (ProcessInstanceKey | Exception)>

<!ELEMENT GetProcessInstanceData.Response (%ProcessInstanceData; | Exception)>

<!ELEMENT ChangeProcessInstanceState.Response (State | Exception)>

<!ELEMENT ProcessInstanceStateChanged.Response (Exception?)>

<!ELEMENT Exception (MainCode, SubCode?, Type, Subject, Description?)>
<!ELEMENT MainCode (#PCDATA)>
<!ELEMENT SubCode (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
```

Appendix B

The application-specific DTD

The following Document Type Declaration (DTD) defines the necessary elements for the interoperability scenarios that can proceed between a Market Research process and a Store Check process.

```
<!--MarketResearch-StoreCheck Processes Interoperability Scenarios Process Specific Data -->
<!ELEMENT ContextData (TravelReason, TravelDestination, BusinessStart, BusinessEnd, TotalBudgetEuro)>

<!ELEMENT ResultData (TravelReason, TravelDestination, BusinessStart, BusinessEnd, TravelStart, TravelEnd, DecisionMadeBy, DecisionDate, TravelDeviated, DeviationCostEuro, DeviationAcknowledgedBy, TotalTravelCostEuro, PaymentDate, ResponsiblePersonnelDepartment, StoreCheckReport)>


<!ELEMENT TravelReason (#PCDATA)>
<!ELEMENT TravelDestination (#PCDATA)>
<!ELEMENT BusinessStart (#PCDATA)>
<!ELEMENT BusinessEnd (#PCDATA)>
<!ELEMENT TotalBudgetEuro (#PCDATA)>


<!ELEMENT TravelStart (#PCDATA)>
<!ELEMENT TravelEnd (#PCDATA)>
<!ELEMENT DecisionMadeBy (#PCDATA)>
<!ELEMENT DecisionDate (#PCDATA)>
```

