Entwurf eines Web Services basiertes Workflow Management System (WfMS)

Diplomarbeit
von
Miyoo Tsanang Yves Stephan

November 2006

Betreuer: Prof. Dr. Jacques Calmet
Statement of Originality:
Hereby, I declare this thesis to be the work of my own, written independently. All sources and references are correctly cited with complete references to their sources.

Erklärung:
Hiermit erkläre ich, die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Karlsruhe, den 04.01.2007

Miyoo Tsanang Yves Stephan
Table of contents

1. Introduction .................................................................................................................. - 7 -
   1.1. Context .................................................................................................................... - 7 -
   1.2. Problem statement ................................................................................................. - 7 -
   1.3. Contribution ........................................................................................................... - 8 -

2. A survey on Workflow management system ............................................................... - 10 -
   2.1. Workflow reference model ..................................................................................... - 10 -
      2.1.1. Definitions ......................................................................................................... - 10 -
      2.1.1.1. Workflow ....................................................................................................... - 10 -
      2.1.1.2. Workflow management system (WfMS) ......................................................... - 11 -
      2.1.1.3. Business process(BP) .................................................................................. - 11 -
      2.1.1.4. Workflow participant .................................................................................... - 11 -
      2.1.1.5. Work item ........................................................................................................ - 11 -
      2.1.1.6. Process definition ............................................................................................ - 11 -
      2.1.1.7. Process instances ........................................................................................... - 12 -
      2.1.1.8. Activity ............................................................................................................ - 12 -
   2.2. The workflow model .............................................................................................. - 12 -
   2.3. WfMS application fields .......................................................................................... - 13 -

3. Web services .................................................................................................................. - 14 -
   3.1. Introduction to web services .................................................................................. - 14 -
   3.2. Definition of web services ..................................................................................... - 14 -
   3.3. Business process aspect of web services .............................................................. - 15 -
   3.4. Web services architectures ..................................................................................... - 15 -

4. web services composition .............................................................................................. - 18 -
   4.1. the need of web services composition .................................................................... - 18 -
   4.2. Web services composition survey ......................................................................... - 18 -
   4.3. Definition ................................................................................................................ - 19 -
   4.4. Relevance of dynamic service composition .......................................................... - 19 -
   4.5. Web services compositions issues ......................................................................... - 20 -
   4.6. Web services composition approaches .................................................................. - 20 -
      4.6.1. BPEL4WS ............................................................................................................. - 21 -
      4.6.1.1. Definitions ....................................................................................................... - 21 -
      4.6.1.2. Goals ................................................................................................................ - 21 -
      4.6.1.3. Composition model ........................................................................................ - 21 -
      4.6.2. Semantic Web approaches ............................................................................... - 22 -
4.6.2.1. Definition and vision ............................................................. - 22 -
4.6.3. OWL-S process model ............................................................. - 23 -
4.6.4. Web components ................................................................. - 24 -
4.6.5. Algebraic process composition ............................................... - 25 -
4.6.5.1. Situational calculus ......................................................... - 25 -
4.6.5.2. Petri-Nets (PNs) ............................................................. - 25 -
4.6.5.3. Final state machine .......................................................... - 26 -
5. Design of the WfMS based on web services .................................... - 28 -
5.1. Process definition tools ............................................................. - 28 -
5.1.1. Workflow process definition language .................................... - 29 -
5.1.2. BPEL4WS as workflow process language ............................... - 30 -
5.1.3. BPEL4WS: really the best choice? ....................................... - 31 -
5.1.3.1. Mapping workflow management concepts onto Petri Net (PN) ........................ - 31 -
5.1.4. Understanding the definition of a business process in r-FLOW ....... - 32 -
5.1.5. Web service based process definition tool in r-FLOW and process definition tool architecture... - 37 -
5.2. Workflow enhancement services .............................................. - 39 -
5.2.1. Defining the r-FLOW workflow engine ................................... - 40 -
5.2.1.1. Example of saving a new process in r-FLOW ...................... - 42 -
5.2.1.2. Starting a new instance in r-FLOW .................................. - 43 -
5.3. Administration and monitoring tools ........................................ - 45 -
5.4. Workflow client applications interface ...................................... - 46 -
5.5. Invoked applications ............................................................... - 47 -
5.6. Workflow inter compatibility .................................................. - 48 -
5.7. Architecture of the WfMS ....................................................... - 49 -
6. Application case: using r-FLOW in the HP configuration management process .................................................. - 50 -
6.1. HP Openview RADIA .............................................................. - 50 -
6.2. Policy resolution and ESL ....................................................... - 51 -
6.3. Our task .............................................................................. - 51 -
6.4. Using r-FLOW for triggering RADIA processes ....................... - 53 -
6.4.1. Translating the policy assignment into a process definition language .................................................. - 54 -
7. Related work on WfMS bases on web services .................................. - 57 -
8. Challenges for designing web services based WfMS ....................... - 58 -
8.1. Security ............................................................................. - 58 -
8.2. Transaction ....................................................................... - 58 -
8.3. Standardization ................................................................... - 59 -
9. Conclusion ........................................................................... - 61 -
10. References .......................................................................... - 63 -
List of Figures

Figure 1~relationship between workflow concepts (source: WfMC Terminology & Glossary) ........................................... - 12 -
Figure 2~Workflow reference model-Components and interfaces .................................................................................... - 13 -
Figure 3~web services architecture .................................................................................................................................. - 16 -
Figure 4~WSDL architecture  Figure 5~SOAP architecture  Figure 6~UDDI architecture ............................................. - 17 -
Figure 7~ Petri net composition ....................................................................................................................................... - 26 -
Figure 8~process definition interface .............................................................................................................................. - 28 -
Figure 9~overview of web services technologies ........................................................................................................... - 30 -
Figure 10~web services composition implemented as BPEL4WS (source: www.ibm.com) ....................................... - 31 -
Figure 11~basis process definition meta model (source: WfMC) ...................................................................................... - 33 -
Figure 12~mapping PNs constructs into BPEL4WS activities ......................................................................................... - 36 -
Figure 13~R-FLOW process definition architecture ....................................................................................................... - 38 -
Figure 14~illustration of the r-FLOW process definition .................................................................................................... - 39 -
Figure 15~class diagrams of the workflow engine ........................................................................................................... - 42 -
Figure 16~saving a new Process ......................................................................................................................................... - 43 -
Figure 17~steps at run time .................................................................................................................................................... - 44 -
Figure 18~administration tools ............................................................................................................................................... - 45 -
Figure 19~client application interface ............................................................................................................................... - 46 -
Figure 20~invoked applications ........................................................................................................................................... - 47 -
Figure 21~workflow inter-compatibility ............................................................................................................................. - 48 -
Figure 22~r-FLOW architecture ......................................................................................................................................... - 49 -
Figure 23~RADIA basic workflow model  Figure 24~RADIA workflow model ................................................................. - 51 -
Figure 25~general overview of the solution ......................................................................................................................... - 53 -
Figure 26~process for assigning a patch to a machine in PNs ............................................................................................ - 55 -
Figure 27~BPEL4WS process map view ........................................................................................................................... - 56 -
Figure 28~EWA process definition publisher ..................................................................................................................... - 59 -
Abstract

Workflow Management System (WfMS) is a domain that has evolved from several fields, from business management, modeling, simulation to a lesser extent, planning and control field. It has primarily evolved from the need to understand, i.e. to organize and, often ultimately, to automate processes on which a business is based. In today’s world of extreme competition on the business front, information exchange and efficient communication is the need of the day. At the same time, the advent of the Internet (World Wide Web) and distributed computing technologies has enabled business organizations to conduct business electronically (web services).

The current trend consist on taking advantage of collaborative web services in which business organizations form virtual alliances under rapidly changing market conditions and make use of the best of their resources for achieving their common business goals. Therefore business processes and application system services of the participating organizations are important resources that need to be used to conduct a joint business. Workflow technology is an enabling technology for managing, coordinating, and controlling the activities of virtual enterprises.

The goal of this thesis is to show how it could be possible to take advantage of these two emerging technology (web service and workflow management system). In this thesis it’s been presented an approach consisting on designing a WfMS entirely based on web services. The issue encountered by most WfMS is the choice of a workflow language. Despite the lack of a standard workflow language, a technique to combine BPEL4WS and Petri Nets (PNs) in order to tackle workflow language’s semantic issue is presented in this thesis. The concepts proposed here have been used in the design of R-FLOW as web services based WfMS.
CHAPTER 1

INTRODUCTION AND OVERVIEW

1. Introduction

1.1. Context
Workflow management system (WFMS) is a domain that has evolved from several fields, business management, modeling, simulation and, to a lesser extent, planning and control. It has primarily evolved from the need to understand, to organize and, often ultimately, to automate the processes on which a business is based. In today’s world of extreme competition on the business front, information exchange and efficient communication is the need of the day. Information Technology has grown by leaps and bounds, and sustained, not because it seems savvy, but because businesses can work more efficiently. Thus it’ll be emphasized in the future on the ability for WFMS to provide a certain level on flexibility and integration.

The advent of the Internet, World Wide Web, and distributed computing technologies has enabled business organizations to conduct business electronically: thus, the e-business (web services) was born. According to a white paper from Morgan Stanley Dean Witter, there have been three major phases in the evolution of web services technology. Web services first appeared first in the form of EDI (Electronic Data Interchange). The second phase of web services was basic E-commerce, where retailers sell their products through their Web sites. Phase three of web services, currently in progress, is in the form of e-Marketplace “portals,” which bring trading partners with related interests together into a common community (i.e., an exchange) to match buyers and sellers and provide other services to serve their interests (this is the emergence of web services composition).

The current trend is to take advantage of collaborative web services in which business organizations form virtual alliances under rapidly changing market conditions and make use of the best of their resources for achieving their common business goals. Business processes and application system services of the participating organizations are important resources that need to be used to conduct a joint business. Workflow technology is an enabling technology for managing, coordinating, and controlling the activities of a virtual enterprise.

1.2. Problem statement
The Workflow is a concept closely related to the reengineering and the automation of businesses and information processes.

Today a great number of commercial Workflows and academic prototypes have been available on the market and in the literature. The emergence of information systems such as the Internet, the mobile communication and the enterprise information systems (i.e. ERP and Supply Chain systems), becomes rapidly a growing area and gaining more and more substantial attention not only in the interrelation of electronic business applications (B2B, e-commerce, B2C) but also in the cross enterprises collaborations. The challenges for the workflow management systems rely on their abilities to align Information and Communication Technologies (ICT) and the new paradigms of businesses. The
evolution of the technology should not make the legacy enterprise applications hard to maintain and to cooperate with partners applications outside the enterprise boundaries. Nevertheless WfMS should take into consideration new paradigms of the business.

