An Agent-Based Platform for the Management of Dynamic Virtual Enterprises

Vorgelegt von
Dipl-Ing.
Evangelos K. Ouzounis
aus Drama, Griechenland

Von der Fakultät IV – Elektrotechnik und Informatik
der Technische Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
-Dr.-Ing.-

genehmigte Dissertation

Vorsitzender: Prof.Dr. K. Obermayer
Berichter: Prof.Dr. R. Popescu-Zeletin
Berichter: Prof.Dr. B. Mahr

Tag der wissenschaftlichen Aussprache: 26 April 2001

Berlin, 2001
D 83

Abstract

The penetration of Internet and the World Wide Web (Web) in accordance with new technological advances urged companies to seize the opportunities offered by Electronic Commerce and Electronic Business. Especial promising are the opportunities for co-operation among companies in terms of Virtual Communities and Virtual Enterprises based on open networks and innovative Information and Communication Technologies (ICT).

Virtual Enterprises provide, from one side, the possibility to share key business processes in a profitable way and, from the other side, access to capabilities and resources offered by other partners. This can lead to shorten development and manufacturing cycles, reduced time to market and operational costs, global operation and reach, and rapid adaptation to new market needs. Virtual Enterprises are goal- and purpose-oriented associations of companies and/or department of companies that have limited duration, flexible configuration, dynamic distribution of roles, and share key business processes using advanced ICTs.

In the literature, two broad, well-defined categories have been identified so far, namely the static Virtual Enterprises and the Dynamic Virtual Enterprises. In Static Virtual Enterprises the relationships among the partners are static, pre-configured and can not easily change. On the contrary, Dynamic Virtual Enterprises use the IC Technologies to precipitate the best configuration of processes and resources for a certain process, as well as, to incorporate the most competent partner for that process. Dynamic Virtual Enterprises improve significantly the Static VEs and take full advantage of the open, global opportunities offered by the Internet and the global economy. Dynamic Virtual Enterprises feature very short lifetimes, dynamic business relationships among partners and flexible and autonomous behaviour.

This thesis analyses, designs, develops, testes, and validates a platform for the management of dynamic virtual enterprises that supports the whole life cycle model, namely the business process specification and registration and business process execution and management.

The main contribution of this thesis is an agent-based virtual marketplace, a virtual marketplace ontology, an agent-based inter-domain workflow management system, a business process definition language for dynamic VEs, an intra- and inter-domain ontology for business process management, a mechanism for dynamic selection of partners, and a negotiation ontology. The proposed platform is based on emerging Internet standards, like XML, and agent standards, like FIPA and OMG-MASIF.

The agent-based platform for the management of dynamic Virtual Enterprises has been fully implemented and tested. The validation and assessment of the platform has been done by the development of four independent business and application scenarios in the context of several research projects. Based on the validation and assessment, the proposed platform reveals openness due to the flexible ontologies used for the management of shared business processes, dynamicity and flexibility due to the dynamic selection of partners and the automated negotiations, asynchronous and loosely coupled coordination of processes due to the deployment of standard autonomous intelligent agents and distribution and scalability due to the distributed execution and management of shared business processes from different intelligent agents across different administrative domains.

This thesis aims to provide a systematic, coherent and state of art solution for the management of dynamic Virtual Enterprises based on standard, intelligent agent concepts and technologies. The acceptance and penetration of solutions like this will depend on the success and adoption of the

intelligent agent paradigm As the Internet and the new digital economy urge companies to collaborate and share critical business processes dynamically, solutions like this will become more important, applicable and effective for day to day business operations.

Zusammenfassung

Infolge der rapide zunehmenden Verbreitung des Internets, des WWW und der zugehörigen Technologien bieten sich zukünftig für Unternehmen vielfältige Möglichkeiten im Bereich e-Commerce und e-Business. Besonders vielversprechend sind die Chancen der Kooperation mit anderen Unternehmen in Form von Virtuallen Gemeinschaften und Virtuellen Unternehmen, die auf der Grundlage der weltweiten Netze und der innovativen Informations- und Kommunikationstechniken (IuK) eingegangen werden können.

Solche Virtuellen Unternehmen bieten einerseits die Möglichkeit, eigene Kernkompetenzen in kollaborativen Geschäftsprozessen gewinnbringend zu nutzen und andererseits von den Kapazitäten und Ressourcen der Partnerunternehmen zu profitieren. Dies kann u.a. zur Verkürzung von Entwicklungs- und Produktionsprozessen führen, Zeit und Kosten der Markteinführung von Produkten und Diensten vermindern und zu neuen Marktpotentialen führen. Virtuelle Unternehmen sind zielgerichtete, zweckbestimmte Vereinigungen von Individuen, Unternehemen und/oder Teilen von Unternehmen, die typischerweise durch begrenzte Lebensdauer, flexible Konfiguration, dynamische Rollenverteilung und IuK-orientierte, vernetzte Geschäftsprozesse gekennzeichnet sind.

In der Literatur wird gewöhnlich zwischen zwei Arten Virtueller Unternehmen unterschieden, nämlich zwischen den statischen und den dynamischen Virtuellen Unternehemen. Die statischen virtuellen Unternehmensformen ähneln weitgehend den traditionellen Formen und unterscheiden sich von diesen hauptsächlich durch die intensive Nutzung der IuK-Techniken in dem Zusammenwirken der Geschäftsprozesse, die jedoch fest den Partnern zugewiesen sind und nicht kurzzeitig mit Hilfe der IuK-Techniken im Unternehmensverbund verteilt und zugewiesen werden. Die dynamischen Unternehmen nutzen dagegen die IuK-Techniken voll dafür aus, für bestimmte Aufgaben die günstigste Konfiguration von Prozessen und Ressourcen dynamisch herbeizuführen und den jeweils kompetentesten Partner für eine Aufgabe einzugliedern. Dabei nutzen sie soweit als möglich die Vorteile des Internets und der globalen Geschäftsumgebung.

Die vorliegende Dissertation analysiert, entwickelt, prüft und bewertet ein Plattform für die Verwaltung von dynamischen Virtuellen Unternehmen, welche Spezifikation, Registrierung, Steuerung und Verwaltung von Geschäftsprozessen unterstützt und damit den gesamten Lebenszyklus von Virtuelle Unternehmen abdeckt.

Der Hauptbeitrag dieser Dissertation ist ein agentenbasierter virtueller Marktplatz, eine Ontologie zur Bestimmung und Ordnung von Begriffen des virtuellen Marktplatzes, ein agentenbasiertes, domainübergreifendes System für die Verwaltung der Arbeitsvorgänge, eine Sprache zur Definition von Geschäftsprozessen für dynamische virtuelle Unternehmen, eine Ontologie für die Verwaltung von domänen-übergreifenden Geschäftsprozessen, ein Mechanismus für die dynamische Auswahl von Partnern und eine Ontologie zur Unterstützung von Verhandlungen. Die vorliegende Plattform gründet sich auf Internetstandards, wie XML, und Agentenstandards, wie FIPA und OMG-MASIF.

Die agentenbasierte Plattform für die Verwaltung von dynamischen virtuellen Unternehmen wurde vollständig implementiert und getestet. Die Einschätzung und Bewertung der Plattform wurde durch vier eigenständigen Geschäfts- und Anwendungsszenarien im Zusammenhang mit verschiedenen Forschungsprojekten vorgenommen. Die Ergebnisse dieser Arbeiten zeigen, daß die vorliegene Plattform folgende Vorteile in sich vereinigt: Offenheit durch flexible Ontologien für die Verwaltung von geteilten Geschäftsprozessen, Dynamik und Flexibilität durch automatisierte Suchs-, Vergleichs- und Verhandlungsverfahren bei der Auswahl von Partnern,

asynchrone und lose gekoppelte Koordinierung von Prozessen durch den Gebrauch von standardisierten, autonomen intelligenten Agenten, und die Verteilungs- und Skalierungsfähigkeit durch die verteilte Ausführung und Verwaltung von geteilten Geschäftsprozessen durch verschiedene intelligente Agenten über verschiedene administrative Domänen hinweg.

Damit bietet die vorliegende Dissertation eine systematische Lösung für die Verwaltung von dynamischen virtuellen Unternehmen, basierend auf den innovativen Konzepten und Techniken der standardisierten intelligenten Agenten. Die Akzeptanz und Verbereitung solcher Lösungen wird vom Erfolg und der Annahme des Paradigmas der intelligenten Agenten abhängen. Da das Internet und die neue digitale Wirtschaft Unternehmen auffordert, zusammenzuarbeiten und kritische Geschäftsprozesse dynamisch zu teilen, werden Lösungen wie diese immer wichtiger, anwendbarer und effektiver für alltägliche Geschäftsabläufe.

Preface

Some years ago when I first came in contact with Computer Science I realised that it is a new exciting world full of interesting subjects. I also realised that as a new science there are a lot of possibilities for new developments, ideas, and research. At this point of time I decided that I wanted personally to contribute to this exciting world by providing my own perception in terms of research and development.

The area that I decided to continue my specialisation was the area of distributed systems, inter-domain services and e-commerce. Four years before it was a brand new world full of excitement and optimism. It was just the beginning, a unique chance to research and develop new concepts and ideas, the right movement for me! Virtual enterprises were a very promising area of work due to the lack of definitions, structure and previous research in the field. I worked hard to research, analyse, and understand the field. In this difficult and complex process, not all the moments were happy and successful. However, I kept going and going, trying to be innovative and scientifically consistent. Suddenly, I realised that I could also put my own small "stone" in the milestones of E-commerce and Virtual Enterprises.

A lot of people helped me during this complex and, sometimes, painful process and I am really grateful to them. First of all, I would like to thank my supervisor Prof. Dr. Radu Popescu Zeletin who gave me freedom in doing my own research. Prof. Dr. Zeletin taught me how to focus on the small, core issues without missing the whole picture and how to put legitimate and meaningful research questions. I would also like to thank Prof. Dr. Mahr for his prompt suggestions and remarks during the last and most difficult phase of this dissertation. Additionally, I would like to thank Dr. Volker Tschammer, head of ECCO in GMD-FOKUS and my boss, for his continue support in both technical and psychological matters. Dr. Tschammer taught me to put structure in my thoughts and to be patience and unsatisfied until I found something really new and innovative. Further, I would like to thank all my colleagues in ECCO at GMD-FOKUS for their good technical feedback in this subject. Finally, but definitely not lastly, I would like to thank my girlfriend Elli Athanasiadou and my family for their support, love and patience that has kept me sane. Without them, I could not accomplish this dissertation.

Berlin, April 2001

Evangelos K. Ouzounis

Table of Contents

Chapter 1: Background and Motivation

1.1 Background and Motivation

In a global marketplace, companies are continuously seeking for new ways to address competitive pressure. Recognising the need to shorten development and manufacturing cycles, reduce time to market and operational costs, increase customer satisfaction, operate on global scale and reach, and rapid adapt to new market changes has historically led companies to automation, collaboration and distribution (Applegate 96, Billington 94, Malone 91, Ouzounis 98a). As a result, the information systems in many of today's mid to large size companies reflect tremendous diversity.

Rapid advances in telecommunication, open networks like the Internet and the Web, interoperable distributed object oriented technologies and platforms like CORBA (OMG 98, Orfali 96) and Java (Java 98), component-based development and middleware like Enterprise Java Beans (EJB, 98), flexible meta-languages like eXtensible Markup Language (XML 98) have been opening and enabling new opportunities for electronically conducted business. But doing business electronically means to shift critical business processes to open networks, like the Internet, and enrich customer and supplier relationships (Malone 91, Ouzounis 98b,c).

The penetration of Internet and the web, in accordance with new technological advances, urged companies to seize the opportunities offered by electronic commerce and to establish a strategic position in the new global networked world. In order to do that, companies should co-operate in different product development phases and share critical business processes, resources, core competencies, skills and know how with each other (Christofer 93, Applegate96, Ouzounis 99a, Camarinha-Matos 99). This new business model leaded to the concept of Virtual Enterprises (VE) that is the foundation of the networked economy (Ouzounis 99a,b, Camarinha-Matos 99, Fielding98).

The original goals for virtual enterprise business systems were to enable deployment of distributed business processes among different partners, to increase the efficiency of existing

provided services, to decrease the cost for these services, and to adapt to new market changes (Banahan 99, Stricker 00, Davis 93). As companies introduced electronic business systems, they started to see new possibilities enabled by them. By more closely coordinating the work of suppliers and manufacturers, businesses see dramatic productivity and efficiency increases in manufacturing processes. As communication barriers and costs drop, businesses are able to engage them selves in many more kinds of relationships. These new relationships open additional possibilities for distribution and participation in virtual trading communities or dynamic virtual organisations and for extending classic value chains to value networks (Doz 98, Adams 97, Malone 91).

Virtual enterprises are not a new concept in management studies (Malone 91, Adams 97, Ouzounis 99b, 00a, Camarinha-Matos 99). Some of the big manufacturing companies, and especially car manufactures, have already business relationships with their suppliers and customers. These "virtual" business relationships enable the sharing of business processes and resources among them. However, the level of integration and the Information and Communication Technologies (ICT) used for enabling Virtual Enterprise concepts is varying. Most of the activities are still performed manually, ad-hoc, and in a complex way, while the cost to implement and integrate these solutions and the time required to deploy them is high (Lin 96a, Lin 96b, Reichert 98).