```
<!ELEMENT TravelDeviated (#PCDATA)>
<!ELEMENT DeviationCostEuro (#PCDATA)>
<!ELEMENT DeviationAcknowledgedBy (#PCDATA)>
<!ELEMENT TotalTravelCostEuro (#PCDATA)>
<!ELEMENT PaymentDate (#PCDATA)>
<!ELEMENT ResponsiblePersonnelDepartment (#PCDATA)>
<!ELEMENT StoreCheckReport (#PCDATA)>
```

BIBLIOGRAPHY

[Aal98a] van der Aalst W.M.P.: Chapter 10: Three Good Reasons for Using a Petri-net-based Workflow Management System. In T. Wakayama et al., editor, Information and Process Integration in Enterprises: Rethinking documents, The Kluwer International Series in Engineering and Computer Science, Kluwer Academic publishers, Norwell, 1998

[Aal98b] van der Aalst W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers,1998

[Aal99] van der Aalst W.M.P.: Interorganizational Workflows. An Approach based on Message Sequence Charts and Petri Nets,1999

[ALSS00] Alonso G.; Lazcano A.; Schuldt H.; Schuler C.: The Wise Approach To Electronic Commerce. ETH Zentrum, Zürich, Switzerland, 2000

[AnAl01] Anderson M.; Allen R.: Workflow Interoperability - Enabling E-Commerce, Workflow Management Coalition, http://www.wfmc.org, 2001

[Boc93] Bock, G: Workflow as Groupware. A Case for Group Language? Proc. GroupWare ’93, San Jose, 1993

[BrPe84] Bracchi G.; Pernici B.: The Design Requirements Office Systems. ACM Transactions on Office Information Systems, Vol. 2 (1984), No. 2, 151-170

[CiDo01] Cingil I.; Dogac A.: An Architecture for Supply Chain Integration and Automation on the Internet. Middle East Technical University, Ankara, Turkey, 2001

[Dog01a] Dogac A. (Ed.): MARIFlow: A Workflow Management System for Maritime Industry. Middle East Technical University, Ankara, Turkey, Hebrew University, Jerusalem, Israel, 2001

[Dog01b] Dogac A. (Ed.): A Workflow System through Cooperating Agents for Document Flow over the Internet. Middle East Technical University, Ankara, Turkey, Hebrew University, Jerusalem, Israel, 2001

[EeNu93] Ellis C.A.; Nutt G.J.: Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, Application and Theory of Petri Nets 1993, volume 691 of Lecture Notes in Computer Sciences, Springer Verlag, Berlin, 1993

[EGR91] Ellis C.A.; Gibbs S.J.; Rein G.L.: Groupware Some Issues and Experiences. Communications of the ACM, Vol. 43 (1991), No. 1

[EKR95] Ellis C.; Keddara K.; Rozenberg G.: Dynamic Change within Workflow Systems. In N. Comstock and C. Ellis, editors, Conf. On Organizational Computing Systems, ACM SIGOIS, ACM, August 1995

[GHJV95] Gamma E.; Helm R.; Johnson R.; Vlissides J.: Design Patterns, Elements of Reusable Object-Oriented Software. Addison Wesley Longman, 1995

[Gre88] Greif I. (Ed.): Computer Supported Cooperative Work. A Book of Readings. Morgan Kaufmann, San Mateo, CA, 1988

[HaAl98] Hagen C.; Alonso G.: Flexible Exception Handling in the OPERA Process Support System, 18th International Conference on Distributed Computing Systems (ICDCS98), Amsterdam, The Netherlands, 1998

[HaLa91] Hayes K.; Lavery K.: Workflow Management Software: The Business Opportunity. Technical Report, Ovum Ltd., London, 1991

[JaBu96] Jablonski S.; Bussler C.: Workflow Management. Modeling Concepts, Architecture and Implementation, International Thomson Computer Press, 1996

[KaWh96] Kalakota R.; Whiston.A.B. : Frontiers of Electronic Commerce. Addison Westley, Reading, Massachusetts, 1996

[Kou95] Koulopoulus T.M.: The Workflow Imperative. Van Nostrand Reinhold, New York, 1995

[Law97] Lawrence P. (Ed.): Workflow Handbook 1997, Workflow Management Coalition. John Wiley and Sons, New York, 1997.

[McBl91] McCarthy J.C.; Bluestein W.M.: The Computing Strategy Report: Workflow`s Progress. Forrester Research Inc., Cambridge, MA, October 1991

[McDa98] McCarthy D.R.; Dayal U.: The Architecture of Active Database Management System. Proc. ACM SIGMOD, Portland, Oregon, 1989

[MEM94] De Michelis G.; Ellis C.; Memmi G. editors: Proceedings of the second Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Zaragoza, Spain, June 1994

[MLML96] Merz M.; Liberman B.; Mueller-Jones K.; Lamersdorf W.: Interorganizational Workflow Management with Mobile Agents in CSOM. In Proceedings of PAAM96 Conference on the Practical Application of Agents and Multiagent Systems, 1996

[Pet62] Petri C.A.: Kommunikation mit Automaten. Ph.D. thesis, University of Bonn, Germany, 1962

[Pet81] Peterson J.L.: Petri Net Theory and the Modeling of Systems. Prentice-Hall, 1981

[Pre97] Pree W.: Komponenten-basierte Softwareentwicklung with Frameworks. Dpunkt.verlag, Heidelberg, 1997

[ShGa96] Shaw M.; Garlan D.: Software Architecture. Prentice-Hall, New Jersey, 1996

[WfMC93] Workflow Management Coalition: The Workflow Reference Model. Version 0.6, 1993

[WfMC95] Workflow Management Coalition: The Workflow Reference Model. Document Number TC00-1003, January-1995

[WfMC96] Workflow Management Coalition: Workflow Standard – Interoperability Abstract Specification. Document Number WFMC-TC-1012, 26-October-1995

[WfMC98] Workflow Management Coalition: Workflow Management Application Programming Interface (Interface 2&3) Specification. Document Number WFMC-TC-1009, Version 2.0, July-98

[WfMC99a] Workflow Management Coalition: Terminology & Glossary. Document Number WFMC-TC-1011, February-1999

[WfMC99b] Workflow Management Coalition: Interface 1 Process Definition Interchange. Document Number WFMC-TC-1016-P, October-1999

[WfMC00] Workflow Management Coalition Workflow Standard – Interoperability Wf-XML Binding. Document Number WFMC-TC-1023, 1-May-2000

[WfMC01] Workflow Management Coalition Reference Model Interface 4 Workflow Interoperability “The key to E-Commerce and to process scalability”. 2001

[WoRe96] Wolf M.; Reimer U.: Proceedings of the International Conference on Practical Aspects of Knowledge Management (PKAM’96), Workshop on Adaptive Workflow. Basel, Switzerland, October 1996