Unfortunately, this trend does not shape the current workflow architectures. Many lacks and different limitations arise around the current WfMS architectures. We state the following problems:

- The early workflow management systems were monolithic, they are considered as black boxes. The extensions of their functionalities were difficult. Most of today workflows are centralized and based on 2-tier architecture (Client/Server). The robustness of the workflows depends on the server side; during the maintenance or the upgrade period, the system is inaccessible. By using components and persistence objects, the workflow architectures become modular, easy to maintain and distributed over the intranet. Whilst, the development platforms are vendors own (J2EE, .NET, CORBA, etc) and thus there are not natively compatible.

- Another problem arises from the interoperability between the workflows and the applications invoked by the workflow engine during the business processes execution. The web services technology sounds a good solution to solve the interoperability problems and they act as neutral interfaces to invoke applications. Yet, they are far away to respect a common reference model. The WfMC initiative attempts to propose a reference model for the workflow community. As we notice, many workflows conform partially to the reference model or they do not agree on it at all.

- Even if some workflows are generic, they need to be customized (i.e. using configuration files) in order to support the business processes inside the enterprise. The new age of the electronic business applications (B2B, e-commerce, B2C), changes radically the management of workflows. E-business applications become dependant on the cyber partner’s collaboration, the inter-workflows interoperability and the enterprise’s agility. They are no more only dependant on a specific enterprise and limited to its boundaries.

- The workflow management systems are strongly dependant (and not exhaustively) on: 1) the programming languages (i.e. Java, C++, etc) 2) the operating systems (MS Windows OS, Linux, UNIX, Mac, etc), 3) the networking protocols (i.e. protocols based TCP/IP), 4) the Remote Call protocols (IIOP, RPC, RMI, etc) and 5) the database management systems (Oracle, MySQL, SQL-server etc). It is clear that a platform independent architecture of workflows empowers the development and facilitates the enactment of the workflows without vendor’s dependency.

1.3. Contribution

These limitations and lacks of some workflow management motivated us to design a new architecture for conceiving and developing workflow engines. The new architecture should respond to the new business paradigms and take full advantages of information systems. We outline the problematic with the following statement: **How could we develop a dynamic, flexible and platform independent Workflow engine based on web services and on a Common Reference Model?**

Along this thesis, we attempt to answer this statement. Thus, we propose a generic architecture for workflow management system based entirely on layers of loosely coupled web services. Furthermore we’ll focus on the functional (i.e. functional decomposition of activities), behavioral (i.e. execution order and dependencies of activities in the workflow), operational (i.e. how the workflow management system should interact with its environment), informational (i.e. data used in the workflow and the data dependencies between activities) and organizational (who should do the work in the workflow) aspects instead of presenting a robust implemented program code.
Since our aim is to design a workflow management system, we start this thesis by giving a survey (or state of the art) on WfMS. Then we also present a survey on web services and web services composition before the relationship that exists between these two domains. After that we’ll present the design of our WfMS and we’ll also present a use case (based on the HP configuration management process). At the end we’ll outline an analysis on the designed system before finally ending this thesis.
CHAPTER 2
STATE OF THE ART ON WORKFLOW MANAGEMENT SYSTEM

2. A survey on Workflow management system

The first available workflow management systems were developed in the late 1980s. Only few of these first generation systems are left in the market. Currently available workflow management systems are ordered as a standalone product or are positioned as part of a process-enabled application system (for example in Enterprise Resource planning systems or in Enterprise Application Integration systems).

Workflow technology allows people and companies to model business processes and to control the execution of these processes. Based on the definition given by the Workflow Management Coalition (WfMC), a business process is a set of activities that collectively realizes a business goal, normally within the context of an organization. Workflow is the automation of a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for actions according to a set of procedural rules. A workflow management system (WfMS) defines, creates, and manages the execution of workflows.

2.1. Workflow reference model

The Workflow Management Coalition (WfMC) is a non profit organization with the objectives of advancing the opportunities for the exploitation of workflow technology through the development of common terminology and standards. It has been recognized that all workflow management products have some common characteristics, enabling them potentially to achieve a level of interoperability through the use of common standards for various functions. The WfMC has been established to identify these functional areas and develop appropriate specifications for implementation in workflow products. 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 within the IT market and in the research, to the benefit of both vendors and users of such technology.

We give the following definition as described in the terminology and glossary of the WfMC in [18].

2.1.1. Definitions

2.1.1.1. Workflow

Workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of
procedural rules. Workflow is defined within a Process Definition, which identifies the various process activities, procedural rules and associated control data used to manage the workflow during process enactment.

2.1.1.2. Workflow management system (WFMS)

It’s 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 externals tools and applications. We can also say that a Workflow Management System consists of software components which purposes are to store and interpret process definitions, create and manage workflow instances as they are executed, and control their interaction with workflow participants and applications. Common synonyms are workflow management, workflow computing or case computing.

2.1.1.3. Business process (BP)

A Business process is 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.
A business process is typically associated with operational objectives and business relationships, for example an Insurance Claims Process, or Engineering Development Process. A process may be wholly contained within a single organizational unit or may span several different organizations, such as in a customer-supplier relationship. It may also consist of automated activities, capable of workflow management, and/or manual activities, which lie outside the scope of workflow management.

2.1.1.4. Workflow participant

It’s a resource which performs the work represented by a workflow activity instance. This work is normally manifested as one or more work items assigned to the workflow participant via the work list. The term Workflow Participant is normally applied to a human resource but it could conceptually include machine based resources such as an intelligent agent.

The term Workflow Participant is normally applied to a human resource but it could conceptually include machine based resources such as an intelligent agent. It may be identified directly within the business process definition, or (more normally) is identified by reference within the process definition to a role or organizational entity, which can then be filled by one or more of the resources available to the workflow system to operate in that role during process enactment.

2.1.1.5. Work item

It means the representation of the work to be processed (by a workflow participant) in the context of an activity within a process instance. An activity typically generates one or more work items which together constitute the task to be undertaken by the user (a workflow participant) within this activity (In certain cases an activity may be completely handled by an invoked application which can operate without a workflow participant, in which case there may be no work item assignment).

2.1.1.6. Process definition

It means the representation of a business process in a form which supports automated manipulation, such as modeling, or enactment by a workflow management system. The process definition consists of a set or network of activities and their relationships, criteria to emphasize the start and termination of
the process, and information about the individual activities, such as participants, associated applications and relevant data, etc.

2.1.1.7. Process instances

It is the representation of a single enactment of a process. A process instance is created, managed and (eventually) terminated by a workflow management system, in accordance with the process definition.

2.1.1.8. Activity

It is 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.

The following picture gives an overview on the above definitions.

![Workflow Model Diagram]

Figure 1–relationship between workflow concepts (source: WfMC Terminology & Glossary)

2.2. The workflow model

According to the WfMC group [17], the Workflow Reference model has been developed from the generic workflow application structure by identifying the interfaces within this structure which enable products to interoperate at a variety of levels. All workflow systems contain a number of generic components which interact in a defined set of ways; different products will typically exhibit different levels of capability within each of these generic components. To achieve interoperability between workflow products a standardized set of interfaces and data interchange formats between such
components is necessary. A number of distinct interoperability scenarios can then be constructed by reference to such interfaces, identifying different levels of functional conformance as appropriate to the range of products in the market.

The following figure illustrates the major components and interfaces within the WfMS architecture.

![Workflow reference model-Components and interfaces](image)

2.3. WfMS application fields

The evolution of workflow as technology has encompassed a number of different areas:

- Image processing
- Document management
- Electronic mail and directories
- Groupware applications
- Transactions based applications
- Project support software
- Business process re-engineering (BPR)
- BPR and structured system designed tool
- Separation of workflow functionality
- ...

The good thing is that, despite the variety in workflow products in the markets, it’s been proved feasible to construct a general implementation model of a WfMS which can be matched to most products in the marketplace thereby providing a common basis for developing inter-operable and compatible scenarios.
CHAPTER 3
INTRODUCING WEB SERVICES

3. Web services

3.1. Introduction to web services
The revolution of computerizing services of companies gave rise to isolated computer systems. Each company had software developed and customized to its specific needs. However, mergers, acquisitions, and business growths saw the need to share information stored in these isolated computer systems. The Internet did solve this problem to some extent. However, the Internet also opened many loopholes in security, making the owners of this information uneasy about the scope of their information's availability.
Hence, it became imperative that, for better B2B (Business-to-Business) communication and integration, these systems must have the ability to link up to each other, grant permissions through a system other than the Internet, and which would make all the systems network with each other like an Intranet.
For example, with the introduction of the telephone came the need to have a directory service. This gave rise to the ever-popular "Yellow Pages," which brought the consumer and the provider closer to each other.

3.2. Definition of web services
The term “Web services” is used very often nowadays, although not always with the same meaning. Nevertheless, the underlying concepts and technologies are to a large extent independent of how they may be interpreted. Existing definitions range from the very generic and all-inclusive to the very specific and restrictive. Often, a Web service is seen as an application accessible to other applications over the Web. This is a very open definition, under which just about anything that has a URL is a Web service. It can, for instance, include a CGI script. It can also refer to a program accessible over the Web with a stable API, published with additional descriptive information on some service directory.
We give here the definition retain by the World Wide Web Consortium (W3C).
A web service is a software system:
- Designed to support interoperable machine-to-machine interaction over a network.
- It has an interface described in a machine readable format (WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages.
- Typically accessible using HTTP with an XML serialization in conjunction with other Web-related standards.
Therefore technologies such as RMI, COM, CORBA, EDI, and ebXML also address the same problem space as web services. If so, what would make Web Services so special and different from the rest?
Web Services is based on the already existing and well-known HTTP protocol, and uses XML as the base language. This is the reason why it’s been widely considered as a very developer-friendly service system. However, most of the technologies mentioned above such as RMI, COM, and CORBA involve a whole learning curve. New technologies and languages have to be learnt to implement these services. Also, Web Services is based on a set of standardized rules and specifications, making it more portable. This is not the case with the technologies we mentioned earlier.

3.3. Business process aspect of web services
Web Services in the business world, in the most simplistic fashion, provides a mechanism of communication between two remote systems, connected through the network of the Web Services. For example, in case of a merger or an acquisition, companies don’t have to invest large sums of money developing software to bring the systems of the different companies together. By extending the business applications as Web Services, the information systems of different companies can be linked. These business systems then can be accessed by using simple SOAP messages over the normal HTTP Web protocol. For example, a manufacturing company requires some raw materials to be supplied whenever the material in stock reaches the threshold levels. These levels can be constantly monitored by the business system of a trusted supplier, and promptly replenished, without having to wait for a supervisor to notice it and generate a work order.