The paradigm of virtual enterprise represents a prominent area of research and technological development for today's progressive industries. The research area is however a growing and multidisciplinary one that still lacks a precise definition of the concepts and an agreement on the used terminology (Camarinha-Matos 98, Applegate 98, Bolcer 99, Carr 96). So far, there is no unified definition for this paradigm and a number of terms are even competing in the literature while referring to different aspects and scopes of Virtual Enterprise (Ouzounis 99a, Filos and Ouzounis 00a, 00b, Alonso 99). For instance, the US-based R&D project NIIP project defines that "a VE is temporary consortium or alliance of companies formed to share costs and skills and exploit fast-changing market opportunities"(NIIP 97, Zarli 99, Wognum 99 a and b). Byrne says that "a VE is a temporary network of independent companies-suppliers, customers, even rivals-linked by information technology to share costs, skills, and access to one another's hierarchy, no vertical integration" (Byrne 97). To Walton and Whicker "the VE consists of a series of co-operating 'nodes' of core competencies which form into a supply chain in order to address a specific opportunity in the market place (Walton 98).

The wide variety of different networked organisations and the emergence of new production and provisioning paradigms have led to the generation of a number of related terms such as the extended enterprise, virtual organisation, networked organisation, supply chain management, or cluster of enterprises (Malone 98, NIIP 97, Ouzounis 98c). Some authors use some of these terms indistinctly to virtual enterprises although there are differences between their detailed meaning (Zarlin 99, Wognum 99).

In the context of this thesis the following definition is adopted (Ouzounis 98e, 99b): "a VE is a network of different administrative business domains that co-operate by sharing business processes and resources to provide a value-added service to the customer. Each partner of the virtual enterprise will contribute primarily what it regards as its core competencies, i.e. business processes and resources. There is a time limit on the existence of the virtual enterprise caused by fulfilment of its business purpose. From the viewpoint of an external observer, i.e. a customer, the virtual enterprise appears as a unitary enterprise."

Although there is no strict academic definition regarding VE, different VE models feature common business and technical characteristics and attributes. The most important features of VE are (Ouzounis 98c, Block 95, CIMOSA 98, Zarlin 99, Georgakopoulos 98, Geppert 98 a and b, Goldman 95, Goldman 95, Gibon 99):

- more than one independent administrative domains are involved in the provision of the service to the customer
- the service provision is performed by sharing business processes and resources, i.e. by establishing business relationships among the different VE partners,
- the sharing of processes and resources lasts for a limited period of time even only for only one service provision,
- the business process interfaces among the business domains, i.e. the way that one domain deploys the processes and resources of the other, might be static, pre-defined, and fixed or dynamic, based on a set of globally specified templates,
- the number of VE partners might be either, static, or dynamic according to the needs and requirements of the partners involved,
- the partners are physically distributed and are connected with electronic means and systems,
- the provision of the services to the customer is done in a transparent way by one representative partner.

Based on the above common features that VEs have, two well-defined categories can been identified (Ouzounis 99b, Malone 97, Zarlin 99, Geppert 98 a and b, Gibon 99), namely the static Virtual Enterprises (SVEs) and the Dynamic Virtual Enterprises (DVEs).

In Static Virtual Enterprises a set of business partners is linked together in a static and fixed way, i.e. the shared business processes are tightly integrated. The business relationships among the partners, i.e. the process interfaces are pre-defined, tightly coupled, fixed, and well integrated and customised among the partners (NIIP 96, Malone 96, Afsarmanesh 99). The network is fixed and pre-determined and thus, the structure of the VE is static and pre-determined.

In Dynamic Virtual Enterprises a set of business partners is linked dynamically, on-demand, and according to the requirements of the customers, by deploying a virtual marketplace (Wognum 99, Ouzounis99b, Mitrovic 99). The business domains do not have fixed business relationships and thus, the VE is not static and might change continuously based on market-driven criteria. The marketplace provides services for the registration of business process offerings based on some generic, well-known, globally specified process templates. Business domains that want to form VE relationships can register offers on the marketplace in relation to the process templates. Whenever a business domain wants to use a particular process, searches the marketplace, and locates all the potential partners that can provide the process. As soon as the list of VE candidate partners has been found, the partner selection process starts. The partner selection process between the domains is usually performed through negotiation. The negotiation process might be either, manual, or automatic, while the result of it is usually a short-term contract that regulates the business relationships that have just established (Ouzounis 00b, Filos and Ouzounis 00, Geppert 98).

By deploying virtual marketplaces, there are no explicit static business relationships among the partners and thus no integration among the processes of the partners is required (Merz 96, McCaffer 99, McCutcheon 94, Nwana 96, Nwana 99). Marketplaces are usually organised

around certain globally specified service or product templates that can be offered by the different vendors. The marketplace is a match making mechanism that brings potential process providers together with potential users of these processes. The primary focus on marketplaces is on efficiency of transactions and maximisation of value per cost of each vendor's offer (Camarinha-Matos 98, Mohan 98, Frederix 98). Organisations may participate in the marketplace only briefly or they may be long term members. Relationships between process users and process providers tend to be short term. Thus, investment returns are gained over single transactions, as well as, over the time span of the marketplace participation. The number of partners can easily change and thus the structure of the VE can also change from one service provision to another according to the specifics of the customers and the current needs of the partners. This is a significant evolution mechanism that takes advantage the demand and supply, i.e. the process offerings by the individual domains (Alonso 98, Schuldt 99).

Due to the open mechanisms of the Internet economy, dynamic, flexible, autonomous VEs that take advantage of the market conditions are preferred. Although from business point of view DVEs are the most promising business model, from technical point of view, the required technical solutions and systems are more complex, sophisticated and distributed (Kalakota 96 and 98, Ducroux, 1998). However, the advent of Internet and open communication protocols, like TCP/IP and HTTP (Gaedke 98, Berners-Lee 94), distributed object oriented middleware systems, like Corba-IIOP (OMG 92-99) and Java-RMI (Java 98), and extensible meta-languages, like eXtenible Markup Language (XML), provide the basic building blocks for the development of management platforms that will realise the concept of DVEs (Ouzounis 98c, Kligenmann 99 a and b, Reichert 97).

The first attempts to realise cross-organisational business systems have been done in the area of Electronic Document Interchange (EDI), where a set of business domains integrate their business processes only for electronic commerce purposes. In that case, different business domains have static business relationships, the communication mechanisms used were message passing, the business processes were internal modules or legacy systems maintained by the different partners, the network protocols used were secure virtual private networks, and the format of the EDI messages were proprietary following certain structure and regulations (Gibon 99, Ouzounis 99b, Lomet 93, Lee 98).

Though EDI was a significant progress towards the direction of cross-organisational business automation, the resulted solutions were very expensive and rather closed to be adopted by small firms (Snapp 90, Srinivasan 93). Certain problems regarding the standard format of EDI messages, the insecure open transport networks, and the rather restricted context of EDI, i.e. only focus on electronic commerce, made EDI not an attractive solution for VEs (Billington 94, Christofer 93, Bolcer 99, Doz 98).

Whereas EDI supported electronic business by automating existing processes and enabling electronic document exchange between separate organisations, a number of other technologies approach inter-domain business relationships by trying to create a single virtual enterprise (Stricker 00, Georgakopoulos 98, Fielding 98). These systems use middleware, a layer of integration code and functionality, which allows multiple distributed systems to be deployed as though they were a single system. Using these middleware services business applications can transparently access the multiple, backend, distributed, legacy systems and applications (OMG 98, EJB 99, Orfali 96, Nissen 99).

Classic middleware systems typically involve tight binding between the systems and processes at the various organisations (Ouzounis 98e, Thompson 99, Sheth 98). By closely coupling the

organisations, classic middleware systems are able to provide rich functionality, but require expensive development and deployments, pre-agreement in the interfaces used, and carefully co-ordinated, ongoing deployment management (Redlich 98, Hull 99). These systems result in tightly coupled inter-domain systems and thus, these solutions are better suited for use in intra-domain, distributed applications or long-term and closely co-ordinated business partnerships, i.e. static VEs (Ouzounis 99a, Spinosa 98).

In contrast to classic component based systems, which seeks to closely bind the enterprise systems and processes of several organisations into a single closely co-ordinated virtual organisation, cross-organisational business systems can be built using exchange of documents, usually described in XML, to bind together multiple organisations (Sheth 97 and 98, Tombros 99 and 00). Ideally, such an approach would combine the strengths of EDI with the rich interaction, integration, and distribution supported by classic, distributed component-based systems (Stricker 99, Reichert 98, Choy 99). The messaging approach of "fire and forget" seems to be better in the area of cross-organisational communication and co-ordination in comparison with the classical distributed object oriented concepts due to the loosely coupled approach (Veloso 98, Wood 99, JMS 99). The specification, execution and management of internal business processes can be still performed by conventional distributed technologies (Tombros 99 and 00). Different approaches have been proposed for the execution and automation of internal business processes, namely business objects and components, workflow management systems, and recently intelligent mobile agents (Eder 95 and 96, Crossflow, Borghoff 97, Barbuceabu 95, Adams 97, Ciacarini 98).

Business objects and components are network accessible entities with standardised interfaces deployed within a distributed object oriented platform (Orfali 96, OMG, EJB). These objects can be easily bought, extended, customised and integrated into the information system of an organisation. The main objective of business objects is actually the deployment for intra-domain distributed applications (Ouzounis 98b, Choy 99, Chung 99). However, the business object concept can also be used in the area of loosely coupled cross-domain business systems only if additional gateways providing dynamic virtual enterprise functionalities are to be included (Debenham 98, Cost 98).

Another approach proposed for the automation of business processes were workflow management systems (WFMS) that execute, manage, co-ordinate and streamline business processes (Adams 97, Alonso 95, Eder 96, Cai 96, Georgakopoulos 95). WFMS provide an easy and generic way to specify business processes by deploying a business process definition language. The WFMS actually interpret, execute and manage the business processes (Grefen 99, Miller 98). Business processes can deploy legacy systems and components and integrate them into the overall business process specification (Judge 98). However, WFMS have been developed and deployed only for intra-domain business applications (Klingemann 99, Lee 93, Georgakopoulos 98). Due to the needs and requirements of virtual organisations and emerging, open, Internet services, certain ideas and concepts towards cross-organisational workflow management systems have been recently emerged (Tombros 00, Filos 00, Khare 99, Martesson 98).

Finally, another emerging technical area for the development of business processes is the concept of intelligent mobile agents (Magedanz 95 and 99, Krause 96 and 97, Lecihsering 98, Maes 94, Nwana 96). The success story of agents started in the early nineties with the parallel appearance of different agent concepts and technologies. These technologies can be roughly separated into intelligent agents and mobile agents (Maes, 94). The interest in agents was coined

by the increasing notion of Multi Agent Systems (MAS) in the early nineties, driven by the Distributed Artificial Intelligence (DAI) research community (Wooldridge 95). The multi agent system concept is largely conceived upon the idea that complex activities can be split into smaller activities and every small activity can be split into smaller ones, until a primitive set of activities can be found. Every primitive activity in this model can be provided by a special-purpose entity called software agent. Each agent co-operates with other agents inside the community to solve a particular complex problem. Therefore, a multi agent system may be defined as a set of agents that interact with each other and with the environment to solve a particular problem or to provide a service in a co-ordinated and distributed manner (Jennings 93, 94 and 95). If these agents are located in different business domains and execute certain business processes on behalf of their domains then a multi-agent system for virtual enterprises can be formed (Ouzounis 99b).

Intelligent mobile agents provide certain benefits in relation to traditional distributed object oriented approaches. Some of the major benefits emerged from the usage of intelligent mobile agents are autonomy and flexibility, due to the co-operation and co-ordination aspects, scalability, due to the migration capabilities, adaptability, due to intelligent behaviour, and integration with existing technologies, due to the object oriented concepts used to implement agent platforms and agents (Breugst 98, Magedanz 95 and 97).

Additionally, intelligent mobile agents seem to combine all the benefits offered by the messaging systems and distributed component-based systems. Agents communicate by exchanging messages in a similar way like messaging systems (Chess 95 and 98, Bradshaw 97, Byrne 99, ComACM 94). However, agents deploy the concept of globally specified ontologies that make them more flexible and autonomous (Harold 98, Byrne 99, Etzioni 94). Agents are deployed within a distributed object oriented platform, like CORBA or Java, and thus can access any type of standard business component. Additionally, agents can migrate to the physical location of business components preserving the network resources (Fugetta 98 a and b, Fünfrocken 98). Furthermore, agents have the ability to execute and co-ordinate complex business processes, in a similar way like workflow management systems. However, the execution of the business process is not controlled in a centralised way, by a workflow engine, but the agents themselves are co-operating in a flexible, autonomous, and distributed way (Assiss-Silva 96, Bellifernine 99, Ambros-Ingenson 88, Fenster 95, Franklin 96). Finally, when business logic is located in different remote business domains, mobile agents can migrate and deploy it.

The acceptance of agents as an implementation and communication paradigm, the extra capabilities that they offer, like mobility, autonomy, intelligence, adaptability, in conjunction with emerging state of the art agent platforms, like FIPA (FIPA 98, 99) and OMG-MASIF (MASIF 97) and standard distributed platforms like CORBA and Java RMI, flexible content description languages for globally specified ontologies, like XML, emerging, XML-based workflow standards, like WfMC, messaging standards like Java Messaging System (JMS 99) and OMG's Messaging Service (OMG 99), and platform independent programming language, like Java, can provide the basis for the new generation of open, flexible, autonomous, and distributed business process management systems for dynamic VEs.

1.2 Objectives

The main objective of this thesis is to research, analyse, design, develop, test, and validate a platform for the management of dynamic virtual enterprises that will be based on FIPA compliant intelligent mobile agent concepts, emerging, agent-based workflow management concepts for cross-organisational business process execution and management, virtual marketplaces with emphasis on OMG Trader integration, and automated negotiation for dynamic partner selection.