Additionally, there are many more important uses of Web Services. These, again, depend on the requirement of the company.

3.4. Web services architectures
Client/server applications in use in the 80s and early 90s were primarily data-centric in nature. The focus was more on retrieval and display of data than on the business rules of the application. Web application architectures changed all this. Because Web applications are based on a distributed architecture, Web applications have moved away from the data-centric nature of client/server applications and tended more to a service-oriented architecture (SOA). The basic distributed technologies from Remote Procedure Calls (RPC), to Remote Method Invocation (RMI), to Common Object Request Broker Architecture (CORBA) are inherently service-oriented. Web applications have evolved from simple applications accessed and deployed in an Intranet to applications of different businesses/organizations "talking" to each other and exchanging data. The next step for Web applications is to provide business services that can be accessed by other applications of different businesses/organizations using the service-oriented architecture of Web Services. Moreover, the advantages of re-use in distributed Web applications have resulted in applications being designed to provide more business-related services.

When we talk about service-oriented architecture for Web Services applications, quite a few things come up. The application providing a service and the client application using the service talk to each other in a common language. Without both applications talking in the same language and exchanging information, the entire architecture will turn into one more addition to the tower of Babel of distributed enterprise applications!

Next, a service providing application and a client application using the service need some way to locate each other before they start talking with each other. This is especially true for distributed applications, where applications have no knowledge of each other's location.

Hence, we can conclude that a basic service-oriented architecture for Web Services has:

- a standard way for communication
Let’s try to make it clearer through an example of the daily life:
We consider a scenario in which it’s needed to locate a particular pharmacy store in our area. It’s not common to go out on the road and ask every person that we meet the way to the store. We might, instead, refer the web site of the pharmacy on the Internet. If we knew the pharmacy’s web site, we would look it up directly and find the location through the store locator link. If not, we would go to a search engine and type out the name of the pharmacy in the language that the search engine was meant to recognize. After getting the location, we would find the direction to the store, and then go to the store.
The structure of Web Services is also very similar. Web Services provide for each of these previously described activities.
If we carefully look at the preceding example, we will see that there is a requestor or a consumer—that is us. There is also a service, the pharmacy store. The central database of information is the Internet, through which we find the location of the pharmacy. In the example, when we fire a search in the search engine, our request is wrapped in a structure, whose language is predetermined and localized, and then passed onto the server running the search engine.

In Web Services, SOAP, UDDI, and WSDL represent the roles mentioned in these steps.
- SOAP (Simple Object Access Protocol): is the method by which you can send messages across different modules. This is similar to how you communicate with the search engine that contains an index with the Web sites registered in the index associated with the keywords.
- UDDI (Universal Description, Discovery, and Integration): is the global lookup base for locating the services. In the example mentioned earlier, this is analogous to the index service for the search engine, in which all the Web sites register themselves associated with their keywords. It maintains a record of all the pharmacy store locations throughout the country.
- WSDL (Web Services Definition Language): is the method through which different services are described in the UDDI. This map to the actual search engine in our example.

The figures below illustrate what we said above.

---

Figure 3: Web services architecture
Figure 4 – WSDL architecture  Figure 5 – SOAP architecture  Figure 6 – UDDI architecture
CHAPTER 4

INTRODUCING WEB SERVICES COMPOSITION TECHNIQUES

4. web services composition

4.1. the need of web services composition

After reviewing the web services features it seems clear to us that web services are stateless. It consists most of the time of a piece of software that contains a method which can be remotely called via an interface. For our workflow purpose, we need more. We want web services to communicate together, exchange data, and take more responsibility.

From a business point of view, web services composition represent a workflow of activities that means we could use it to model business processes.

Thus in our WfMS environment, we need a web services orchestrator or coordinator, that will guarantees that the services are not coupled tightly.

Before choosing a web services composition method that we’ll use for modeling our business process, we need to compare and analyze the techniques that exist on web services compositions.

4.2. Web services composition survey

The basic web services infrastructure as previously presented suffices to implement simple interactions between a client and a web service. If the implementation of a web service’s business login involves the invocation of other web services, it is necessary to combine the functionality of several web services. In this case, we speak of a composite service. The process of developing a composite service in turn is called service composition. Service composition can either be performed by composing elementary or composite services.

When composing web services, the business logic of the client is implemented by several services. This is analogous to workflow management, where the application logic is realized by composing autonomous applications. This allows the definition of increasingly complex applications by progressively aggregating components at higher levels of abstraction. A client invoking a composite service can itself be exposed as a web service.

Since it is a widely used approach to use conventional programming languages to link components to a composite web service, and thereby bridge heterogeneous middleware platforms, it becomes necessary to develop a Service Composition.

As the number of available Web services continues to grow and as the business and academic environment keeps demanding newer applications that have to be rolled out according to very tight schedules, both the opportunity and the need for service composition technologies provided as part of
the Web services middleware arises. These technologies resemble those of workflow systems, and have the potential to enable the rapid development of complex services from basic ones, as well as to simplify the maintenance and evolution of such complex services.

4.3. Definition

A Web service can be implemented by invoking other Web services, possibly provided by different companies. For example, a reseller of personal computers may offer a Web service that allows customers to request quotes and order computers. However, the implementation of the requestQuote operation may require the invocation of several Web services, including for example those provided by PC manufacturers and shippers, for the latest prices and delivery schedules. A Web service implemented by invoking other Web services is called a composite service, while a Web service implemented by accessing the local system is called basic service. Observe that whether a Web service is basic or composite is irrelevant from the perspective of the clients, as it is only an implementation issue. In fact, they are all Web services and can be described, discovered, and invoked in the same way.

As resume we could say that composed web services are:
- Individual components implemented at different places
- Execute in different contexts
- But need to communicate to yield desired behavior

This means that the main task of web services composition is to make a set of services into a single service. Thus web services composition are collection of:
- Services that collaborate in order to realize a request.
- Services arranged into a workflow.
- Operations arranged into a single execution.

4.4. Relevance of dynamic service composition

The process of creating new services at runtime from a set of service components is called dynamic service composition. If other businesses provide newer services, or the old services are replaced by other ones, inconsistencies might be caused. In that case, it is unavoidable to change the software architecture and bind to other services or, in the worst case, even change the process definition and redesigns the system. In this case, static composition may be too restrictive and components should automatically adapt to unpredictable changes.

With dynamic service composition complex services are autonomously created by combining primitive components on the fly based on user request and context. We obtain more flexible and adaptability than statically deployed applications.

Thus the potential benefits of dynamic service composition are: greater system flexibility, and availability:
- New services, potentially unanticipated during design time, can be constructed to address a specific problem, on demand of the system or its users
- A relatively large number of services can be constructed from a set of basic service components
- Minimization of human involvement and minimization of system disruption to perform upgrades and addition of new functionality
4.5. Web services compositions issues

Before presenting here some web services compositions techniques, there are many questions that need to be answered, in other words, we have to go through many barriers (technically, syntactically…).

How should we model business process? How to model available services? How to model and orchestrate composite services?

Intuitively, we can regroup all these questions under the followings categories:

- representation of a abstract web process
  - representing/specifying the abstract process in a proper form
- discovery and interoperability of Services
  - need to manually or automatically search for appropriate services
  - the discovered services should interoperate
- process execution
  - adapting a suitable algorithm for executing the composed concrete process
- process monitoring
  - using a monitoring technique for run time analysis of the web process execution
- efficiency of a composed web process need to compose processes which are efficient in terms of performance

4.6. Web services composition approaches

Nowadays we the growing use of web services thus of web services compositions, there are many composition approaches based on various perspectives.

Nowadays we the growing use of web services thus of web services compositions, there are many composition approaches based on various perspectives. After searching the literature we can identify five categories of composition strategies:

- Static vs. Dynamic Composition
- Manual vs. Automated Composition
- Declarative Composition
- Model driven Composition
- Business Rule driven Composition
- Context-based Composition

Static and dynamic composition strategies concern the time when Web services are composed. They are equivalent to design-time and run-time composition. Among other approaches for service compositions are: model driven, business rule driven service composition and declarative composition as well. Another topic concerns automated and manual service composition. It exists a survey on ontology based composition and semantic Web services composition as counterpart to manual composition techniques (such as provided by BPEL).

In the following pages we will not follow the logic or the order of the above given approaches (Static vs. Dynamic Composition, manual vs. automated composition …) to present web services composition techniques. It seems to us more interesting to present the most widely used approaches so that a concrete feeling about web services composition can be created and furthermore having an idea about the composition technique that we’ll use for our WfMS.

Thus we’ll present the BPEL approach, the semantic web, the web component, the algebraic process composition, and finally the Petri Net based approach.
4.6.1. BPEL4WS

4.6.1.1. Definitions

In July 2002, BEA, IBM, and Microsoft released a trio of specifications designed to support business transactions over Web services. These specifications, BPEL4WS, WS-Transaction, and WS-Coordination, together form the bedrock for reliably choreographing Web services based applications, providing business process management, transactional integrity, and generic coordination facilities respectively.

The value of BPEL4WS is that if a business is the sum of its processes, the orchestration and refinement of those processes is critical to an enterprise’s continued viability in the marketplace. Those businesses whose processes are agile and flexible will be able to adapt rapidly to and exploit new market conditions.

This are the key features of Business Process Execution Language for Web Services, and shows how it builds on the features offered by WS-Coordination and WS-Transaction to support the reliable orchestration of business processes.

4.6.1.2. Goals

We present here some goals that are reflected in the charter of WSBPEL TC.

- Goal 1: BPEL4WS should define business processes that interact with external entities through Web service operations defined using WSDL 1.1 and that manifest themselves as Web services defined using WSDL 1.1. The interactions are “abstract” in the sense that the dependence is on port type definitions, not on port definitions.
- Goal 2: BPEL4WS defines business processes using an XML based language. BPEL4WS is not concerned with the graphical representation of processes nor defines any particular design methodology for processes.
- Goal 3: BPEL4WS should define a set of Web service orchestration concepts that are meant to be used in common by both the external (abstract) and internal (executable) views of a business process. Such a business process defines the behavior of a single autonomous entity, typically operating in interaction with other similar peer entities. It is recognized that each usage pattern (i.e. abstract view and executable view) will require a few specialized extensions, but these extensions are to be kept to a minimum and tested against requirements such as import/export and conformance checking that link the two usage patterns.
- Goals 4: BPEL4WS should provide both hierarchical and graph-like control regimes, and allow their usage to be blended as seamlessly as possible. This should reduce the fragmentation of the process modeling space.
- Goal 5: BPEL4WS provides limited data manipulation functions that are sufficient for the simple manipulation of data that is needed to define process relevant data and control flow.

4.6.1.3. Composition model

BPEL composition interacts with a Web service’s subset to achieve a given task. In BPEL, the composition result is called a process, participating services are partners, and message exchange
or intermediate result transformation is called an **activity**. A process thus consists of a set of activities. A process interacts with external partner services through a WSDL interface. To define a process, we use:

- A BPEL source file (.bpel), which describes activities;
- A process interface (.wsdl), which describes ports of a composed service; and an optional deployment descriptor (.xml), which contains the partner service’s physical locations (a partner service’s implementation and location can be changed without modifying the source file).

In fact, BPEL\(^1\) has several group elements, but the basic ones are:

- **process initiation:**
  \[
  \text{process} > \text{...} / \text{process} \equiv
  \]

- **definition of services participating in composition:**
  \[
  \text{partnerlink} \equiv
  \]

- **synchronous and asynchronous calls:**
  \[
  \text{porttype} \equiv \text{operation} \equiv \text{input variable} \equiv \text{output variable} \equiv \text{receive partnerlink} \equiv \text{...}/ \equiv
  \]

- **intermediate variables and results manipulation:**
  \[
  \text{variable} \equiv \text{assign} \equiv \text{copy} \equiv
  \]

- **error handling:**
  \[
  \text{faulthandlers} \equiv
  \]

- **sequential and parallel execution:**
  \[
  \text{sequence} \equiv \text{flow} \equiv
  \]

- **logic control:**
  \[
  \text{switch} \equiv
  \]

### 4.6.2. Semantic Web approaches

#### 4.6.2.1. Definition and vision

The Web, solely seen as a repository for text and images, is evolving into a provider of services, such as flight information providers, temperature sensors, *world-altering services* such as flight-booking programs, sensor controllers, and a variety of e-commerce and business-to-business applications. Web-accessible programs, databases, sensors, and a variety of other physical devices realize these services.

By “service” we mean Web sites that do not merely provide statics information but allow one to effect some action or change in the world, such as the sale of a product or the control of a physical device. Today’s Web was designed primarily for human interpretation and use. Nevertheless, we are seeing increased automation of Web service interoperation, primarily in B2B and e-commerce applications. Generally, such interoperation is realized through APIs that incorporate hand-coded information-extraction code to locate and extract content from the HTML syntax of a Web page presentation layout. Unfortunately, when a Web page changes its presentation layout, the API must be modified to prevent failure.

The Semantic Web should enable users to locate, select, employ, compose, and monitor Web-based services automatically. The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. This is realized by marking up

---

\(^1\) we’ll give concrete BPEL4WS examples when we’ll design our WfMS engine
Web content, its properties, and its relations, in a reasonably expressive markup language with a well-defined semantics.

The Semantic Web vision is to make Web resources accessible by content as well as by keywords. Web services play an important role in this: users and software agents should be able to publish, discover, select, compose, invoke and execute content using complex services.

4.6.3. OWL-S process model

The Ontology Web Language for Services (OWL-S) provides developers with a strong ontological language to describe the properties and capabilities of their Web Services in such a way that the descriptions can be interpreted by a computer system in an automated manner. The information provided by an OWL-S description of a Web Service includes an ontological description of the inputs required by the service, the outputs it provides, and the pre- and post-conditions of each invocation. OWL-S (previously known as DAML-S) is a service’s ontology that enables automatic service discovery, invocation, composition, interoperation, and execution monitoring.

OWL-S models services using three-part ontology:
- A service profile;
- A service model;
- Service grounding.

The service profile tells "what the service does", in a way that is suitable for a service-seeking agent (or matchmaking agent acting on behalf of a service-seeking agent) to determine whether the service meets its needs. This form of representation includes a description of what is accomplished by the service, limitations on service applicability and quality of service, and requirements that the service requester must satisfy to use the service successfully.

The service model tells a client how to use the service, by detailing the semantic content of requests, the conditions under which particular outcomes will occur, and, where necessary, the step by step processes leading to those outcomes. That is, it describes how to ask for the service and what happens when the service is carried out. For nontrivial services (those composed of several steps over time), this description may be used by a service-seeking agent in at least four different ways: (1) to perform a more in-depth analysis of whether the service meets its needs; (2) to compose service descriptions from multiple services to perform a specific task; (3) during the course of the service enactment, to coordinate the activities of the different participants; and (4) to monitor the execution of the service.

A service grounding ("grounding" for short) specifies the details of how an agent can access a service. Typically grounding will specify a communication protocol, message formats, and other service-specific details such as port numbers used in contacting the service. In addition, the grounding must specify, for each semantic type of input or output specified in the Service model, an unambiguous way of exchanging data elements of that type with the service (that is, the serialization techniques employed).

The upper ontology for services specifies only two cardinality constraints: a service can be described by at most one service model, and grounding must be associated with exactly one service. The upper

---

2 Ontologies describe the hierarchical organization of ideas in a domain in a way that can be parsed and understood by a software program. They are similar to object-oriented classes in a software program (without the runtime functional components). We’d like to mention that the FZI in Karlsruhe has developed the KAON (Karlsruher Ontology) tool.
ontology deliberately does not specify any minimum cardinality for the properties presents or described by. (Although, in principle, a service needs all three properties to be fully characterized, it is easy to imagine situations in which a partial characterization could be useful.) Nor does the upper ontology specify any maximum cardinality for presents or supports. (It will be extremely useful for some services to offer multiple profiles and/or multiple groundings.)

Figure 5~ upper ontology

4.6.4. Web components

Web components are a packaging mechanism for developing web-based distributed applications in terms of combining existing (published) web services. Web components have a recursive nature in that they can be composed of published web services while in turn they are also considered to be themselves web services (albeit complex in nature). Once a web component class is defined, it can be reused, specialized, and extended. The same principle applies to service composition activities if we view a composite service as a special web service, which contains composition constructs and logic.

Figure 6~ web component ingredient
4.6.5. **Algebraic process composition**

"Process Algebra = algebra about processes!"

Algebraic service composition aims to introduce much simpler descriptions than other approaches, and therefore to model services as mobile processes in order to ensure verification of properties such as safety, liveness (correct termination, for example), and resource management.

4.6.5.1. **Situational calculus**

Mobile-processes theory is based on calculus, in which the basic entity is a process — it can be an empty process, a choice between several I/O operations and their continuations, a parallel composition, a recursive definition, or a recursive invocation. I/O operations can be \`receive\` or \`send\`.

For example:

- \`x(y)\` denotes receiving tuple \(y\) on channel \(x\)
- \`x \_\{y\}\` denotes sending tuple \(y\) on channel \(x\).

Dotted notation specifies an action sequence, such as \`c \_\{1, d\}.d(x, y, z).c \_\{x – y – z\}\` in which a process sends tuple \(\{1, d\}\) on channel \(c\), then receives a tuple at channel \(d\) whose components are bound to the variables \(x\), \(y\), and \(z\), and finally sends the sum of \(x+y+z\) to channel \(c\).

Parallel process composition is denoted with \`A | B\`. Several processes can execute in parallel and communicate using compatible channels.

Describing services in such an abstract way lets us reason about the composition’s correctness. Thus using calculus, we can describe our example composition as

\[ A(\text{processInput}).B \_\{A0\text{output}\}.C \_\{A0\text{output}\}|B(\text{BInput}).out \_\{B0\text{output}\}|C(\text{CInput}).outC\text{Output}|put(\text{processOutput}) \]

With algebraic process composition, the general question is what information to type. Typing too little can make it impossible to verify some properties, such as security. On the other hand, typing too much creates a complexity that renders verification unusable or impractical.

4.6.5.2. **Petri-Nets (PNs)**

A Petri net (Petri 1962, Peterson 1981) is an algebraic structure \((P, T, I, O)\) composed of:

- Finite set of places, \(P \equiv \{p_1, \ldots, p_n\}\)
- Finite set of transitions, \(T \equiv \{t_1, \ldots, t_m\}\)
- Transition input function, \(I\) maps each transition \(t_i\) to a multi set of \(P\)
- Transition output function, \(O\) maps each transition \(t_i\) to a multi set of \(P\)

Therefore PNs are a well-founded process modeling techniques that have formal semantics. They have been used to model and analyze several types of processes including protocols, manufacturing systems, and business processes (Alas 1999). A Petri net is a directed, connected, and bipartite graph in which each node is either a place or a transition. Tokens occupy places. When there is at least one token in every place connected to a transition, we say that the transition is enabled. Any enabled transition may remove one token from every input place, and depositing one token in each output place. The use of visual modeling techniques such as Petri nets in the design of complex
Web services is justified by many reasons. For example, visual representations provide a high-level yet precise language which allows expressing and reasoning about concepts at their natural level of abstraction.

A Web service behavior is basically a partially ordered set of operations. Therefore, it is straightforward to map it into a Petri net. Operations are modeled by transitions and the state of the service is modeled by places. The arrows between places and transitions are used to specify causal relations. We can categorize Web services into material services (e.g., delivery of physical products), information services (create, process, manage, and provide information), and material/information services, the mixture of both. We assume that a Petri net, which represents the behavior of a service, contains one input place (i.e., a place with no incoming arcs) and one output place (i.e., a place with no outgoing arcs). A Petri net with one input place, for absorbing information, and one output place, for emitting information, will facilitate the definition of the composition operators (see Section 3.2) and the analysis as well as the verification of certain properties (e.g., reachability, deadlock, and aliveness). At any given time, a Web service can be in one of the following states: Not Instantiated, Ready, Running, Suspended, or Completed.

When a Web service is in the Ready state, this means that a token is in its corresponding input place, whereas the Completed state means that there is a token in the corresponding output place. We can model services as Petri nets by assigning transitions to methods and places to states. Each service has an associated Petri net that describes service behavior and has two ports: One input place and one output place.

After we define a net for each service, composition operators perform composition: sequence, alternative (choice), unordered sequence, and iteration, parallel with communication, discriminator, selection, and refinement. These operators guarantee the closure property. Thus, by composing two or more Web services, we produce another service.

Let ⊗ be a sequence operator, for example, and || be a parallel operator with communication. We can then write our example as $A \otimes (B || C)$ where $A$, $B$ and $C$ are 3 web services. We use a parallel operator with communication to compare and select between outputs of services $B$ and $C$. Graphically, our service would look like in the following figure.

![Petri net composition](image)

After specifying composition with a Petri net, we can use it to prove some algebraic properties, such as absence of deadlocks or live locks (whether composition will terminate in a finite number of steps). Correct termination is very important for composed service; we verify this property by determining whether the Petri net is live and bounded.