More particularly, this thesis will define, specify, develop and validate the following entities:

- an agent-based, FIPA compliant, virtual marketplace and integration with the standard OMG Trader,
- a XML-based virtual marketplace ontology for business process registration, management of offers, and dynamic partner selection in virtual marketplaces,
- a negotiation ontology and protocol for dynamic partner selection based on FIPA compliant FIPA-Contract Net protocol,
- an XML-based business process definition language for the specification of business processes in the context of dynamic virtual enterprises and a business process repository for the storage of business processes,
- a distributed, agent-based, FIPA compliant, workflow management system for the execution and management of shared business processes across different organisational boundaries,
- an XML-based intra- and inter-domain ontology for cross-organisational, agent-based, business process execution and management,
- provision of shared business processes to the web by integrating the agent based workflow management system with standard web integration technologies,

1.3 Outline of the thesis

In addition to this chapter, the thesis is organised in 9 Chapters that present in detail the state of the art in the area of dynamic VEs, the problem statement, the proposed architecture and the different autonomous agents, the specification and design of each one of them, and the validation and assessment of the proposed concept.

More specifically, in chapter 2 introduction and analysis of the Virtual Enterprise concept is presented. The different terms and definitions of VEs that have been proposed in this area are presented and further analysed. In the sequel, the two broad categories of VE, namely the static and dynamic VEs, are presented and the main characteristics of them are extensively discussed. Finally, a life-cycle model for the specification, registration and management of business processes in the context of dynamic VEs is presented. As a result of this model, a set of key technical and functional requirements that should be fulfilled by a management platform for dynamic VEs are outlined.

Chapter 3 presents an exhaustive analysis of the state of the art in both, technical and functional issues. The chapter starts with an analytical description and assessment of the current projects and academic and scientific results in the area of VEs in relation to the requirements presented in the previous chapter. In the sequel, an assessment of the different technologies and standards

proposed and deployed so far in the area of VEs is given. For all these technologies, an extensive individual assessment regarding their applicability to the dynamic VE concepts and requirements is done. Based on the state of the art and the assessment of the proposed and deployed technologies, the problem statement and the main objectives of the thesis in relation to the most adequate technologies are described.

Chapter 4 presents an overall description of the layered architecture of the system under analysis and identifies the key business domains and relationships, the human roles in each domain and the responsibilities that they have. The architecture consists of three layers, namely the Distributed Processing Environment (DPE), the Mobile Agent Platform layer (MAP) and supporting services, and the virtual marketplace and business process specification, registration and management layer. For the virtual marketplace and business process specification, registration and management layer, which is the core work of this thesis, a set of autonomous and intelligent agents and internal components are identified and presented. These agents will be further analysed, designed and specified in subsequent chapters.

Chapter 5 presents extensively the Mobile Agent Platform layer. More specifically, this chapter is split into two parts, namely the core services of the Mobile Agent Platform (MAP) and the FIPA compliant add on services. In both cases, an analytical description of the provided services is presented. Due to the fact that all the agents under design and analysis are FIPA compliant agents, certain details concerning the design and implementation of FIPA compliant agents on top of the MAP are presented and further discussed.

Chapter 6 presents detailed specification and design of the virtual marketplace agents and the integration and deployment of the standard OMG-Trader service. More specifically, three agents are proposed and analysed, namely the Service Type Agent (STA), the Service Offer Agent (SOA) and the Service Retrieval Agent (SRA). For every agent, the internal architecture and the key components are specified. Then, for every operation that the agent supports, a set of UML sequence diagrams are provided and discussed. In addition to these three agents, the Virtual Marketplace ontology is specified in ACL/XML format.

Chapter 7 presents the detailed specification and design of the business process specification and registration phase. More specifically, the XML-based business process definition language for shared business processes and the business process repository that stores and maintains the business processes is introduced and specified. This phase is actually the phase that the different providers are using to register offers regarding specific business processes in the virtual marketplace.

Chapter 8 presents the detailed specification and design of the business process execution and management phase. More specifically, five FIPA compliant agents are introduced and analysed, namely the Personal User Agent (PUA), the Domain Representative (DR), the Workflow Provider Agent (WPA), the Resource Provider Agent (RPA) and the Requestor Negotiation Agent (RNA). For every agent, the internal architecture, the internal modules, the relationships among them and a set of UML sequence diagrams are provided and discussed. Additionally, for the execution and management of shared business process the inter- and intra-domain ontology are specified and described. Finally, the negotiation protocol and the negotiation ontology used for the automated negotiations are further explained and analysed.

Chapter 9 presents the implementation, testing, validation and assessment of the proposed solution. More specifically, certain details regarding the implementation of the platform and the proposed agents are provided. Furthermore, certain test cases have been developed in order to

test and validate the correctness of the functional specification and the proposed design. The validation of the platform has been done by developing and testing four individually application scenarios. Finally, the assessment of the system has been done in three phases, namely assessment of the emerging XML-based workflow management systems, assessment of FIPA standards, and assessment of the proposed approach.

Finally, chapter 10 presents the conclusions drawn from the analysis, design, development, validation and assessment of the proposed approach. More specifically, an assessment of the contribution of the thesis in relation to the initial objectives and requirements of the thesis and the state of the art is provided. Finally, a set of open issues for future R&D work are presented and discussed. This description can function as a motivation for future and more extensive research in the area of dynamic VEs.

Chapter 2: Virtual Enterprises

2.1 Introduction

Virtual enterprises (VEs) are not just a new area of research and study. Many large industrial companies, e.g. car manufacturers, used to maintain remote or "virtual" business relationships with their suppliers and even with corporate customers. However, the level of integration and the enabling ICTs used, are often not adequate. Many activities are still performed manually and in a complicated way with associated high costs.

Typical examples of virtual enterprises include value added service providers. For example, a telecommunication organisation that provides virtual private networks to its customers. In case that one customer requires a private network connection between two physical locations that belong to two different states, e.g. Germany and Australia, then the initial telecom organisation should establish the appropriate international leased line connections by utilising the network infrastructure of another telecom organisation. This means that Deutsche Telekom, that represents the initial customer service company in Germany, should co-operate with MCI WorldCom and Telecom Australia to provide the international leased line. All three companies should jointly co-operate, share resources, and business processes in this complex activity so as to provide a final end-service to the customer. The provided leased line is a service offered by three companies that committed to serve this customer for the whole duration of the existence of the leased line. The final service is provided in a transparent way to the user; the user does not know the existence of the three telecommunication organisations.

With the broad advent of networked organisational forms and the emergence of new electronic business paradigms, business terminology is becoming confusing and expressions such as the "extended enterprise", the "virtual organisation", the "networked organisation", "supply chain management", and “cluster of enterprises” sometimes used interchangeably and need further clarification.

The concept of extended enterprise is better applied to an organisation in which a dominant enterprise "extends" its business boundaries to all or some of its suppliers, whilst the VE can be seen as a more general concept of including other types or organisations, namely a more democratic structure in which the co-operation is peer to peer (Afsarmanesh 99, Reichert 98). In this sense, an extended enterprise can be considered as a particular case of virtual enterprises.

The concept of virtual organisation is similar to that of a VE, comprising a network of organisations that share resources and skills to achieve its mission/goal, but no limited to an alliance or enterprises (Doz 98, Adams 97). An example of virtual organisation could be a virtual municipality, associating via computer networks, all the organisations of a municipality, e.g. water distribution services, leisure services, etc. A Virtual Enterprise is, therefore, a particular case of virtual organisation (Doz 98).

The term networked organisation is perhaps the most general one referring to any group of organisations inter-linked together by a computer network, but without necessarily sharing skills, resources, processes, or having a common goal (NIIP 96). Typically, networked organisations correspond to a very loose type of organisation.

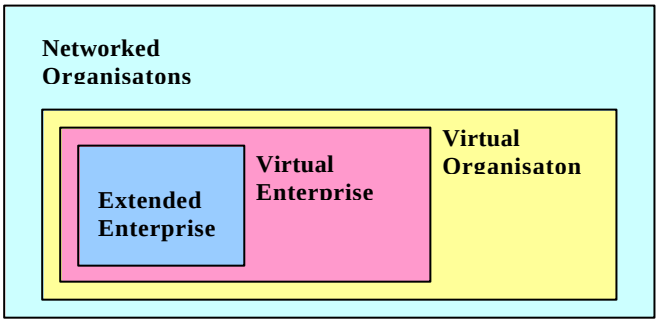

Figure 1: Virtual Enterprise Models

The supply chain management term refers to the policies and supporting mechanisms to manage the flow of materials in a value chain, possibly covering several aspects from the raw material suppliers to the consumers, and involving the product manufactures, distributors, retailers, etc. and supported by the flow of information between the supply chain participants (Camarinha-Matos 98, Zarlin 99). This concept is traditionally applied to organisations that are relatively stable, i.e. where the core competencies remain the same for a large period of time, however more dynamic supply chains are becoming current. The focus in this approach is on the logistics for material and related business information (Georgakopoulos 98, Geppert 98).

Finally, cluster of enterprises is a group of enterprises that have the potential and the will to co-operate and therefore, may become partners in a VE. These enterprises are normally registered in a directory, where their core competencies are declared. Based on this information, a VE initiator can select partners when a business opportunity is detected.

In the context of this work the following generic definition is adopted (Ouzounis 98c, Block 95, CIMOSA 98): "a VE is a network of different administrative business domains that co-operate by sharing business processes and resources to provide a value-added service to the customer. Each member of the virtual enterprise will contribute primarily what it regards as its core competencies, i.e. business processes and resources. There is a time limit on the existence of the virtual corporation caused by fulfilment of its business purpose. From the viewpoint of an

external observer, i.e. a customer, the virtual enterprise appears as a unitary enterprise." Figure 1 presents the relationships among the different types of virtual enterprises.

However, most of these business scenarios have specific, business sector related characteristics, and they, in general, follow some generic principles and models. The most important features of VE are (Ouzounis 98e, Block 95, CIMOSA 98, Zarlin 99, Carr 96):

- more than one independent administrative domain is involved in the provision of the service to the customer
- the service provision is performed by sharing resources and business processes, i.e. by establishing business relationships among the different VE partners,
- the sharing of processes and resources lasts for a limited period of time even only for only one service provision,
- the business process interfaces among the business domains, i.e. the way that one domain deploys the processes and resources of the other, might be static, pre-defined, and fixed or dynamic, based on a set of globally specified templates,
- the number of VE partners might be static or dynamic according to the needs and requirements of the partners,
- the partners are physically distributed and are connected with electronic means and systems,
- the provision of the service to the customer is done in a transparent way by one representative partner.

The above mentioned features prescribe the key selection criteria that will be used to classify the existing VE concepts and models. Some of the most important criteria are (Ouzounis 99a, Doz 98, Adams 97):

- **Lifetime of the relationship:** whether the time limit of business relationships among the VE partners is short, medium, or long term. The relationship can last for only one service provision up to some months or years,
- **Number of VE Partners:** whether the number of partners is static and pre-determined or it is dynamic and flexible and can change any time leading to the evolution of the VE,
- **Degree of Autonomy:** whether the VE partners are tightly coupling their business processes or they can change any internal process maintaining their autonomy within the context of the partnership,
- **Degree of Distribution**: whether the network is controlled by a centralised entity, is distributed or it is market oriented with the support of a virtual marketplace that provides directory services for identification and selection of suitable partners,
- **Degree of Evolution and Scalability:** whether the VE can evolve in terms of new members and relationships and if the business model is scalable,
- **Degree of integration**: whether a VE member has fully integrated their business processes with other organisations following a tightly-coupled model or a loosely coupled model with limited integration,
- **Focus on process efficiency vs. focus on per-transaction efficiency and value:** whether the VE members are co-operating on process level or on transaction level with direct

implications on integration and lifetime relationships.

The above key criteria will be used to classify the existing proposed models of VE. In the following section a classification of different VE models is given and certain conclusions are drawn regarding the applicability and complexity of each model.

2.2 Categories of Virtual Enterprises

Although there is no strict academic definition regarding VE, different VE models feature common business and technical characteristics and attributes. Deploying the above specified criteria as classification criteria, two well-defined categories of VE can been identified (Ouzounis 00a, Malone 91, Banahan 99, Stricker 00). These are:

- Static Virtual Enterprises,
- Dynamic Virtual Enterprises.

In the following section, these two generic VE categories are further analysed and certain examples are provided to better clarify the different models, concepts, and benefits and drawbacks that both approaches share.

2.2.1 Static Virtual Enterprises

In Static Virtual Enterprises (SVE) a set of business partners is linked together in a static and fixed way, i.e. the shared business processes are tightly integrated. The business relationships among the partners, i.e. the process interfaces are pre-defined, tightly coupled, fixed, well-integrated, and customised among the partners (NIIP 96, Malone 91, Afsarmanesh 99). The network is fixed and pre-determined and thus, the structure of the VE is static and pre-determined. Based on the distribution and management style of the network, two types of Static VEs can be identified, namely centralised and decentralised (Ouzounis 99a, Stricker 00).

In the Centralised Static VEs (CSVEs) a dominant business domain co-ordinates the business relationships among the members of the network, enforces the technical interfaces for business integration among the partners, integrates the processes of the partners by creating shared processes, and manages the underlying technical infrastructure and the shared business processes of the partners in a static and centralised way (NIIP 96, Wognum 99 a and b). Partners and the central organisation form long-term relationships and focus on investment returns over the lifetime of that relationship. Finally, the establishment of the VE is performed manually, in a customised way, and under the full control of the dominant organisation. The required integration, development, and re-engineering costs are high for all members (McCaffer 99, Nwana 99).