4.6.5.3. Final state machine
Other approaches for Web service composition include model checking, modeling service composition as Mealy machines (described below), and automatic composition of finite-state machines (FSMs). Model checking is used to formally verify finite-state concurrent systems. We describe system specification using temporal logic, then traverse and check the model to see whether the specification holds. We can apply model checking to Web service composition by verifying correctness inside a workflow specification. Among the properties we can check are data consistency, unsafe state avoidance (deadlock), and business-constraint satisfaction. Researchers have also proposed the conversation specification for Web service composition. According to this approach, understanding constituent services’ local behavior and the composed service’s global behavior are important to verifying and guaranteeing correctness. The approach models services as Mealy machines, which are FSMs with input and output. Services communicate by sending asynchronous messages, and each service has a queue. A global “watcher” keeps track of all messages. The conversation is introduced as a sequence of messages. By studying and understanding conversation properties, the method provides new approaches for designing and analyzing “well-formed” service composition. Automatic Web services composition is the ultimate goal of most composition efforts. In the process, the composition is proved correct and the algorithm’s computational complexity characterization is given, ensuring that the automatic composition will finish in the finite number of steps.
CHAPTER 3
R-FLOW: WFMS BASED ON WEB SERVICES

5. Design of the WfMS based on web services

Previously in this document we’ve introduced some key concepts on web services, web services composition and workflow management systems. Here we started the core of our work which consists on combining these technologies for a common goal.

We propose the design of a new type of WfMS: r-FLOW (RADIA flow). R-FLOW’s architecture is entirely based on web services in order to take advantage of web services mobility or flexibility, and furthermore at the same time to minimize all the disadvantages we’ve mentioned at the beginning of this work [1.2].

At this point we would like to outline the fact we are designing an intranet based WfMS and as such, most of his function are designed regarding to some specific business constraints. Nevertheless (and therefore more interesting) it is also our contribution to the growing discussion in the WfMS research world on using web services in the context of workflow. Thus we’ve tried to design a workflow as specified by the WfMC standard. In the following section we’ll present the design of each part of our WfMS.

5.1. Process definition tools

In [2.2] we’ve presented a general overview of the architecture of WfMS as proposed by the WfMC which consists on 5 interfaces. We first begin with the design of the process definition tool.

Figure 8—process definition interface
A process definition specifies which steps are required by the process and in what order they should be executed: routing definitions, procedures, workflow script.

A process definition consists of:

- Tasks (step, activity, process element): a task is atomic i.e. commit or rollback.
- Conditions (state, phase, requirements): a condition is used to determine the enabling of a task.
- Sub processes

A process definition tool is an application tool or software that is used to define such a process. To complete the definition we’ve given above, it’s also important to mention that a process is defined using constructs:

- Cases: it is the 'thing' which needs to be processed by following the process definition.
- Routing cases: it can be sequential or parallel, can provide choice or iteration
- Condition: it’s the requirement which must be satisfied.
- Trigger: timer, message, events …

Now we must face the issue of the language we should use for defining our process.

### 5.1.1. Workflow process definition language

As for every WfMS we need to define a process definition language. This language will be used by the process definition tool to describe the process in a workflow engine readable language. With other words, the resultant process definition will normally be held within the workflow product domain and may, or may not, be accessible via a programming interface for reading and writing information.

In the previous section we outline the need to use web services composition for our web services based WfMS and we’ve also presented some composition approaches. Now we must face the problem of choosing a process definition language which in reality hides the bigger problem of having a standard workflow language.

In fact, the challenge consists on using a workflow language that is widely used so that our defined process could be used with other WfMS solution.

After having a look on existing workflow language we observed that there is may be a predominance of some workflow language on others. Nevertheless (and that is the most surprising observation) none has been established or recognized has standard.

We totally agree with the author in [12] when he claims that “there are too many of workflow language and most of them die before becoming mature. A simple indicator of this development is the increasing length of acronyms: PDL, BPSS, EDOC, BPML, WSDL, WSCI, ebXML and BPEL4WS”. The abundance of overlapping standards for web services compositions is really overwhelming and we understand why some authors refers to these competing standards without clear added value as the Web Services Acronym Hell (WSAH). The figure bellow presents most languages around web services and workflow systems.
Despite the absence of a standard which would have made everything easier, we must choose one language between all these languages. In the scope of this thesis we’ve chosen to use BPEL4WS as our workflow language because it’s seems best to match our need.

5.1.2. **BPEL4WS as workflow process language**

As shown in the previous chapters BPEL4WS [21] can be used to describe executable business processes and abstract processes. Abstract processes are used to create behavioral specifications consisting of the mutually visible messages exchanged between transacting parties executing a business protocol.

From a structure point of view, a BPEL4WS file describes a workflow by stating whom the participants are, what services they must implement in order to belong to the workflow, and the control flow of the workflow process. The BPEL4WS process model is built on the top of the WSDL service model and assumes that all primitive actions are described as WSDL port types. With others words, a BPEL4WS file describes the choreography of messages which are described by their WSDL definitions. Furthermore, WSDL is also used to describe the external interface to the workflow. This allows BPEL4WS to be compositionally completed, which means that the composition of Web services is exposed as a single Web service eligible to participate in other compositions [9]. The following figure shows web services implemented as BPEL4WS.

![Figure 9—overview of web services technologies](image-url)
5.1.3. **BPEL4WS: really the best choice?**

Typically, a technology such as BPEL4WS does not fully address the challenge of composing workflow systems from Web services. BPEL4WS just gives the activities sequence i.e. the order or the orchestration scenario of participating web services. Thus regarding a workflow driven aspect we can say that the main focus of BPEL4S is based on expressing the mechanics of the workflow, but not its semantic aspect.

So we definitely miss here a web services semantics aspect of certain approaches such as DAML-S, OWL, or others algebraic formalisms described in [22].

Nevertheless we still believe that BPEL4WS presents so many advantages from a web service orchestration point of view that we feel like we must find a way to complete his “weaknesses” by joining BPEL4WS with another technology recognized as more semantic oriented, in order to use it as our workflow language.

Thus we propose here an approach which consists of combining BPEL4WS and another language so that semantic information can be derived from a defined process.

Our solution is to use PNs (and his extensions) as such a language.

5.1.3.1. **Mapping workflow management concepts onto Petri Net (PN)**

Nowadays there are many papers and researches that have been written to show how important are the advantages we could take on using Petri Net in the workflow world.

The approach we present here has been initially introduced in [8] and [14]. We find to it another interpretation and adjust it especially for a workflow and web service context.

Since the invention of classical PNs by Carl Adam Petri in the sixties, PNs have been used to model and analyze all kinds of processes modeling with applications ranging from protocols, hardware, and embedded systems to flexible manufacturing systems, user interaction, and business processes. In the

---

3 At the end of the WfMS design, we will discuss in detail the issues of several workflow languages
last two decades the classical PNs have been extended with color, time and hierarchy as shown by the author in [7] and [6]. Thus the modeling of complex processes has been facilitated and furthermore data and time are important factors now. There are several reasons for the widely success of using PNs for modeling workflow:

- **Formal semantics**: A workflow process specified in terms of a Petri net has a clear and precise definition, because the semantics of the classical Petri net and several enhancements (color, time, hierarchy) have been defined formally.

- **Graphical nature**: Petri nets are a graphical language. As a result, Petri nets are intuitive and easy to learn. The graphical nature also supports the communication with end-users.

- **Expressiveness**: Petri nets support all the primitives needed to model a workflow process. All the routing constructs present in today’s workflow management systems can be modeled. Moreover, the fact that states are represented explicitly, allows for the modeling of milestones and implicit choices.

- **Properties**: In the last three decades many people have investigated the basic properties of Petri nets. The firm mathematical foundation allows for the reasoning about these properties. As a result, there is a lot of common knowledge, in the form of books and articles, about this modeling technique.

- **Analysis**: Petri nets are marked by the availability of many analysis techniques. Clearly, this is a great asset in favor of the use of Petri nets for workflow modeling. These techniques can be used to prove properties (safety properties, invariance properties, deadlock, etc.) and to calculate performance measures (response times, waiting times, occupation rates, etc.). In this way it is possible to evaluate alternative workflows using standard Petri-net-based analysis tools.

- **Vendor independent**: Petri nets provide a tool-independent framework for modeling and analyzing processes. Petri nets are not based on a software package of a specific vendor and do not cease to exist if a new version is released or when one vendor takes over another vendor.

These are some reasons that motivate us to use Petri Net. As we can see it perfectly matches to what we need and furthermore advantages of using PNs go beyond the semantic aspect.

### 5.1.4. Understanding the definition of a business process in r-FLOW

r-FLOW uses the WfMC meta-model (see figure bellow) for the process definition, which identifies a basic set of object types appropriate to an initial level for the interchange of relatively simple process definitions. As it’s the case in most WfMS, process definition in r-FLOW will define the process structure, activities & navigation, role & participant, trigger conditions, application invocations, etc. And of course it will be used constructs provide by the underlying modeling language for defining them. The figure below gives an idea on a process definition model.
As we decided to describe a process using BPEL4WS and regarding the model shown in the figure above here are the primary constructs needed to define a process in BPEL4WS and therefore for r-FLOW:

- `<sequence>`: it specifies that its contents must be executed in the order presented;
- `<flow>`: it specifies that the contents are executed in parallel;
- `<while>`: it indicates that an activity must be repeated until the given criteria has been met;
- `<switch>`: allows the conditional execution of one of many activities;
- `<pick>`: blocks until a specified message arrives or a time-out occurs; when either one occurs its associated activity is executed;
- `<receive>`: designates that a message is to be received;
- `<reply>`: sends a message;
- `<invoke>`: construct invokes an operation on a specified partner, port Type pair;
- `<throw>`: is used to raise a fault;
- `<wait>`: pauses the execution for a specified time; and
- `<links>`: are used to specify complex ordering constraints. Specifically, each link has a `<source>` and `<target>` construct. The target only execute after the source has finished.

All these constructs can be used during a process definition. For a better understanding, we present bellow a sample BPEL4WS process definition with the structuring activity tags (in boldface).

The defined process is a service that provides the ability to obtain a stock quote from the NYSE in any currency. For example, the service could be called to obtain a quote for shares of IBM expressed in Swiss Francs. An analysis of this process reveals that the service is composed from 3 web services. These 3 (external) web services are the `stockQuoteProvider` service, the `currencyExchangeProvider` service, and the `simpleFloatMulProvider` service.

Example:\(^4\)

---

\(^4\) This is a classical example mostly taken when an illustration of BPEL4WS activity need to be made. It can be found on the IBM web site. [www.ibm.com](http://www.ibm.com)
<sequence>
    <receive name="request partner="requestor"
              portType="tns:stockLookupProcessPT" operation="requestLookup"
              variable="request" createInstance="yes">
    </receive>

    <flow>
        <assign>
            <copy>
                <from variable="request" part="symbol"/>
                <to variable="stockQuoteProviderRequest" part="symbol"/>
            </copy>
        </assign>

        <invoke name="getStockQuote" partner="stockQuoteProvider"
                 portType="nsl:StockQuotePortType" operation="getQuote"
                 inputVariable="stockQuoteProviderRequest"
                 outputVariable="stockQuoteProviderResponse">
        </invoke>

        <assign>
            <copy>
                <from expression=""usa""></from>
                <to variable="currencyExchangeProviderRequest" part="country1"/>
            </copy>

            <copy>
                <from variable="request" part="country"/>
                <to variable="currencyExchangeProviderRequest" part="country2"/>
            </copy>
        </assign>

        <invoke name="getCurrencyExchange" partner="currencyExchangeProvider"
                 portType="ns2:CurrencyExchangePortType" operation="getRate"
                 inputVariable="currencyExchangeProviderRequest"
                 outputVariable="currencyExchangeProviderResponse">
        </invoke>

        <assign>
            <copy>
                <from variable="stockQuoteProviderResponse" part="Result"/>
                <to variable="simpleFloatMultProviderRequest" part="f1"/>
            </copy>

            <copy>
                <from variable="currencyExchangeProviderResponse" part="Result"/>
                <to variable="simpleFloatMultProviderRequest" part="f2"/>
            </copy>
        </assign>

        <invoke name="doSimpleFloatMult" partner="simpleFloatMultProvider"
                 portType="ns3:SimpleFloatMult" operation="multiply"
                 inputVariable="simpleFloatMultProviderRequest"
                 outputVariable="simpleFloatMultProviderResponse">
        </invoke>

        <assign>
            <copy>
                <from variable="simpleFloatMultProviderResponse" part="multiplyReturn"/>
                <to variable="response" part="Result"/>
            </copy>
        </assign>
    </flow>
</sequence>
In the previous chapter we’ve decided to combine BPEL4WS and Petri Nets (PN) as modeling language in order to face the limitation of BPEL4WS as a stateless and semantic less workflow language. Our idea is to really take advantage of both technologies. Thus for defining a new business process in r-FLOW following steps should be done:

1. Define the process logic using PN construct component (state, transitions, message, token…). In this way we make sure that the finally process really does what he is intended to do (we could also later easily imagine a graphical user interface where users could easily model the process by using PN constructs).

2. After the logical and semantic process have been defined using Petri Nets, we’ll transform it into a machine readable workflow language i.e. BPEL4WS.

We’ve derived this approach from the work done by Van der Aalst in [14] where he presented a methodology to generate PNs via the repetitive replacement of elemental building block with other PNs.

In the following figure it can be seen how a process defined with PNs construct could be “translated” into a BPEL4WS file. Our idea can be summarized as designing (or defining) a process using PNs constructs and transforming the PNs constructs into BPEL4WS components.

---

5 Since the main focus of this thesis is not really about implementing the solution but designing it, we will not concentrate ourselves on showing how to implement such as graphical tool.
As the figure above shows, outgoing from a process defined with PNs construct, we map each PNs construct into a corresponding BPEL4WS.

To demonstrate the application of the replacement property we’ve mentioned before, the BPEL4WS workflow description found in the previously example will be used to illustrate the conversion of PN to BPEL4WS form. To show commutability of this mapping process, since we’ve already shown the resulting BPEL4WS file in [5.1.4], let’s present here how to gain the PNS that could have been used to obtain these files. The reverse mapping process i.e. translating PNs into a BPEL4WS file works exactly on the same way.

**Method to extract BPEL4WS file from PNs activities and vice versa**

→ At first, represent each BPEL4WS basic activity as a building block.

A. ![Diagram](image1)
   where x is the `<receive>` activity named request.

B. ![Diagram](image2)
   where x is the `<invoke>` activity named getStock Quote.

C. ![Diagram](image3)
   where x is the `<invoke>` activity named getCurrencyExchange.

D. ![Diagram](image4)
   where x is the `<invoke>` activity named doSimpleFloatMult.

E. ![Diagram](image5)
   where x is the `<reply>` activity named response.

→ By using the mapping presented in figure 9, repeatedly apply the replacement property, consuming the BPEL4WS process description. Starting with a sequence building block, we replace x with A and replace y with a flow construct, yielding F.
Starting with F, we replace x with B and replace y with net C, yielding G.

In a sequence building block, we replace x with G and y with D, yielding H.

In a sequence building block, we replace x with H and y with E, resulting in I.

Thus we obtain the final PN that produced the BPEL4WS file in the previous example.

It’s been proven that the both languages (PNs and BPEL4WS) are suitable for process definition or process modeling. Thus the rest consists on how the mapping between these two languages can be programmatically done. Because the main goal of this thesis consist on designing a WfMS entirely based on web services, so we limit ourselves on showing and using the feasibility of such a mapping for our WfMS design and we let this point as a future work. Nevertheless, we are aware that it’s impossible to do a perfect match i.e. not every building block in PNs could have a match in BPEL4WS. This is due to the fact that the both languages are still improving and are always adding new features. By example classics PNs concepts have been extended by the author in [7] and [6] with event, color or time concept which are not yet defined in BPEL4WS. Nevertheless the basic PNs building blocks have a match in BPEL4WS and we could still complete the resulting BPEL4WS file using other particular activities (<link>, <terminate>, <wait>…) or PN extension as also described by Van der Aalst in [15].

Notes: This example can be a bit confusing because we were going from the BPEL4WS to show the different corresponding PNs component. But in a normal r-FLOW scenario we have at the beginning a PN process that looks by example as the one as above and we should easily be able to derive a BPEL4WS file by using this approach.

5.1.5. Web service based process definition tool in r-FLOW and process definition tool architecture

In the previous section we showed how a process which uses a combination of BPEL4WS and PNs can be defined. As we stated in the beginning of this work, we want to design a web services based WfMS. Thus each of the five interfaces of the WFMC standard should be defined using web services.
We’ve also previously explained the two steps that must be done to define a process in r-FLOW. It fails the mention of web services aspect i.e. how web services come in play in this process.

Normally at the end of the process creation or process modeling, we know the expected behavior of our process i.e. it’s specified which tasks need to be executed and in which order. Furthermore modeling a workflow process definition in terms of a Petri Net is rather straightforward: tasks are modeled by transitions, conditions are modeled by places, and cases are modeled by tokens.

We make the following assumption: “each task in our process definition represent a web service and places represents conditions to be fulfilled by web services during their composition”.

To understand what we mean we present here the architecture of the R-FLOW process definition tool.

![R-FLOW process definition architecture](image)

In the previous sections we said many times that we will combine PNs and BPEL4WS that’s why we have a PN designer and a BPEL4WS generator (working as described in 6.1.4).

The r-FLOW process definition tool contains:

- A Petri Net designer: the PN designer will be used to model our process using PNs constructs.
- An axis engine: the axis engine will help us to map process components to web services, located in WS registry (it can be seen here as an internal UDDI). The idea is to search for each declared PNs transition (i.e. task) a corresponding web services in the registry.
- A BPEL4WS generator: it will be used to generate a BPEL4WS file that could be used by the workflow engine.
- A WS registry: it represents a registry of business stored web services that could be used in our process modeling. This part is very important because it’ll help to validate a process defined with PNs constructs in order to be later utilized for generating a BPEL4WS file.
- JRE: r-FLOW process definition tool will be implemented in Java.

The figure below illustrates the step that must be followed by our r-FLOW process definition tool.
5.2. Workflow enhancement services

The workflow enactment service provides the run-time environment in which process instantiation and activation occurs, utilizing one or more workflow management engines, responsible for interpreting and activating part, or all, of the process definition and interacting with the external resources necessary to process the various activities. As described in the WfMC reference model, it is a software service that may consist of one or more workflow engines in order to create manage and execute workflow instances.

Depending on the utilization of our WfMS we can distinguish two types of workflow enhancement services:

- **Centralized workflow systems**: all operations and transactions are vendor specific. By doing this, there is no need to find any standardization.
- **Distributed workflow systems**: they make use of specific protocols and interchange formats between Workflow engines to synchronize their operations and exchange process and activity control information. Workflow relevant data may also be transferred between workflow engines. We will talk later about the issues of this approach. Nevertheless most of the academic and research work are concentrated on this approach.

In addition to what we’ve mentioned above, most workflow engines help to:

- Interpret the process definition received by the process definition tool in a specified workflow language.
- Control the process instances: it is usually said that a workflow engine works like a final state machine where we could observe the different steps and transition for a given action. In the same logic it’s possible to control the creation, activation, suspension, termination of a specific process instance within a workflow engine.
- Navigate between process activities, which may involve sequential or parallel operations, deadline.
- Scheduling, interpretation of workflow relevant data, etc
- Sign-on and sign-off of specific participants
- Identification of work items for user attention and an interface to support user interactions
- maintain workflow control data and workflow relevant data, passing workflow relevant data to/from applications or users
- serve as an interface to invoke external applications and link any workflow relevant data

For r-FLOW we will use a centralized workflow enhancement approach with one workflow engine as process environment execution. We choose to use a centralized approach because, as already said, we are designing a solution that is specific to a certain business.

The r-FLOW engine is the hearth of our architecture; it consists of a minimal set of web services such as the process instance service, the task execution services, the trigger service, the condition service and the engine core service. In the following, we’ll describe in detail the role of these services. In the meanwhile the most important thing to keep in mind is that as for every WfMS, the r-FLOW core engine primary objective is to allocate selected work items to an appropriate resource.

5.2.1. Defining the r-FLOW workflow engine

As for the others components of r-FLOW, the engine is entirely composed of a set of communicating web services. Here we give an exact definition of these services:

- **Process instance service**: according to a general process definition language, the goal of this web service is to generate a specific case to be launched or started once case attributes received relevant values.

- **Task execution service**: a very important step in the execution of the workflow engine relies on the preparation of the resources and the scheduling of work items in the trash (work item list handler). The main objective of this web services is to execute an activity corresponding to a work item by receiving a trigger. The list of work is maintained by the workflow handle service. The orchestrator or scheduler services takes in account during the execution of these services the value of the specific case registered in the process instance services.

- **Trigger service**: the role of this web service is to provide functions or methods in order to send triggers message to corresponding participants. In such a case, we could consider different triggers implementations. By example a trigger for user actions or for resource actions can be implemented. Additionally we should define other form of triggering called external event (i.e. the arrival of a message) or reaching a particular time signal (i.e. the generation of a list of orders to be executed at 10 o’clock). Of course, this list can be extended to more specialized triggers without any side effect on the other services.

- **Condition services**: Similarly, we delegate the management of conditions associated tasks to a special service called condition service. Before a task can be performed as part of a specific case, it must fulfill certain conditions. A condition is therefore a requirement that must occur before an activity can be carried out.