Typical examples for CSVEs are models that have been applied in the automotive manufacturing business (Zarlin 99, Geppert 98). In that case, a big automotive manufacturer has a network of suppliers, distributors, and re-sellers that are working together in different phases of the production, distribution, and reselling process. The big manufacturer has specific needs and requirements and enforces his requirements in order to increase the degree of automation and decrease the production and distribution costs. The network of suppliers, resellers, and distributors closely co-operate with the central dominant business domain by adopting and integrating the pre-specified interfaces.

In Decentralised Static Virtual Enterprises (DSVE) different business partners are linked together in a rather autonomous and decentralised way. This type of network is similar to the previous one except that there is no central, dominant, management organisation and each member of the network may co-operate with many other domains (Malone 91, Ouzounis 98c). None of the partners has full control over the network and the underlying infrastructure, while integration among the business processes of the members is being performed in a jointly, co-ordinated, and incremental way. Partners form long-term business relationships and gain investment returns over the lifetime of those relationships (Mohan 98, Fredederix 98). Finally, the establishment of the VE is performed manually and in a customised way addressing the specific technical requirements of the partners. The development and integration costs are rather high, while the evolution of the network is rather impossible (Zarlin 99, Wognum 99a).

High tech manufacturing today exemplifies this model. Organisations such as semiconductor fabs and board assembly houses focus on one activity in a complete value chain and then partner with multiple other organisations in order to play a role in multiple value chains. Every partner plays a role in the VE and contributes primarily its own core competencies, i.e. business processes and resources. In high Tech manufacturing the VE members can work on the production and assembly of new products, as well as, on the distribution of products to different re-sellers.

A more recent approach to automate the process of forming a static VE is to use a virtual marketplace or a directory service where potential VE members register their resources and business processes (CrossFlow99, Tombros 99 and 00, Ouzounis 98b). The virtual marketplace provides matchmaking services to business domains that want to locate VE partners (Spinosa 98, McCaffer 99). Human operators searching the marketplace and locate potential partners that can provide specific processes. Then, a manual, human-driven negotiation process starts for the selection of the most appropriate VE candidate partner. With this approach, the time required to find partners and establish business relationships is improved. This approach takes advantage of the new, open, Internet economy and significantly improves the formation process of the virtual organisations. However, after the formation of a VE, the business relationships among the partners, i.e. the interfaces among the shared business processes, remain static and fixed, while the evolution of the VE in terms of new members, that might provide better processes with better terms, is rather impossible (Hoffner 97, 98 and 99).

Marketplaces can be used in a more effective and dynamic way not only in the formation phase of VEs but also in the execution phase (Ouzounis 99b). This means that the partners that are involved in the provision of the shared process are changing continuously and dynamically according to the requirements of the customer and the processes. In that case, for every business process execution a new VE is being created in a dynamic way addressing the needs and requirements of the customer and the individual partners. The deployment of marketplaces not only for the establishment of VEs but also during the provision of shared processes can lead to significant improvements (Ouzounis 99b).

In the following section, the benefits that virtual marketplaces introduce during the business process management and execution are further explained.

2.2.2 Dynamic Virtual Enterprises

In Dynamic Virtual Enterprises (CSVE) a set of business partners is linked dynamically, on-demand, and according to the requirements of the customers, by deploying a virtual marketplace.

The business domains do not have fixed business relationships and thus the VE is not static and might change continuously based on market driven criteria (Ouzounis 99b, Fielding 98, Doz 98).

The virtual marketplace provides services for the registration of partner process offerings based on some generic, well-known, globally specified process templates. Business domains that want to form VE relationships can register offers on the marketplace related to certain process templates. Whenever a business domain wants to use a particular process, searches the marketplace, and locates all the potential partners that can provide the service. As soon as the list of VE candidate partners for one particular process has been found, the selection process starts. The selection process between the domains is usually performed through negotiation. The negotiation process might be either, manual, or automated, while the result of it is usually a short term contract that regulates the business relationship among the involved domains (Gepert 98, Grefen 98, Weitzel 99).

By deploying virtual marketplaces, there are no explicit static business relationships among the partners and thus, no integration among the processes of the partners is required. Marketplaces are usually organised around certain globally specified service or product templates that can be offered by the different vendors. The marketplace is a match making mechanism that brings potential process providers together with potential users of these processes. Although marketplaces and matchmaking mechanisms have been used for some time for business to consumer electronic commerce purposes (Kasban 98, Ebay 98, Yahoo 96) they have not been actually deployed for dynamic VE purposes (Ouzounis 98e). The main reason was the lack of technologies that enable the easy and flexible definitions of process templates, mechanisms for automated negotiation, and autonomous interaction among different domains. Due to the advent of eXtensible Markup Language (XML) (W3C) and its ultimate acceptance, as Internet meta-language, concepts like virtual marketplaces have started to appear (Ouzounis 99a, Zarli 99, Mitrovic 99).

The primary focus on virtual marketplaces is on efficiency of transactions and maximisation of value per cost of each vendor's offer. Organisations may participate in the marketplace only briefly or they may be long term members. Relationships between process users and process providers tend to be short term. Thus, investment returns are gained over single transactions, as well as, over the time span of the marketplace participation. The number of members of the network can easily change and thus, the structure of the VE can change from one service provision to another according to the specifics of the customers and the current needs of the members. This is a significant evolution mechanism that takes advantage the demand and supply, i.e. the process offerings by the individual domains.

Based on the distribution and management style of the network, two types of Dynamic VEs can be identified (Malone 91, Alonso 98):

- Centralised, when the owner of the marketplace is a VE partner. This domain manages and administers the virtual marketplace and enforces specific process templates. Although, from technical point of view, it is possible to organise a VE in terms of a Centralised Dynamic VEs (CDVEs), from business point of view is rather unusual. The main reason is that the marketplace should be a trusted, third party provider that is not involved into the VEs. Centralised dynamic VEs can be deployed by very big organisations that would like to transit from the Centralised Static VEs into more dynamic cases (Zarli 99, Geppert 98 a).
- Decentralised, when the owner of the marketplace is a third party provider that has no relationship to the registered partners. This is probably the most advanced and flexible

model that features the most benefits. However, the required business systems and technologies are far too complex and for the moment immature (Ouzounis 98c).

An interesting area where Decentralised Dynamic VE (DDVEs) concepts are applied is the area of trading communities. A characteristic example is the area of logistic companies. In that case, logistic companies can register their processes into a specialised marketplace. A potential process might be the delivery of parcel where the properties of the process might be the reached destinations, the price, the time needed to transfer the parcel, the offered guaranty, the transformation media, etc. Then, business domains that want to use a logistic service, search automatically on the trading community, select the best partner that exist at this moment on the marketplace, based on certain requirements, and use the service. For the initial customer of the VE the whole process is total transparent. As the offers in the marketplace change, i.e. new companies register and deregister with better terms, conditions and prices, then the selection of the best partner depends on negotiation practices. The VE might exist for only one service invocation or for more. However, if companies want to take advantage of the market conditions enabled by Internet-based commerce, they should frequent deploy the capabilities of the virtual marketplace in order to get better prices and quality of service.

The above scenario illustrates the key elements of DDVE. The VE usually exists only for the duration of a single service provision. Certain selection and negotiation requirements specify the VE partner that will be selected each time (Billington 99). The evolution of the VE is granted due to the loosely coupled relationships among the partners and the marketplace capabilities. The registration of process offerings on the marketplace is based on globally specified service templates (Tombros 99, Wognum 99). The marketplaces are becoming more specialised and closely related to specific industrial sectors. In long term, special trading communities for specific industry sectors will be created. The form and relationships among the partners of the VE can change continuously. The process offerings registered in the marketplace can change dynamically and on-demand according to the demand and supply (Ouzounis 99b).

It is obvious that dynamic VEs improve significantly the static ones and take full advantage of the open, global, opportunities offered by the Internet and the global economy. In the following section a more formal and focused assessment of the static and dynamic VEs is provided.

2.3 Evaluation of Virtual Enterprise Categories

The classification and assessment of the basic VE models, proposed so far, will be done with the previous mentioned classification criteria (Camarinha-Matos 99, Malone 91, Mitrovic 99). Therefore, a comparison of each model against the classification criteria leads to the following conclusions:

- **Lifetime of the relationship:** DVEs feature very short lifetimes, while SVE feature longer ones. In the former case, the relationships are static, well-integrated and thus, no flexible enough for alterations, modifications, and evolution. This dimension also determines the time period over which investment returns must be achieved,
- **Degree of integration**: Tightly coupled SVEs, which function essentially as a single virtual organization, exhibit high process integration between partners. Loosely coupled DVES are at the far end of the spectrum and show very low process integration between the partners,
- **Number of VE Partners:** In SVEs, the number of partners participating in the VE is static and pre-determined due to the specialised integration activities required. In DVEs the

number can change dynamically, upon demand and supply, and based on the requirements of the individual members of the marketplace,

- **Degree of Autonomy:** SVEs require high degree of integration among the partners and thus, the degree of autonomy is rather low. The business processes of one partner are highly depend on the others. On the contrary, DVEs feature more autonomicity because the relationships among the partners are not static and well-integrated. Thus, any changes to business processes can easily be done.
- **Degree of Distribution**: All the above mentioned models have a good level of distribution among the business processes of the partners. However, SVEs are based on a centralised dominant model, while DVEs reveal, due to the nature of the model, the highest level of distribution and autonomicity among the business processes and partners.
- **Degree of Evolution and Scalability:** In SVEs the relationships among the partners are static and thus the level of scalability is low. It requires high development costs to re-design the network and change the interfaces among the partners. On the contrary, on DVEs there are no tightly coupled interfaces among the partners and thus, scalability and business evolution is a key issue.
- **Focus on process efficiency vs. focus on per-transaction efficiency and value:** As we saw above, partners that work as part of a larger virtual organisation focus on achieving overall process efficiency. Partners that work on a per-transaction basis need to focus on achieving efficiency and value within the individual transactions.

In the following Table 1, a summary of the above analysis and discussion is illustrated.

	Static Virtual Enterprises	**Dynamic Virtual Enterprises**
Lifetime	High	Low
Integration	High	Low
Number of Partners	Static	Dynamic
Autonomy	Low	High
Distribution	Medium	High
Evolution/Scalability	Low	High
Process Efficiency	High	Medium
Transaction Efficiency	Low	High

Table 1: Comparison of VE Categories

Based on the above selection and categorisation criteria, it is obvious that DVEs are a more promising business model with a lot of interesting features. Due to the open mechanisms of Internet economy dynamic, flexible, autonomous VEs that take advantage of the market conditions are preferred.

Although from business point of view DVEs are the most promising business model, from technical point of view the required technical solutions and systems are more complex, sophisticated and distributed (Ouzounis 98d, Alzaga 99, Carr 96). However, the advent of

Internet and open communication protocols, like TCP/IP and HTTP, distributed middleware systems, like CORBA-IIOP and Java RMI, and extensible meta languages, like XML, provide the basic building blocks for the development of management platforms that will realise the concept of DVEs.

In the following section, a life-cycle model for the establishment and management of VE is presented. This model prescribes the key phases, activities, and domains required for the management of dynamic VEs.

2.4 Life-Cycle Model for Dynamic Virtual Enterprise

A life-cycle model usually describes the key phases and activities required during the existent of an entity. According to ISO (ISO 91 and 94), a life cycle can be defined as "the finite steps a system may go through over its entire life history. The different life cycle phases define types of activities which are pertinent during the life cycle of the entity". In our case, the VE life-cycle model consists of two key phases that should be followed for the establishment and management of a VE. Every phase consists of more specific steps that describe the main operations that should be done by different human roles from technical point of view. It should be noted that the following life-cycle model is best applied in the DVE model that is the core work of this thesis.

In order to better understand the life-cycle model, the following scenario is provided. A Company called *On-line-Books* sells books to customers on-line. Part of the book-selling business process is the distribution process, i.e. the delivery of the book to the customer. On-line-Books has not a particular way to distribute the books and looks for a logistic partner to outsource this process. The On-line Books company knows exactly the properties and attributes of the distribution process. In order to find potential partners with logistic capabilities, On-Line Books deploys a third party virtual marketplace that provides matchmaking services for logistic companies.

The virtual marketplace specified several logistic process templates for different logistic services. One of the logistic process templates is the book delivery process. This process template has, for example, as properties the destination, the price for the delivery, the payment method, when the payment should be done, delivery day, the guaranty in case of problems, etc. Logistic companies, that can deliver books, use the standard book delivery process template and register their process offerings into a particular marketplace by specifying their terms and conditions.

When a customer orders a book from the On-line-Books, the company searches the marketplace, locates the potential logistic partners, negotiate with them about the price, location, delivery day, time, quantity, etc. and selects the most suitable one. The selection of the partner is highly associated with the characteristics of the customer, i.e. his location and preferences. Then, the Book-On-Line uses the logistic process provided by the selected partner and serves the customer. When a new customer comes and places a book order into the shopping system of the company, the company uses again the marketplace and locates, negotiates, and selects probably another logistic company that can better satisfy the requirements of the new customer.

The above scenario is a very typical, though simplistic, one that reveals most of the characteristics of the DVEs concept. In order to support a scenario like this, a management platform for the management of DVEs is required (Ouzounis 98d, Stricker 00, Camaritha-Matos 99). In more general terms, the following definitions and concepts are provided in relation to the life cycle model and this thesis specifically.

In general, a VE is a set of business domains that jointly and dynamically co-operate to provide value-added services to a customer in a transparent way, i.e. the customer does not know about the existence of the different business domains involved in the service provision. A business domain is a administrative domain that pose its own resources, infrastructure, and services and impose its own restrictions and regulations on them in terms of access control and authentication. Business domains jointly co-operate by sharing services, i.e. one business domain deploys services provided by one or more other business domains in a consistent and well-regulated way. Every request for a service deployment from one domain is checked for permission by the requested domain. The access control and authorisation of requests before the service provision is based on a temporary contract that has been agreed by both domains, i.e. by both the requestor and supplier. The business domain that requests a service by one domain is called the requestor while the domain that provides the service is called the supplier.

The technical representation of a service is a business process. The business process can be either, local or remote. All the processes provided in self-contained manner by this domain are called local processes. Therefore, a domain has full control of its own local processes and can impose any type of access control constraints. On the contrary, when a process could not be provided by one particular domain, but should be deployed by a remote one, is called remote. A process that is considered remote for domain A, is local for the supplier of this process. This means that local and remote processes can be represented technically in a similar way but they differentiate to the way that domains "look at and interpret " them.

The domain that provides a service, i.e. a business process, directly to a customer is called the VE representative. The process that is provided by the VE representative to the customer is called VE process. The VE representative domain represents the VE in the outside world in a similar way like a normal company. The domains that participate in the provision of VE processes in the context of the VE are called VE partners. Initially, when the different business domains have no relationships among them, i.e. they do not share any processes, are called VE candidate partners. A VE candidate partner is becoming VE partner after a negotiation process that involves the potential requestor of the process and the potential supplier of the process. When an agreement is reached then the potential supplier becomes a VE partner. This negotiation process is done dynamically and during the provision of VE process to the customer. The agreement might last for only one business process deployment or for several ones.

A normal business domain becomes potential VE partner when it registers the business processes that can offer to a third party marketplace. In that case, the domain specifies which local processes can be provided to other domains and under which terms and conditions these processes will be provided, e.g. price. The virtual marketplace provider maintains for different processes different service templates that describe the service. A service template has a name, a set of named properties and can be associated with other existing service templates. When a VE candidate partner declares that can offer a particular service, it always associates this offer with an existing service template. When a service template is not available, a new one can be generated by the marketplace administrator for consistency. When a service can be provided by different VE candidate partners, then offers associated with this service are stored and managed by the marketplace. Each individual VE candidate partner can change dynamically the context of its own offer by updating or improving it.

Local and remote business processes are represented technically in the same way using for example a business process definition language. In general, a process has a name, a set of input parameters, a set of output parameters, a set of sub-processes, a set of tasks, and a set of

conditions. The input parameters are the input values to the process, while the output parameters are the output values of the process. A process might consist of sub-processes in a recursive manner. For every subprocess, in a similar manner like the process, a name, input and output parameters, sub-processes, sub-tasks and conditions can be specified. This leads to a directed acyclic graph of processes, subprocesses and tasks. A task is considered the final, unique, elementary piece of activity that can be included within a process. Actually, tasks are the computational elements of the process while processes and sub-processes orchestrating and co-ordinating the scenario of the process by scheduling the tasks based on conditions. With every process, sub-process, and task conditions can be associated. Conditions are logical expressions related to input, output, and external values with some logical operators. When a condition, which is related to a process or task, is true, then the associated process or task should be scheduled. By this decomposition of processes into sub-processes and tasks a complex service can be easily described. The specification of a business process could be done by using a business process definition language. This business process definition language provides all the necessary syntactic means to specify processes, sub-processes, tasks, and conditions.

When a process consists of sub-processes and tasks belonging to the same administrative domain and can be provided in an autonomous way by this domain, then the process is called local. If there is one sub-process that cannot be provided by this domain, then this sub-process is called remote and, in that case, a supplier domain should be found. If, for this remote process, one static supplier has been found, then the VE relationship is called static. If, for every remote process, a supplier is found dynamically during the process provision, then the VE relationship is called dynamic. If the partners of a VE have not static relationships among them, but on the contrary, they negotiate among each other during process execution then the VE is called dynamic VE.

From the above description and definitions, it is obvious that different administrative domains participate in the execution and management of dynamic VE services. These domains are the:

- **Customer domain**: this is the domain of the user that deploys the services of the VE. The user in this domain can start a service, suspend, resume, or terminate it. When the service is completed the results of the service are returned to the customer. Additionally, if, during the execution of the service, a critical situation occurs, the service is aborted and the customer is informed about the event.
- **VE representative domain**: this is the domain that the customer logs on and requests certain services. This is actually the domain that represents the VE to the external world. It executes and manages processes in a transparent to the customer way by deploying the capabilities of the marketplace and the remote processes of other business domains. The VE representative domain provides and manages the execution of the VE services by conducting the marketplace, locating candidate partners, negotiating with them, and selecting the best one for the execution of the remote processes,
- **VE Candidate/Partner domain**: this is the domain that offers a set of business processes to the marketplace community and registers certain offers related to specific service templates for potential co-operation with other domains. If this domain is finally selected after a negotiation process it becomes the VE partner domain that will provide the agreed processes to other domains,
- **Virtual Marketplace domain**: this is a third party domain that provides the service templates that the VE candidate partners use to register their offers. This domain manages

the service templates, the offers registered by the VE candidate domains, and provides retrieval services for the selection of VE candidate partners. This domain does not actively participate in the VE and thus does not provide any type of business process management services.

Having defined the key domains and the roles taking part in dynamic VEs, the lifecycle model finally consists of the following key phases:

- **Business Process Specification and Registration Phase**: during that phase the different business domains should specify their local and remote business processes. The specification of business process is performed by deploying a business process definition language. In the sequel, every domain registers for every local process corresponding offers to the virtual marketplace by declaring the terms and conditions under which these local processes will be offered to other domains,
- **Business Process Management Phase:** during that phase, a business domain offers services to customers by deploying the dynamic model of the marketplace. Whenever a customer requests a service, the corresponding domain initially starts the provision of the service. If the requested service consists of remote sub-processes, that should be provided by other domains, then the business domain conducts the marketplace, locates all the potential VE candidate partners, and negotiates with them dynamically in order to select the best one that satisfies certain selection and negotiation criteria. When a VE partner has been found, then the initial domain, that serves the customer, requests the remote process from the newly selected VE partner. The provision of the whole process is totally transparent for the customer. During the provision of the service, the customer can manage the service, i.e. he can suspend, resume, or terminate the execution of it. Every management request from the customer, e.g. suspend, is forwarded to all VE partners, i.e. the management of VE services should be performed in a autonomous, distributed and cross-organisational way.

2.5 Requirements for the development of Dynamic VE Systems

In order to support such dynamic business scenarios an open, flexible, underlying management platform that supports easy integration and automation of business processes that span different business domains in an effective and well managed way is required (Malone 91, Gibon 99, Wognum 99).

The key functional and technical requirements for the development and deployment of such a platform are:

- specification and storage of business processes that can be executed and managed in a distributed, autonomous, and dynamic way in the context of dynamic virtual enterprises, i.e. across-organisational boundaries,
- flexible and dynamic mechanisms for distributed, autonomous, and loosely coupled co-operation and business process execution among different business domains (Tombros 99 and 00, Ouzounis 99b),
- registration and management of core business processes that can be offered to potential VE partners in an open, third party, virtual marketplace (Ouzounis 98d),

- dynamic selection of VE partners based on business process offerings stored in virtual marketplaces and support for automated negotiations through simple selection criteria,
- access control and authentication of business process requests coming from remote business domains based on electronic contracts that have been established during the automated negotiation process (Carr 96, Borghoff 97),
- flexible mechanisms for business process template management and maintenance and administration of process offers within virtual marketplaces,
- flexible and easy adaptable ontologies for business process execution and management across organisational boundaries (Zarlin 99, Georgakopoulos 98),
- flexible and easy adaptable ontologies for virtual marketplace deployment from both business process providers and requestors (Tombros 99),
- flexible and easy adaptable ontologies for automated partner selection and negotiation (Ouzounis 99b),
- provision of shared business processes to customers through the web in a transparent and flexible way by hiding the dynamic relationships among the different business domains (Bolcer 99),
- integration of existing legacy systems and business components with business processes in the context of dynamic VEs (Orfali 96, Choy 99, Breugst 98).

In the following chapter 3, the emerging state of art technologies that can be used for the development of dynamic VEs systems are presented and analysed. These technologies vary from the traditional EDI systems, to distributed, component-based business systems, to XML-based messaging systems, to workflow management systems, to intelligent mobile agents, and to virtual marketplaces. Analysis of each technology and the benefits and drawbacks that it has in the context of dynamic VE is presented.

Additionally, an exhaustive analysis of the current state of the art in the area of dynamic VEs in relation to certain research and development projects and technologies is presented. The objective of this analysis is to evaluate and assess the existing solutions and proposals in the area of VEs and identify the key open R&D issues that need to be solved. These key open issues, in relation to the technologies selected in the previous chapter, and the above stated requirements will be the basis for the analysis, design and development of the management platform for the dynamic VEs.

2.6 Summary

In this chapter analysis of the Virtual Enterprise concept is provided. More specifically the different terms and definitions of VEs that have been proposed in this area are presented and analysed. Though a fully agreed term of VE has not been emerged in the academic world, this thesis proposes and adopts one. In the sequel, the two broad categories of VE, namely the static and dynamic ones, are presented and the main characteristics of them are extensively analysed. Based on some classification criteria these two categories are evaluated and assessed and certain conclusions are drawn. Furthermore, a life-cycle model for the creation and management of dynamic VEs is presented. The life-cycle model specifies the main activities required for the specification, deployment, and management of shared business processes in the context of

dynamic VEs. Additionally, certain key definitions are provided and discussed. This model actually determines the key administrative domains, the human roles, and the functional activities involved in the establishment and management of dynamic VEs. As a result of this model, a set of key technical and functional requirements that should be fulfilled by a management platform for dynamic VEs, are presented.

Chapter 3: Virtual Enterprise Infrastructure

3.1 State of the Art in Virtual Enterprises

A rapidly increasing number of projects and R&D activities worldwide are addressing different technical and business aspects of virtual enterprise technologies and infrastructure.

The National Industrial Information Infrastructure Protocols (NIIP) project started at late 1995 in the USA and it was perhaps the first biggest and most significant project in the area of VE. In reality, NIIP is more a workprogram than one consistent project. NIIP intends to support the formation of industrial VEs and to provide technologies that allow VE participants to collaborate within a heterogeneous computing environment. In its general scope, NIIP addresses the complete VE life-cycle, i.e. establishment, execution, and completion. The NIIP bases its developments on open, standard, core technologies such as the Internet and CORBA (OMG 98), related distributed object oriented technologies, product modelling and description techniques, like the Exchange of Product Model Data standard (STEP 96), and information modelling technologies, like workflow management systems (Georgakopoulos 98). Based on this reference architecture, a number of pilot projects have been launched to develop prototypes, e.g. SMART, Solutions for SME Adaptable Replicable Technology. NIIP is based on a very "harmonised" view of the business world and it is too much focused on the US-based reality and interests. According to NIIP's concepts, all enterprises should work co-operatively by sharing all kinds of services, and resources, including humans. This approach is rather too generic and optimistic and probably, not in compliance with the current reality in most business sectors (NIIP 96). Therefore, although NIIP can be considered as a Reference Architecture to be considered before any new development in the VE area, it can not be easily adopted and deployed due to its generality and high level of abstraction. More specifically, the NIIP project developed concepts and prototypes for the static VEs. The selection of partners is performed manually, without using any type of matchmaking mechanisms, and as a consequence, the evolution of the VE could not be easily performed. The execution and management of shared business processes is done by

shared, tightly coupled, business objects located in different physical and administrative locations. The interfaces among these business objects are static, pre-defined and well-agreed by the different partners. An alternative way for executing and managing business process es was the deployment of workflow management systems. In that case, the workflow management system is used for the management of internal business processes. The cross-organisational business process management is performed by the exchange of events generated and consumed by specialised gateways. In general, the NIIP project has not proposed so far a consistent approach for cross-organisational business process execution and management.

The X-CITTIC, Planning and Control Systems for Semiconductor Virtual Enterprises, is an Esprit funded project focused on VEs for the semiconductor industry (X-CITTIC 97). In this application domain, the manufacturing process is associated with sales order originated by a customer that can be located anywhere in the world. The management of sales orders can be accomplished through a globally distributed manufacturing network that can manufacture different pieces of the product on-demand. X-CITTIC expected to raise, to the virtual enterprise level, some of the techniques currently available in a modern shop floor (Veloso 98). Examples of such techniques are event-driven planning, scheduling, dispatching, and order release. The project also worked towards the direction of static VEs, where the establishment and configuration of the VE is performed manually and in a centralised manner. The execution of the shared business processes is done though special gateways that control the manufacturing control units (Adams 97). The management of shared process is performed in terms of events generated and consumed by the different partners. The semantic meaning of these events is tightly coupled with the business process that will handle the events. Events generated from one business process in one domain are forwarded in the domain's gateway (Debenham 98). The gateway locates the corresponding VE partner domain that will consume the event and forwards it to its gateway. The receiving gateway is responsible for the management of it by forwarding it to the internal process that will handle it. The relationships among the event consumers and providers are static and are not regulated or controlled by market driven approaches, like virtual marketplaces. The links between the different gateways are specified statically and could not be changed easily. Every domain pre-defines the events that can handles (Grefen 99).