- **Engine Core Service**: The function of the Engine Core Service hinges on allocating a selected work item to an appropriate resource. There are two ways in which this can be done.
  - The workflow engine matches work items and resources. Within pre-set conditions, the workflow engine can choose which resource performs each work item. The resource
itself thus is unable to choose. As soon as it has finished performing one activity, it is
given a new work item. We refer to this as push driven: the engine "pushes" work items
onto resources.
- The resources themselves match work items and resources. In this scenario, it is the
resources that take the initiative. Each has studied the work items that it is able to carry
out. It then chooses one. We call this pull driven: the resources "pull out" work items
and all "eat" from the same basket of work items.

- **Workflow Handle Service:** The objective of a workflow system is to complete work items
as quickly as possible. After all, a hold up affecting work items can result in the case as a
whole taking longer. In order to transform work items into activities, two decisions always need
to be made:
  1. In what order are the work items transformed into activities?
  2. By which resource are the activities carried out?
The order can be important when selecting a resource. Conversely the choice of a resource can
affect the order in which work items are transformed into activities. The workflow handle service
applies to select a particular order. In particular, the service implements various queuing
disciplines for production management that are used in factories such as First-In, First-Out,
Shortest Processing Time and Earliest Due Date.

- **Resource Information Service:** The resource information service manages the resources in
a workflow management system. We mainly distinguish three categories of resources: Role,
System, and Organization Unit. The roles designate human resources such as operators and
clients whilst the system resource presents an application or a production machine. According
to our general architecture, the resource information service can be supported or extended by
web services of enterprise applications. Thus the workflow engine is scalable.

- **Work list handler service:** The work list handler is a web service component which
manages the interaction between workflow participants and the workflow enactment services. It
is also responsible for progressing work requiring user attention and interacts with the workflow
enactment software via the work list.

- **Scheduler service:** This web service could be compared as an orchestrator that coordinates
distributed autonomous web services. Any service can be substituted by one another. Existent
web services published by enterprises applications can be easily integrated into existent
workflow engine or create a new workflow engine by assembling services. In all these
situations, the orchestrator is the central point that one should be modified.

In the figure below we present an extract the class diagram that composes the r-FLOW engine.
5.2.1.1. Example of saving a new process in r-FLOW

Here, we explain via an example how the r-FLOW engine works i.e. how web services work together to save a new process.

So as usual the first thing to do is to define the process using our process definition language. The user of r-FLOW defines a process in PNs and a translation from PNs components to BPEL4WS occurs. Thus the r-FLOW engine receives from the process definition tool a BPEL4WS file. Then the scheduler service will parse the BPEL4WS file to take all relevant information and send them to the corresponding service. The following figure illustrates a sequence diagram that shows which steps need to be done in order to save a new defined process in r-FLOW.
5.2.1.2. Starting a new instance in r-FLOW

Once a process has been saved inside our WfMS a new process case or instance is created with the process instance service and it must be launched via a workflow administration tools or via a workflow client application.

For each new process instance, in order to be started following steps must be done:

1. The user/client application send his request (start/stop process)
2. The workflow item associated to this process are investigated
   i. For each work item do following:
      1. Investigate which activity should be executed as next
      2. For each activity:
         a. Investigate all participants
            i. For each participant:
               1. Send message or triggers
         b. Investigate all relevant data
         c. Prepare and start the application service

The following picture illustrates again how an instance is started and in which order.
Steps at run time

User send request (start/stop)

For each work item
- Investigate activities to be started

For each activity
- Investigate participants
- Investigate relevant data
- Start application/prepare Start

For each participant
- Send message to Participant

Figure 17—steps at run time
5.3. Administration and monitoring tools

This is the place where administration functions are included such as:
- Set up and management of user names, passwords and roles
- Assignment or re-assignment of work items
- Processing exception conditions
- Control of process definitions or versions thereof
- Monitoring of work or process instance progress
- System audit functions

In r-FLOW the administration tool interface is integrated in the client application. The idea is to give a complete access to the client application on each process that has been started. So we just have to add additionally methods to the client application web services.
5.4. Workflow client applications interface

Client application interface help users to communicate with r-FLOW. It can be seen as a front end application or a client program. Since the client application is composed of web services, WSDL will be used as a protocol to exchange message between the r-FLOW engine and the interface. In r-FLOW, client application should help to interact with the workflow engine and the worklist handler web services plays and important role in this manner. Other common functions provided by this web service except worklist handling are:

- Process instance initiation and others such as suspend or resume
- Retrieving and manipulating process definition data
5.5. Invoked applications

As described in the WfMC glossary, an invoked application is a workflow application that is invoked by the workflow management system to automate an activity, fully or in part, or to support a workflow participant in processing a work item. Most of the time invoked applications are used to exchange application relevant data with the workflow engine. Generally, the invoked application may be local to the workflow engine, co-resident on the same platform or located on a separate, network accessible platform; the process definition contains sufficient application type and addressing information (specific to the needs of the workflow engine) to invoke the application.

In r-FLOW, the invoked application plays a major role. In fact it’s a part of every process execution. To be more concrete, the invoked application in r-FLOW is a data base ESL\(^6\) which will communicate with the r-FLOW engine. More information about ESL will be better understood later in this document [6].

---

\(^6\) See the part talking about the application case of r-FLOW
5.6. **Workflow inter compatibility**

The workflow community defines this interface as the ability for two or more workflow engines to communicate and work together to co-ordinate process execution. It can also be seen as the ability for two or more engines appearing to provide a single workflow enactment service with process execution share between engines.

In reality the inter-compatibility aspect of every WfMS hides the problem of the standardization. For specific purpose or business oriented case, we just have to make sure that all workflow engines participating in a process have the same way of communication.

For r-FLOW this is not a big problem since the usage of r-FLOW is business specific. Nevertheless we still have to face the issue of the standardization as it was already the case when we had to choose a process definition language.
5.7. Architecture of the WfMS

The main idea on the r-FLOW architecture is to maintain the original architecture proposed by the WfMC model [2.3] and then adapt it for enabling a web service environment support. We took the model initially proposed by Jinyoung Jang [5] because it exactly corresponds to what we need i.e. a WfMS architecture based on web service and at the same time it’s been made following the WfMC model.

As we can see in the figure below there is a kind of services container called web services resource. In this container we store all necessaries web services for designing new processes and for communicating with the invoked applications.

If we still have in mind the process definition tool we’ve proposed in [5.1.5], we can easily see that the web services resource corresponds to the web services registry.

![Diagram of r-FLOW architecture](image)

Now we can give a general overview on how our designed WfMS based on web services (r-FLOW) works:

1. The first thing to do is of course to define a process using PNs that will be converted into BPEL4WS for later usage by the workflow engine.
   - Each activity of our process is represented by a web service.
2. Then the BPEL4WS file corresponding to the new define process is transmitted to our workflow engine.
3. The workflow engine parses the file and creates a new case (among other things).
4. ESL is a database that serves as our invoked application from which users could start/stop process.
5. Process execution consists of interaction between the workflow engine and ESL or user inputs.
CHAPTER 6
EXAMPLE OF A USAGE OF R-FLOW

6. Application case : using r-FLOW in the HP configuration management process

In the following section we are going to present an application case of the r-FLOW machine. To put this work in its context we’d like to mention that we did this thesis over the last 6 or 7 months in the research laboratory of Hewlett Packard (HP) in Boeblingen (Germany). So that it becomes easier to understand the application of r-FLOW we’ve made, we firmly believe that we must present: the context and also how to use r-FLOW in order to improve the existing solution.

6.1. HP Openview RADIA

In this part we give very briefly an overview of RADIA in order to understand the application case of r-FLOW.

RADIA is a production management software packet from HP which supports connectivity to customer systems (deployment/collection) on multiple platforms/cross-firewall. It requires a client installation on each supported node in order to ensure that each computing device is maintained in the right configuration. Key items here are: inventory collection, patch management, packaging, and software distribution. The RADIA packet contains many software modules such as:

- The RADIA client: it must be installed on every machine managed by RADIA
- The RADIA configuration server (RCS): it listens connection from RADIA client in order to computes machine desired state.
- The RADIA policy server (RPOLS): This is responsible for managing policies information over the policy server portal.
- ...

We present in the following section an introduction of the RADIA workflow model. Typically the RADIA workflow process works as followed:

1. Each system which is managed by RADIA has a client module of RADIA (RADIA client)
2. Twice per day, at a moment of less traffic usage, the RADIA client opens automatically a connection to a remotely located RADIA module called RADIA Configuration Server (RCS).
3. After the connection has been established it sends to the RCS the list of the currently installed software and patches. We say that the client sends his actual state.
4. The RCS identifies each system through a unique ID (system id) and computes the list of software and patch that need to be updated, uninstalled or installed. We say that the RCS computes the **desired state** of the RADIA client.

5. After the desired state of the machine is computed, the RCS respond to the client by sending him his desired state. In this way the RADIA client downloads and installs the software or patches and replaces his actual software and patch list. We say that the client has his new desired state.

6.2. Policy resolution and ESL

After introducing RADIA, another important point is the policy resolution. As explained before with RADIA it's possible to install/remove automatically software/patches on remote located machines. If we think again at the RADIA workflow process (see figure 26), intuitively we see that the question of assigning new software or patches to certain machines comes up.

In fact when a RADIA client send to the RCS his actual state (software a, software x, patch a, patch x), meaning the actual list of existing software and patches, he receives as return from RCS his desired state (software a’, software x’, patch a’, patch x’). The process of deciding which software or patches will be installed on which machines and not on others is called policy resolution or policy assignment. This is the heart of the RADIA workflow because it defines which action should take place and where it should be taken.

ESL is a database, which contains among others, information on the enterprise architecture managed by RADIA. ESL contains also software and patch deployment information.

6.3. Our task

Above the company interest, our tasks should present high academics and research interest in order to be reused, developed or improved by any other independent organization (university, researchers …). Our main task consisted on studying a new approach for the existing policy resolution model which will use web services and workflow management systems. This is how the idea to design rFLOW (r for RADIA) was born.

Precisely our jobs consisted in:

---

7 See on figure 6 the pm_deployment and sw_deployment tables
• Design a dynamic web services based WfMS (r-FLOW)
• Propose a worldwide solution architecture that will and support RADIA
• Design a new policy assignment model for RADIA and proof his advantages with the existing policy model.
• Design how to leverage the policy model of the RADIA workflow
• Implement a prototype of the solution

In this thesis we’ve decided to put the focus on the design of r-FLOW because, if we had put all the other parts of our work in this document this would have gone beyond the scope of this work.\(^8\)

\(^8\) A report concerning all the implementations work we did is available on request.
6.4. Using r-FLOW for triggering RADIA processes

The final solution we proposed consisted on replacing the old way of defining the policy with r-FLOW. Thus users will define policy models in form of a process. In addition to that, once RADIA policy models would have been stored in ESL, we proposed an approach based on LDAP servers to propagate policies to other RCS via LDAP servers.