The goals of MARVELOUS, an end user driven ESPRIT funded project, are the identification and harmonisation of generic requirements for use of advanced IT in manufacturing and engineering across the maritime industry (MARVELOUS, 97). The project intended to guarantee consensus on requirements across the whole range of maritime users and to work closely with the technology providers in order to facilitate the formation of VEs. It also tried to ensure that the end-user requirements are feasible and can be translated into product developments. The project deployed open standards and distributed object-oriented technologies for the execution of business processes. The execution of shared processes is performed by specialised business objects, which are located in different partners (Cost 98). The relationships and integration among these entities is static and pre-defined leading directly to the concept of static VEs, while the business objects have tight coupled relationships among them. This means that no direct market oriented mechanisms are involved for the selection of partners. The integration of shared business processes is done in a manual and static way (Miller 98).

The VEGA project, Virtual Enterprise using Groupware tools and distributed Architecture, aims to establish an information infrastructure to support the technical and business requirements and operations of Virtual Enterprises (VEGA 98). Groupware tools and distributed architectures are being developed in compliance with product data standardisation activities (STEP) and the current trends adopted by the forthcoming international industrial groupware specifications, for

example the OMG (Zarli 99). The approaches and developments resulting from a number of other ESPRIT projects were extended and the strategy for application integration by the distribution of a concurrent access to STEP databases were explored (Zarli 99). A complementary route involves the design of a CORBA Access to STEP models (COAST) infrastructure to support the distribution of a product data by means of updated object broker technology. The VE partners are sharing production data stored in distributed federated databases managed by different domains. The main objective of VEGA was to provide a mechanism for sharing STEP oriented product designs across different domains for the manufacturing and production phase and thus, no major emphasis has been placed on the business process specification, execution, and management (Zarli 99).

The PRODNET II project, Production Planning and Management in Virtual Enterprise, aimed at the design and development of an open platform to support industrial manufacturing VEs with special focus on the needs of Small and Medium Enterprises (SMEs) (PRODNET II 98). The basic platform of PRODNET II includes, a Messaging System, for the exchanges of EDIFACT and STEP messages, a Co-ordination Module, for the execution of shared VE processes based on event management and CORBA remote requests, a Configurator, allowing the definition and parameterisation of the VE and the behaviour of each node, a distributed business process management system, that provides a proprietary first level coordination mechanism of business process execution at the VE level by supporting monitoring mechanisms, and finally a user driven partner search and selection mechanism without negotiation support based on public virtual marketplaces (Camarinha-Matos 99a). The execution and management of shared business processes is based on message passing among distributed CORBA objects (Camarinha-Matos 99b). The integration of shared business processes is pre-determined and based on user driven matchmaking services. The VE partners exchange EDIFACT messages for only electronic commerce purposes. The matchmaking service is a general-purpose directory service used to store company profiles related to certain processes and products. In general, the project does not address a generic mechanism for inter-domain business process execution and management. Additionally, the selection of partners in the VE is done in a manual and add-hoc way without negotiation process (Camarinha-Matos 99c). Finally, the co-ordination of business process is done by the exchange of standard EDIFACT messages. The EDIFACT messages are only adequate for electronic commerce purposes and could not be applied for generic business processes (Pereira 99). Other similar projects working in the area of static virtual organisations/enterprises, manufacturing and distributed business process execution based on messages and events include Globeman21 (Globeman21 99), ELSEWISE (ELSEWISE 98), INDEMAND (INDEMAND 98), and MISSION (MISSION, 97).

The VENTO project, A Virtual Enterprise Organiser-Development of Advanced Groupware tools supporting synergy among enterprises in the emerging global market, aimed in the adaptation and integration of groupware tools to an integrated, inter-domain system that will operate in a distributed environment and will provide workgroups facilities and workflow management (VENTO 98). VENTO consists of a Workgroup Engine, that offers document management, history facilities and email functionality, Workflow Management System, providing with functions for workflow administration, process definition and process tracking, and Integration Engine, establishing an object-oriented communication between workgroups and workflow and offering multilingual facilities (Miller 98). The VENTO platform is based on conventional, client-server, communication interactions. The business processes are specified using a business process definition language related to the workflow management system, while the execution of them is performed internally to each business domain and in a centralised way

(Grefen 99). The workflow management system actually supports not the execution and management of shared business processes, but actually co-ordinates the execution of groupware services and the sharing of documents (Georgakopoulos 98). The coordination and management of shared groupware processes is performed by the exchange of proprietary messages based on TCI/IP protocol. The project deals directly with closed and well-integrated group of companies, i.e. static VEs that have static business relationships and tight coupling business processes. Additionally, the project does not specify any generic mechanism for inter-domain business process execution (Grefen 99). Finally, the project does not put emphasis on the dynamic selection of partners. On the contrary, the partners participated in the VE constitute a closed group of co-operating partners (Wognum 99).

The GENIAL project, Global Engineering Network (GEN) Intelligent Access Libraries, aims in the establishment of a Common Semantic Infrastructure (CSI) (GENIAL 98). The CSI infrastructure enables enterprises from different business sectors to combine internal knowledge with engineering knowledge accessed on-line and world-wide via GEN services. The GENIAL platform consists of a framework for the systematisation of engineering knowledge, i.e. a generic software for the access, insertion, and administration of distributed engineering information and knowledge, and an electronic marketplace, that enables different companies to locate partners and establish co-operation with them. The project addresses only the establishment phase of VEs, i.e. the selection of partners and thus, no business process specification or execution mechanisms are provided (Fielding 98). The approach of the project is related to the sharing of information, e.g. industrial designs and modules, among different business domains. In general, it can be considered as a virtual marketplace or industry specific portal system for industrial modules and designs accessed by different industrial companies (Hunt 99). However, these domains neither share processes nor co-operate among each other, i.e. they do not explicitly constitute a virtual enterprise (Fielding 98). The execution and management of business processes is considered out of the scope of the project.

The VIVE project, Virtual Vertical Enterprises, aims at developing a general methodology that enables SMEs to exploit the opportunities of higher competitiveness offered by co-operative technologies (VIVE 98). The VIVE concept and its implementation is based on the development of robust methods for selecting and adapting information and communication technology solutions to enable the operation of such distributed business ventures and on the creation of a new entity, the "Business Integrator". This new entity is capable of identifying market opportunities, specifying the required business process, and integrating the enterprise integration infrastructure in terms of communication and information. The VIVE concept leads to the static VE case where the integration of shared business processes is pre-determined and fixed, while the execution and management of shared business processes is achieved in a centralised way, i.e. the Business Integrator (Georgakopoulos 98, Zarli 99). The VIVE concept neither provides any means of dynamic partner selection and negotiation nor loosely coupled business process execution (Fielding 98). The Business Integrator is actually a centralised node that undertakes the responsibility to co-ordinate and manage the relationships, i.e. the shared business processes among the partners. The co-ordination mechanism is based on integration of distributed objects, i.e. the Business Integrator plays the role of the information broker (Grefen 99).

The TEAM project, Technologies Enabling Agile Manufacturing, provides the critical enabling technologies needed to implement agile manufacturing concepts (TEAM 97). The main goal of the project is to design, develop, and test globally defined manufacturing business processes. The shared manufacturing business processes are executed and managed by workflow management systems that co-ordinate their execution through the exchange of events, i.e. the

management of shared business processes is event-driven. The business domains taking part in the virtual enterprises have static and well-defined interfaces, while the integration among local and remote processes has been performed manually and in a static way (Camarinha-Matos 99). This project aims to deliver an event-driven workflow management system for the execution and management of static and pre-determined shared business processes. The events are generated by certain business entities within one domain and are forwarded to the corresponding partners. The exchange of events is performed by using a general-purpose domain gateway, which receives and forwards specialised events to other domains. The gateway on the receiving domain analyses the event and starts the corresponding process that will handle it. In such a way the co-ordination and management of shared processes is performed (Zarli 99). The project does not deploy any dynamic concepts for the selection of partners and automated negotiation (Filos 00). Additionally, the project does not use any type of virtual marketplace where different VE partners can register process offerings (Spinosa 98).

Another project, related to some extent to the area of VEs, is CrossFlow (Crossflow 99). The main aim of the CrossFlow project is to provide a mechanism for cross-organisational workflow management system without explicitly mentioning VEs (Hoffner 98). The key technical objectives of the project are to develop a detailed architecture that addresses the open issue involved in cross-organisational workflow, to develop an integration tool for setting up the link between the different workflow management systems of the co-operating organisations and "harmonizing" semantically and syntactically the shared business processes, and monitoring of out-sourced processes to regarding progress and resources consumed (Hoffner 99). The project deals with both, the selection of partners, and the execution and management of shared business processes across-domains. For the selection of VE partners, the project deployed the standard OMG-Trader (OMG 98) as the key matchmaking mechanism. The VE candidate partners that would like to offer services to other domains register their offerings in the Trader. VE partners that would like to use specific business processes provided by other partners, conduct the OMG-Trader using CORBA-IIOP (OMG 98), and get a list of potential partners that can provide the service. The selection of the most appropriate partner is done by human-driven negotiation process (Hoffner 98 and 99). After the partner has been selected, the integration process starts. This process actually involves significant manual steps that both partners should take in order to "adjust and harmonise" their business processes from both syntactically and semantically point of view. As soon as the integration process finishes, management of shared processes can be done. The internal business processes within each domain are executed and controlled by a proprietary, workflow management system. Cross-organisational business process execution is done through specialised gateways that have been previously configured. The communication mechanism among these gateways is based on a well-defined CORBA-IIOP interface. The project does not provide any generic mechanism for business process definition for cross-domain business process execution (Zarli 99, Filos 00). Additionally, the result of the business process integration is a set of closed domains that share business processes. The evolution of the VE is rather impossible and the alteration of one domain requires the re-integration of the whole network. It is clear that the CrosssFlow project aims at the establishment and provision phase of only static VE, though the partner selection process is semi-automated with the support of OMG-Trader. The project does not address dynamic aspects of virtual enterprises at all, but, on the contrary, put major emphasis on the integration phase for the syntactic and semantic "adjustment " of shared business processes.

Other projects that work towards the direction of cross-organisational workflow management systems without deploying directly mechanisms for dynamic negotiation and selection of

partners during process execution are MARIFLOW (MARIFLOW 99), ACE-Flow (ACE-FLOW 98) and WISE (WISE 98, Alonso 98). All these projects are trying to investigate ways for distributed execution of business processes across organisational boundaries without considering dynamic VE concepts, like virtual marketplaces and negotiation. In all projects the shared business process are pre-determined and well-defined while the domains that provide the different business processes have been pre-selected with or without automatic virtual marketplace mechanisms.

Similar ideas and concepts like CrossFlow have also the ACE-Flow project, Deploying Agile Customer-Supplier Chain and Efficient Process Management with Federated Workflow Systems. The project aims to develop a solution for inter-organisational workflow management and has as its objective to support the automation of enterprises' business-to-business operations, i.e., specification and execution of global workflows in a workflow federation formed by the collection of distributed autonomous workflow systems (Miller 98). The specification of global workflows will allow the "import" of workflows offered by other parties of the federation; it will relay on an open database that maintain information about workflows that are offered/provided by some party of the federation. Secondly, middleware will be developed that is required in order to establish inter-operability among workflow systems of the federation in such a way that the operational workflow management systems are not required to be extended. However, the relationships between the "workflow provider" and "workflow consumers" are static and pre-defined in the centralised database of federated workflows (Grefen 99). The project has not provided a generic way for cross-organisational business process execution. Additionally, the project does not address dynamic aspects of VE, i.e. the deployment of a matchmaking service and the automated negotiation for the selection of partners on-demand and during business process execution (Hoffner 99).

The WISE project, Workflow based Internet SErvices, aims at designing, building, and testing a viable infrastructure for distributed workflow based applications over the Internet. Such infrastructure will include an Internet based workflow engine, acting as the underlying distributed operating system, that controls the execution of distributed applications, a set of brokers enabling the interaction with already existing systems, that are to be used as building blocks, and tools for programming in the large to allow final users to configure and develop distributed applications. The project aims to solve the limitations of current workflow systems and to extend their applicability to the Internet by providing a broker based platform for interacting with heterogeneous, stand-alone applications and implementing transactional mechanisms as a way to provide execution guarantees (Alonso 98). These solutions will be integrated into a robust, reliable, and scalable execution engine able to control the execution of distributed applications over the Internet in a distributed way. The WISE project mainly concentrates on the provision of cross-organisational business processes without deploying virtual marketplace mechanisms. The project follows a centralised approach where the Internet-based workflow engine plays the role of the co-ordinator and manager of the shared business processes. Extensive technical details regarding the coordination mechanisms for cross-organisational process execution are not directly provided. In general, the project addresses the area of cross-organisational workflow management systems but does not cover dynamic aspects like virtual marketplaces, dynamic selection of partners and automated negotiation (Georgakopoulos 98, Tombros 99).

The main objective of the WIDE project, Workflow on Intelligent Distributed database Environment (Grefen 99), is to extend the technology of distributed and active databases, in order to provide added value to advanced, application-oriented software products implementing

workflow techniques. Specifically, the main goals of WIDE are to define an advanced conceptual model for describing both, the flow of activities, and the organizational environment in which these activities are performed. Particular emphasis has been put on specifying exceptions in the normal flow of activities, and on supporting different types of exceptions and abnormal situations. Additionally, special mechanisms have been developed to provide flexible workflow management through advanced database systems including active database technology and advanced transaction management in a distributed environment with long running transactions (Grefen 98). WIDE is inspired by a coherent, component-oriented vision; modern software systems will be built by composing, enhancing, and integrating software components. Thus, flexible and extensible active rules and enhanced transactional models will be developed on top of existing database kernels, with a kernel-independent approach that warrants maximum portability and inter-operability. In particular, compliance towards the CORBA standard will be enforced. From a technical standpoint, WIDE will provide tightly integrated features concerned with advanced transactions, by supporting distributed and asynchronous processing in the context of long-running and co-operative activities, and with reactive processing, by supporting a rich event language, as well as, enhanced, flexible coupling to transactions (Grefen 99). The WIDE project has a very clear focus towards distributed, intra-domain workflow management systems and deployment and integration of conventional distributed components. The project does not address, as such, the area of inter-domain workflow management and dynamic selection of partners.

Another project, related to WIDE, is TRAMS, Transactions and Active Database Mechanisms for Workflow Management (TRAMS 98). The aim of TRAMS project is to develop a workflow management system supporting the modelling and enactment of business processes. The project interprets as process a timely or logically ordered sequence of activities. The project distinguishes between two types of activities namely, the manual and the automatic ones (Geppert 98, Tombros 99). The activities are carried out by humans, possibly supported by software tools, while the automatic one are carried out by software systems without human intervention. In TRAMs, the main focus is on the modelling of workflows. The objective of the project is to come up with a holistic approach that integrates all relevant aspects of process modelling, like modelling the process itself, including its structure and constraints like ordering constraints, data dependencies, and time dependencies, modelling the behaviour of the participating entities, modelling the services offered by and within the environment, and modelling related and required transactional properties of workflows and activities, i.e. agent-specific semantic concurrency control, recoverability and compensation, etc. A second focus of TRAMS project is on enactment of workflows using advanced database technology, i.e. the controlled execution of workflow specifications (Geppert 98, Tombros 00). Currently, the project uses only the broker/services architecture model for designing the software architecture of process-oriented environments. This model in turn is implemented on top of an object-oriented database management system. Furthermore, the third area where the project will put emphasis is on advanced database technology for WFMSs. TRAMS, like WIDE, is trying to improve the functionality of workflow management systems and business process definition languages for intra-domain purposes and applications with the integration of distributed business object concepts and technologies. The project does not address neither, cross-organisational workflow business process execution, nor dynamic selection and negotiation of workflow providers (Zarli 99, Ouzounis 99b).

The objective of the C3DS project, Control and Coordination of Complex Distributed Services, is to exploit distributed object technology in order to create a framework for complex service

provisioning (C3DS 99). By complex service provisioning the project primarily mean the ability to compose a given service out of existing ones, as well as, the ability to exercise dynamic control over the execution of the service. Mechanisms will be needed to dynamically add, extend, remove, or move component services in a dependable and predictable manner. At the same time, end users, most of whom will not be programmers, must be able to specify, create, configure and manage services easily. The C3DS approach to building a framework for complex service provisioning, unlike other approaches, will be based on unifying three technologies: software architecture based development environments, software agents and transactional workflow management systems. According to the C3DS project, an agent may be defined as a software entity, e.g. a process, an active object, that performs operations on behalf of a user or another software entity in order to achieve an assigned goal. Workflows are rule based management software that direct, coordinate, and monitor execution of multiple tasks arranged to form complex organisational functions. Workflow management systems are ideally suited to meeting service level requirements. Software architecture specifications, expressed in a high-level Architecture Description Language (ADL) describe the structure of the components of a software system, their interrelationships, principles, and guidelines governing their design and evolution. The C3DS will achieve its objective by developing ADL-based tools and techniques for the specification of the software architecture of complex services and for the specification and usage of services through combination of components and integrating agent and workflow technologies for the development of a distributed Task Control and Coordination Service (TCCS) platform that will provide the basic infrastructure for the deployment of software agents and control and coordination of service provisioning activities. The features of the ADL will be expressive enough to permit descriptions of inter-task dependencies and coordination as expressed in workflow scripts and agent programs thereby providing a unified way to build agent and workflow based systems. To achieve interoperability in a heterogeneous environment, the TCCS platform will make use of the object request broker (ORB) middleware and CORBA services, such as, object transaction service, to support a novel dependable workflow execution environment using flexible transactions composed of transactional and non-transactional activities. The main result will be the C3DS framework, software toolkit for specifying, controlling, and coordinating complex, distributed services that will be easy to manage and customise. Whilst the Framework will be the most identifiable way of demonstrating the effectiveness of the project results, the concepts and techniques that underpin these results will be of generic value, capable of being incorporated in proprietary systems. The C3DS project's objectives lie in the area of intra-domain distributed applications and automation of processes by deploying distributed middleware technologies (Filos 00, Camarinha-Matos 99). Therefore, the project addresses slightly cross-organisational workflow management issues or dynamic selection and negotiation of VE partners.

The EvE project, EVent Engine, has as its objective to investigate event-driven workflow execution (EvE 98). In EvE, every interesting situation is expressed as a possibly, complex event and the operations of all the components in EvE, including processing entities, are defined by generating and reacting to event occurrences. For that matter, EvE combines the technology of active database management systems and event-based systems, e.g., event-based software architectures. The major, but not exclusive, purpose of EvE is to provide a runtime system for the Broker/Services Model, which is well-suited to define the software architecture of workflow systems and cooperative process-oriented environments. EvE has a multi-server architecture, where each server typically serves all processing entities in a local-area network (Geppert 98). Distributed workflow enactment is accomplished through multiple EvE-servers connected by a wide-area network. The main objective of EvE project lies in the area of intra-domain workflow

management systems. The co-ordination and management of business processes is done by the exchange of events (Tombros 99). The project emphasis on loosely coupled, distributed execution of processes but does not focus on inter-domain process management. Additionally, the project does not propose any type of cross-organisational mechanism not dynamic selection of partners (Geppert 99).

The MOBILE project takes a general and application independent approach to workflow management and covers aspects reaching from business process modelling to the implementation prototype of a high performance, reliable, distributed workflow management system (MOBILE 98). The basis of the project is the Mobile workflow model. The main characteristic of this model is to perceive a workflow as a collection of independent perspectives, hence Mobile is a perspective oriented workflow model. The overall project can be divided into several subjects, namely integration of business process modelling and workflow management, mobile workflow model and language, ad hoc workflows and dialogs, application integration based on transactional and non-transactional base services, like CORBA, Encina, Remote Procedure Calls (RPC), and architecture development for scalable, reliable, and distributed system design of the Mobile WfMS prototype. MOBILE project is directly related to large-scale workflow management systems for intra-domain purposes and deploys middleware services and emerging standards for workflow management systems, like OMG's-JointFlow (OMG 98). Though the approach of the project towards the integration of business processes with workflow management systems is interesting, the project as such does not propose or covers the area of cross-organisational workflow management (Zarli 99, Filos 00).

The ProcessLink project conducted in Stanford University is developing an agent-based framework consisting of generic agents and a message protocol for integrating multidisciplinary engineering software and managing distributed design projects (ProcessLink 98). This framework allows to "wrap" legacy software with backend code that will disturb the existing software interface, as little as possible, while providing useful co-ordination functions. The main emphasis is on open process management. This differs from workflow and process re-engineering because a distributed collaborative process does not impose a process definition on it. Though the project did not provide a generic business process definition language and a workflow management system, it can offer a set of coordination mechanism using autonomous agents. The project uses a "weak" agent approach, in which every agent is first wrapped with specialised software entities, like components, and becomes ready to send and receive messages corresponding to interaction semantics (ProcessLink 98). However, the agents do not necessarily have to be "smart" or conform to any particular theory of agent construction and language. The only commitment is to send and receive messages conforming to a defined set of interactions, protocols, and ontologies. The ProcessLink project takes a very simplistic view of distributed, co-operated agents towards process automation and execution. The major emphasis is on coordination mechanisms and integration of agents with legacy systems. Finally, the project does not address neither multi-domain process execution nor dynamic selection and negotiation of task providers (Cost 98, Ciacarini 98).

The MIAMI project, Mobile Intelligent Agents for Managing the Information Infrastructure, is one of the first projects worldwide dealing directly with standard compliant intelligent mobile agents and virtual enterprises (ACTS-MIAMI 98). The major objective of MIAMI is to develop a complete framework for the establishment and management of virtual enterprises based on intelligent mobile agents based on a unified OMG-MASIF (OMG 98) and FIPA (FIPA 98 and 99) compliant mobile agent platform. MIAMI introduced the concept of virtual marketplaces for the selection and negotiation among VE partners in order to enable dynamic VEs concepts. The

specification of business processes is done based on an open, state of the art, business process definition language specified in XML, while the execution of the shared business processes is done by an agent-based workflow management system. Specialised workflow agents, located in different business domains, co-operatively and in a distributed manner execute and control different instances of the shared VE business process by conducting the virtual marketplace for the selection and negotiation of VE partners on-demand and during business process execution (Ouzounis 00a). Additionally, MIAMI introduced special agent-based mechanisms for the management of the VE network layer. The network links between the domains are controlled and managed by a new third party network provider, the Active Virtual Pipe, that monitors the network and takes certain actions when the performance of the network connections is not the adequate one. MIAMI was one of the first projects that introduced, developed, validated, and demonstrated dynamic virtual enterprise concepts based on standard unified mobile agent platforms (Ouzounis 00b). MIAMI is considered as one of the most influential projects in the area of dynamic virtual enterprises (Filos 00). The work proposed in this thesis is directly related to the MIAMI project. More specifically, the MIAMI project fully adopted, developed, and successfully demonstrated the proposed concepts.

Finally, the EURESCOM P815 Project, aimed in the development of an open, distributed, adaptable agent-based workflow management system for cross-organisational business domains (P815 Project 98). The main contribution and innovation of the P815 project is the development of an inter-domain ontology for cross-organisational business process execution specified in XML (XML 98, Harold 98), an open and adaptable XML-based business process definition language for the specification of business processes, an intelligent workflow engine for the execution and management of distributed business processes, and a set of specialised workflow intelligent agents that execute, manage, control and co-ordinate shared business processes in co-operatively and distributed manner (Ouzounis 99b). The project did not directly address dynamic VE concepts like virtual marketplaces and dynamic selection of partners, however provides a very good conception regarding the execution of shared business processes provided by different business domains. The key contribution of the project is the open and adaptable approach towards cross-organisational execution and management of business processes based on standard mobile agent concepts like OMG-MASIF and FIPA. Actually, P815 project significantly contributed to the standardisation committees towards the specification of open ontologies for cross-organisational agent-based workflow management systems. The work proposed in this thesis is directly related to the P815 project. More specifically, the P815 project fully adopted, developed, and successfully demonstrated the proposed concepts related to autonomous, distributed, inter-domain execution and management of business processes.

Virtual Enterprises is a rather new technology research area where a rapidly increasing number of projects and R&D activities are starting to consider it (Zarli 99, Ouzounis 98e, Malone 91, Georgakopoulos 98). From the above description and analysis of the most influential projects in the area of Virtual Enterprises, certain conclusions can be drawn:

- Most of the emerging R&D projects and scientific activities have different conception, definition and interpretation of the term virtual enterprise. A clear definition and distinction of the VE model in comparison with supply chain management, virtual organisation and extended enterprise concept is still missing. Most of the projects did not even consider the major distinction among static and dynamic VEs,
- Most of the projects are analysing, designing, and developing solutions for static VEs, i.e for pre-defined number of partners with fixed business process interfaces among them and

static proprietary co-ordination mechanisms,

- Most of the projects have as a selected business sector the manufacturing area where the co-ordination and management of processes is tight coupling, while the customisation and integration of shared business processes is static, manual and pre-defined. Therefore, the business relationships among the partners are rather medium to long term and consequently, the static VE model is more suitable,
- Most of the projects use Electronic Document Interchange (EDI) as a preferred solution for cross-organisation business process execution. However, EDI is restricted only in the area of simple electronic commerce business processes and has certain drawbacks (see analysis below). A more generic, flexible, and adaptive mechanism for cross-organisational business process execution and management is required,
- Some of the projects, in order to overcome the problems introduced by EDI, deployed distributed component-based technologies, like business objects and components. These technologies impose tightly coupling mechanisms among the distributed inter-domain components and thus, produce solutions for static VEs (see analysis below). Although some projects, like Crossflow, introduced matchmaking approaches for the semi-automatic selection of VE partners, still the execution and management of shared business processes is achieved through static interfaces among the domains. In general, business object concepts have been proposed and deployed for intra-domain distributed application and are inadequate for cross-organisational business process execution,
- In order to solve the problem of tight coupling among domains, specialised messaging systems have been recently introduced. The execution of shared processes across domains is done though the exchange of specialised messages usually specified in XML. A new generation of XML-based messaging systems and protocols has been evolved, like BizTalk (Biztalk 98), CXML (CXML 99), etc. that try to provide solutions for dynamic and loosely coupled inter-domain business process execution and management (see analysis below). These solutions are restricted only in the area of electronic commerce, as EDI did, and thus, can not be used for any type of business process management,
- Recently, some projects started investigating the deployment of workflow management systems for cross-organisational business process execution and management. Although most of the projects are not directly related with dynamic virtual enterprise concepts, they aim to provide a complete and generic framework towards inter-domain business process management. The basic coordination mechanism used is CORBA based communication among specialised gateways, e.g. Crossflow, that in the sequel deploy internal workflow management system. Very recently message-based approaches have been also proposed for the co-ordination and execution of processes based on Internet based transport protocols like the SWAP protocol (SWAP 98, Bolcer 99). Workflow management standardisation organisations, realising the benefits and potential of XML-based, message-based, cross-organisational workflow execution and management, started to work towards standard interfaces like the Workflow Management Coalition (WfMC) (see below analysis).
- Finally, the most emerging and new concept towards the management of dynamic VEs is the intelligent mobile agent approach. Agents feature some very important attributes, like autonomy, adaptability, distribution, mobility, and intelligence and are best suited to solve certain problems in this area. Especially, the execution and management of business processes through autonomous intelligent agents that have workflow management

capabilities and co-operate among each other is a very interesting and promising approach since it combines the benefits of workflow management systems, the benefits of agents, the deployment of open and flexible ontologies and the interoperation and integration of conventional distributed object oriented technologies. This research area is considered very new and a lot of issues remain open and under investigation (see analysis below).