For the purpose of this thesis we will focus on r-FLOW. Our final solution looks as shown in the following figure.

As it can be seen our idea consists on using r-FLOW as a process policy definer. In order words, r-FLOW replaces all external and manual applications that fulfilled the ESL database with policies information.

Thus the new RADIA process will consist on following steps:
1. Define the policy assignment as a new process within r-FLOW
2. R-FLOW is now responsible for fulfilling the corresponding tables in ESL and at the same time will extract specifics information from ESL to execute its process.
3. Once the policy changes has been made, a trigger will be started from ESL to send the policy information to a master LDAP server which replicates it to other LDAP server in order to complete the RADIA resolution model.

After a closer look on the global architecture of r-FLOW we’ve proposed it can be seen that we used a web services container where a set of web services are stored. In this container it can be found all necessaries web services for process definition and its execution.
6.4.1. Translating the policy assignment into a process definition language

Let’s explain with a simple concrete example how to define a policy model in r-FLOW. As example we can consider following policy assignment rule: “the machine with the identification X123 from sub business ERICSSON Canada must have installed the patch number PTCH7234”. This policy assignment rule means that we want to install a patch (with a specified number) on a specific machine that belongs to the business unit group ERICSSON, located in Canada. In order to that, we need to translate this assertion into a workflow process language. In fact, the policy assignment in RADIA is more complicated than what we’ve defined in the previous section [6.2]. Nevertheless this is an important point because the process definition is derived from the policy rule. Thus for our example in order to extract the process that will be handled by r-FLOW, outgoing from the given policy assignment rule, we must follow these steps:

1. Check out the machine configuration where the patch must be installed
2. Verify that the patch to install is compatible with the target machine
   a. If YES then go to step 3
   b. If NO then go to step 6
3. Resolve the delta⁹ between the current and desired state
4. Verify that the patch is conformed to system role and customer requirements
   a. If YES then go to 5
   b. If NO then go to 6
5. Store the new policy assignment into ESL
6. Report the actual state of the machine into ESL
7. End of the process

As we previously said in this document, to define a process in r-FLOW we must define it using PNs and then translate them into BPEL4WS so that the process can be read and interpret by the workflow engine. The figure bellow illustrates the process in PNs for our example.

---

⁹ Delta is the difference between the actual machine state and his desired configuration
After defining the process in PNs we are sure that this process respects our logic i.e. the semantic of the process. So the next step is to provide the workflow engine with a readable language (BPEL4WS). The BPEL4WS can be easily generated using the constructs we proposed in [5.1.4] but, for readability purpose, we only present here the corresponding BPEL4WS process map view of the process instead of pasting the whole xml file.
The figure above shows us a list of available web services that is used for the process execution. In reality the web service container has more web services available depending on the process we want to design.

We took this easy example to show the principles of r-FLOW but in fact a process definition in r-FLOW can be very complex and many web services or external components could be involved. This is due to the fact that the policy resolution in RADIA is more complicated. In fact, for policy assignment we must generally:

- Ensure the assignment of OS, Application and Patch baseline standards
- Ensure the assignment of Customer specific standards
- Ensure the assignment of system Role specific standards
- Implement software change management decisions within agreed timelines
- Provide mechanism to input and store assignment policies
- Provide mechanism to logically group systems and apply policy by group
- Facilitate software content and distribution mechanism testing prior to actual deployment
- Provide mechanism for reporting current system state
- Automatically resolve delta between current and desired state

And for each of this step, it should exist web services in the web service container.
CHAPTER 7

RELATED WORK ON WFMS BASED ON WEB SERVICES

7. Related work on WfMS bases on web services

Recently, a number of studies have employed the web service technology to resolve the heterogeneity and interoperability problems in implementing inter-organizational workflow management systems:

- DySCo: [1] is an extension to traditional WfMS that enables web services to be composed into business solutions.
- eFlow: [2, 3] is a service platform that facilitates service process composition and enactment by adapting the templates of predefined composite processes.
- MicroFlow: [4] uses object-oriented framework approach. MicroWorkflow does not employ the web service technology despite the fact that, it can easily handle the heterogeneity and distribution issues in a more efficient and standard way.
- WISE (Workflow-based Internet Services): [15] has both a runtime component and a development environment associated to it. WISE is organized around three service layers: database services, process services and interface services. A close view of the architecture shows that the services are tightly coupled and the consequence between these services depends on the communication between them. It is not easy to replace a service with another and to dynamically change the interaction between web services.
- CrossFlow: [16] proposes a contract based inter-organizational workflow system to control the execution of business services offered by different organizations. The extended workflow management system is tailored for supporting standard cooperation for a supply chain in a static virtual enterprise environment. Although the architecture fits well the network of distributed enterprises, it does not respect standards such as WfMC model or communication protocols (XML-SOAP).

All these innovative approaches have contributed to the application of web services in the workflow world but not in the same matter as we’ve done in this work.
CHAPTER 8
EXISTING ISSUES FOR FLEXIBLE WFMS THROUGH WEB SERVICES

8. Challenges for designing web services based WfMS

Nowadays most workflow initiatives try to embrace the web service model. This is due to all the big promises and expectations behind the web services model. Web services create dynamics environment requiring adaptive responses. But given the current state of technology, web service based workflows typically are deployed behind corporate firewalls and are widely used for intra-organizational workflow (like r-FLOW).
The reason for this is that web service specifications are weak in regards to issues such as described in the following parts.

8.1. Security
This issue comes through the fact that the research world hasn’t proved the perfect security of web services. Although the fact that lot of works have been done to improve the security of web services (by example the ws-security model developed by IBM) they aren’t perfectly safe. Consequently lots of companies see this aspect as a high risk to transit their business from intranet to a more open area (internet).

8.2. Transaction
This issue is due to the fact that errors or failures can occur during process execution. By example with r-FLOW, an unexpected behavior from our invoked application (ESL) could damage or make a process execution not to terminate. Nevertheless the approach consisting on adapting Petri Net to the workflow world and proposed by Van der Aalst in [8] shows that we can considerably face the transaction issue by using Petri Net. Thus a task can be seen as a Logical Unit of Work and an activity as a transaction. Just like a transaction, an activity should satisfy the well-known ACID properties\(^\text{10}\):

- **Atomicity**: an activity is executed successfully (commit) or is rolled back completely, i.e., a task cannot be completed partially.
- **Consistency**: the execution of an activity leads to a consistent state.
- **Isolation**: the effect of the execution of an activity in parallel with other activities is equal to the effect of the execution of one activity in isolation.
- **Durability**: the result of a committed activity cannot get lost

\(^{10}\) Here we defined the ACID properties as proposed by Van der Aalst in [8] in relation to WfMS
8.3. Standardization

This is may be the biggest problem of WfMS based on web services. Many time during the design of r-FLOW we had to face with this issues (see [5.1.1]).

Some authors compare today’s situation of designing and implementing WfMS, to the situation as it was regarding the need to have database standard language in the early seventies. It existed in this period a big disorder and a lack of consensus that resulted in an incomprehensive set of DBMSs. Despite the efforts of the workflow community a real conceptual standard is still missing. As example of these efforts, we could mention the fact that the reference model proposed by the community has been adopted. Most implementations nowadays utilize this model. But this is by far not enough!

There is urgently a need for a standard workflow language. The problem is that each vendor or organizations want their solutions or approaches to be adopted. There is not really a need to find a consensus or a solution that will integrate others approaches. Thus certain people want BPEL4WS as workflow language and others want WSCI, BPML, ebXML… as standards workflow language.

As a result, the world is confronted with too many standards which are mainly driven by concrete products and/or commercial interests. May be the only way to stop this is to ignore standardization proposals that are not using well-established process modeling techniques. This will force vendors to address the real problems rather than creating new ones.

The author in [5] proposed the EWA architecture where he uses a technique to overcome this language dependence. In fact he proposes a process definition publisher (PDP) which represents process definition in various workflow languages to improve the exchange and communication between most existing workflow. The figure below shows the PDP.

![Figure 28-EWA process definition publisher](image-url)

When an intranet-based workflow system executes, it does so with a collection of services that are owned and managed by the same organization. In this environment, service interruptions are infrequent and typically scheduled due to consolidated system management. In contrast, Internet-based workflows must be designed for resilient operation as service partners periodically become unavailable due to decentralized system management and the lack of network service guarantees. The evolution
from intra- to inter-net based workflows will increase the design and run-time complexity, since the coordination mechanism must become more faults tolerant.
CHAPTER 9

CONCLUSION

9. Conclusion

In this thesis our goal was to conceptualize a dynamic WfMS entirely based on web services (r-FLOW). We followed the reference model proposed by the WfMC community for designing r-FLOW.

As process definition language, we proposed an approach that consists on combining BPEL4WS and Petri Nets through a repetitive replacement of components in PNs with the corresponding components in BPEL4WS. The advantage of using the combination of these two techniques is that BPEL4WS is one of the most known process definition language and the utilization of Petri Net adds a semantic aspect to the defined process. The workflow enhancement service of r-FLOW is composed of a centralized workflow engine which contains a set of communicating web services. Other components of our workflow system are a set of communicating web service.

At the end of this work we could say that r-FLOW presents many advantages. Among other:

- It doesn’t have inter-compatibility issues because it’s an intra net WfMS adapted to a specific business.
- R-FLOW is a dynamic system because it adapts itself to changes which occur during the process. This is implicitly done by the fact that we our solution is based on communicating web services.

During this work we focused ourselves on the design of r-FLOW, but not its implementation. So the next step of this work will be to implement the whole system and the associated components. Due to the time given and the list of items we had to resolve, a complete implementation of the system would have not been possible. In addition to the implementation of r-FLOW a future work will consist on extending r-FLOW with many others functionalities and properties. As example it must be included in r-FLOW, the support for transactional workflow, versioning of workflow processes, security, resource management, communication and inter-compatibility. Another important point is, to add the treatment of exception that may occur during the process definition or execution.

We could definitely observe that, despite of the good aspect of r-FLOW a lot question and issues are not (yet) resolved. By example we didn’t find a definitive answer or a way to overcome the abundance of process definition languages, which has as consequence the problem that nowadays no standard process definition language exists. Our feeling is that this problem would not find a solution since each process definition provider is sure to have the best solution.
Thus if all participants (vendors, research community, institutes...) do not work together with the perspective to find a consensus then, as it was the case for data base, the most powerful vendors will impose their solution as standard.
10. References