In the next section an exhaustive description, analysis, and comparison of the above stated influential technologies and concepts deployed in the area of virtual enterprises is presented.

3.2 Technologies and Standards for Virtual Enterprises

The development of virtual enterprise concepts, models, and technologies has been based on different, emerging technologies. The most influential ones where:

- Electronic Document Interchange (EDI)
- Distributed Component based Business Systems (DCBS)
- Messaging Systems (MS)
- Intelligent Mobile Agents (IMA)
- Workflow Management Systems (WfMS)
- Virtual marketplaces, partner selection, and automated negotiation (VMP)

In the following section the above-mentioned technologies and emerging standards related to these technologies are analytically discussed and compared for their suitability in the dynamic VE concept.

3.2.1 Electronic Document Interchange

Electronic Document Interchange (EDI) was the first approach to be widely adopted for inter-domain business process management. Many implementations of EDI have shown impressive returns, although EDI deployment has required sufficient high levels of investment and integration work to limit it to only the largest enterprises. In recent years, EDI deployment costs have dropped and enabled more organisations to take advantage of it, but EDI implementations still tend to be expensive and require significant integration and customisation work with the back-end information systems (Gibon 99, Ouzounis 98d).

The development of EDI was motivated by the realisation that simple cross-organisation business processes such as purchasing, shipment tracking, and inventory queries were tremendously inefficient (Lomet 93). Therefore, EDI focused initially on producing electronic versions of traditional business documents, such as purchase orders and invoices, and then enabling automated processing and transmission of those documents.

In a typical EDI application to support purchasing, an EDI system is integrated with the existing purchasing system at the buying company. When a buyer enters a new purchase request into the purchasing system, a corresponding request is sent to the EDI system. The EDI system then constructs an electronic purchase order document and transmits that to the selling company. Originally, EDI transactions were all sent over dedicated communications channels, which meant that such channels had to be set up between any pair of organisations wishing to use EDI

between themselves. To alleviate this bottleneck, 3rd party organisations have emerged offering Value Added Networks, or VANs. These VANs take care of the transmission details between subscribers. Thus, a company can subscribe to a single VAN and require all its partners to subscribe to that VAN. In this way, the company does not need to set up dedicated networking connections to each of its partners. Currently, work is underway to enable the delivery of EDI transactions over the Internet.

When the purchase order is received at the selling company, over a dedicated connection, via a VAN, or via the Internet, it is processed by the receiver's EDI system. This system transforms the message as required and inputs it into the receiver's enterprise system. Once in that system, the new order is handled just as any other order would be.

This process is illustrated in Figure 2, EDI Architecture, below.

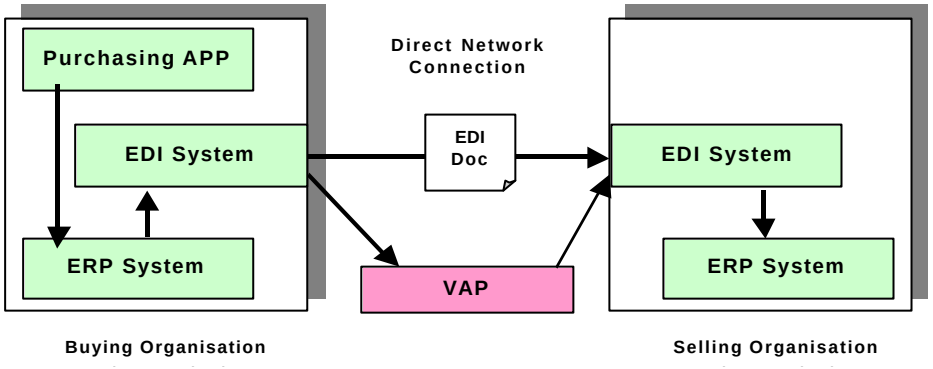

Figure 2: EDI Architecture

This example illustrates the source for the costs of an EDI implementation. First, the two partner organisations need to agree on the exact formats of the documents exchanged between them. The EDI standards provide initial definitions of common business documents, but historically, these have been inadequate for actual use. Instead, each EDI deployment has involved manual negotiations on and agreement to a set of Implementation Conventions describing the extensions to the standard documents and the actual formats that will be exchanged. This negotiation and agreement process represents a significant cost for EDI deployments (Lee 98).

To address this issue, the EDI standard organisations, EDIFACT (EDIFACT 98) and ANSI X.12, have undertaken an effort to standardise sets of documents for various industries. For example, ANSI X.12 has recently released a set of standard EDI document definitions for the health care industry. Using these industry standard document definitions, the customisation required per relationship can be reduced, although in general, per-relationship integration and customisation work is still required.

Due to the fact that EDI is based on document interchange, one significant integration cost is avoided. The EDI and enterprise systems at the two partner organisations do not need to directly reference each other. Instead all interactions are accomplished via document exchange. But, because the set of documents supported by EDI is relatively limited and extending this set is expensive, it is difficult to use EDI as the basis for a closely coupled relationship (Billington 94, Christofer 93). EDI transactions, as currently defined, simply don't support a rich enough set of

possible business interactions. Current work in EDI is addressing this issue, extending EDI to support more fine-grained and transactional interactions.

Given the set of tradeoffs involved in the use of EDI, it is best suited for long-term and stable business relationships between organisations that can make significant investments in mechanics to support their relationship. Work that requires tight coupling and co-ordination, such as supply chain optimisation, or product design, is best done outside the EDI context (Srinivasan 93). Straightforward business transactions, such as purchase orders, can be well supported using EDI. In general, each new EDI relationship requires new customisation and integration work. These relationships are thus not entered into lightly and return on EDI investment is gained over long periods of time, not over short term transactions (Bolcer 99, Doz 98).

3.2.2 Distributed Component-based Business Systems

Whereas EDI supports electronic business by automating existing processes and enabling electronic document exchange between separate organisations, a number of other systems approach electronic business by trying to create a single virtual organisation (Stricker 00, Fielding 98). These systems use middleware, a layer of integration code and functionality that allows multiple diverse and distributed systems to be used as though they were a single system. Using these middleware tools, business applications can be transparently accessed the multiple backend systems (Georgakopoulos 98).

The first approach used in developing such enterprise systems was proprietary custom-engineered solutions on top of Internet's TCP/IP protocol. These systems were traditional client/server applications with proprietary message formats and customised integration of third party purchasing management systems. The provided solutions were in general closed and tailored to company's needs and requirements. The development time and the cost were rather high, while the maintenance and re-engineering was also difficult and ineffective (Orfali 96, Nissen 99). These solutions adopted mostly by big companies due to the high development and integration costs and are characterised by inflexibility, limited degree of interoperability and security. The main reason of low acceptance of such solutions was the high re-engineering time. This means that as the business processes and activities of the company were changing in order to respond to market needs, the systems could not respond to these changes effectively (Bolcer 99).

Due to the rapid development and acceptance of distributed object oriented platforms, a new generation of enterprise systems started appearing. The concept of re-usability and middleware has been introduced in the development and integration phase. These concepts resulted in the creation of Distributed Component-based Business Systems (DCBS). The DCBS composed of basic building blocks or components, mostly based on object-oriented technologies, that can be bought "off the self", reused or extended, customised, configured, and integrated into the overall, distributed business information system (Orfali 96, Doz 99). This approach enables the development of solutions faster, in a cost-effective way, with easy maintenance and accepted level of interoperability and distribution. These technologies are actually integrated development and run time environments that isolate much of the conceptual and technical complexity involved in building business applications. Due to the advent of Java and open distributed platforms, like CORBA, object-oriented middleware business systems gain strong momentum and support (OMG 98, EJB 98, Ouzounis 98c). However, these systems have been initially designed and developed for intra-domain distributed applications and not for dynamic inter-

domain business process execution (Carr 96). Even when they are used for inter-domain business process execution and management, their goal is to create a single unified view of a virtual enterprise (Camarinha-Matos 99).

Classic middleware systems typically involve tight binding between the systems and processes at the various domains. By closely coupling the different domains, classic middleware systems are able to provide rich functionality, but require expensive initial development and deployments, pre-agreement in the interfaces used by the different components and carefully coordinated ongoing deployment management (Thompson 99, Sheth 98). These systems are thus most appropriate for use in intra-domain, distributed applications or long-term and closely co-ordinated business partnerships (Redlich 98, Hull 99).

In the following sections the most influential DCBS frameworks are discussed and analysed in relation to their applicability to the dynamic VE concept.

3.2.2.1 OMG's Business Objects

OMG's Business Object was an industrial activity towards standardised DCBS based on a set of well-defined middleware CORBA services (OMG 98). The activity started mid 97 by OMG members and failed to produce a common framework due to technical and political differences among the members.

The main idea behind the proposed framework was the business object, i.e. "specialised CORBA objects that they are network accessible through an object request broker. A CORBA "object reference" uniquely identifies an active business object within the distributed object environment for purposes of communication through an object request broker." A business object also has a unique identity that associates it with the entity it represents in the business domain. This identity, or key, uniquely identifies a business object within its type and is always associated with a corresponding entity in the real business world.

The main attributes and characteristics of business objects are the following:

- **transactional**: due to the fact that business objects are sharable in a distributed, multi-user, transactional environment, there must be concurrency control and transaction serialisation to maintain the integrity of the model they represent,
- **persistent**: persistence is necessary to maintain the state of objects, i.e., the data, when the system is shut down or fails. Persistence is not required for all business objects, but if they have state and represent current information about the business, they will be persistent,
- **relationships**: business objects will have relationships with other business objects that represent associations between their business-world counterparts. Relationships may be one-to-one or one-to-many, and they may be bi-directional or one-way,
- **ad-hoc Notification**: Most business objects also support ad hoc event notification. A message can be sent to a business object requesting notification of certain events be sent to a designated consumer. Notices will be sent whenever any specified event occurs until the request is terminated.

Two type of business object have been identified:

- **common business objects**: objects that represent the key elements of a business domain, like an employee. These objects are persistent and are used for building high level business objects.
- **business process objects**: objects that perform a business operation or a process within a business domain or context. Usually, these objects are not persistent and utilise existing common business objects.

The layered architecture of the Business Object concept is depicted in the following figure:

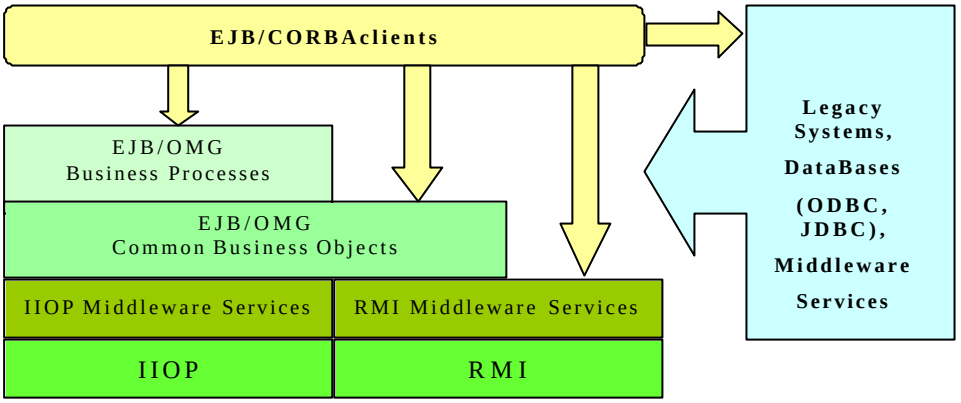

Figure 3: Distributed Component-Based Business Framework Architecture

The Business Objects were one of the first activities in the area of standardised DCBS and introduced a set of innovative ideas. One of the main benefits was the shorten development and deployment costs, as well as, lower costs in integration and distribution (Orfali 96). However, the differences among the OMG members did not allow a concrete, stable, and well-defined framework for business objects.

3.2.2.2 Enterprise Java Beans

Enterprise Java Beans (EJB) was a competitive to the OMGs Business Objects proposal from Sun Microsystems that has been proposed in late 98 (EJB 98). The main goal of EJB was to propose an architecture for building distributed object-oriented business applications in Java by combining, not only basic middleware services, but also existing business components. In that respect, EJBs, as well as OMG Business Objects, share the same goals and principles, i.e. reusability and fast development and integration.

An EJB runtime environment is composed of a server and a set of containers. The server is not an application server; instead, it routes method calls to the enterprise beans deployed under its containers and provides services to these containers and their components. The services provided are defined by electronic contracts between the various parts of the EJB architecture like the contract that exists between the container and the beans in it. These electronic contracts provide interfaces that decouple parts of the architecture into roles (Nissen 99).

The EJB specification defines a number of roles necessary in implementing the EJB component architecture. The roles are logical in nature, so multiple roles may in fact be performed by the same party or human operators. EJB defines the following roles:

- **server provider:** the server provider provides the EJB server, which handles distributed object, distributed transactions management, and other services for enterprise beans in containers,
- **container provider:** the container provider produces a container, which is the context that interfaces with enterprise beans at runtime. The container can implement the session bean contract or the entity bean contract,
- **enterprise bean provider:** the Enterprise bean provider writes enterprise beans to make up specific applications.
- **deployer:** the deployer takes beans produced by the enterprise bean provider and deploys them in the backend runtime environment. This process may involve mapping the security roles set by the beans to the security roles required by the organization,
- **application assembler:** the application assembler uses the client view contract of the enterprise beans deployed at the backend to assemble client applications. The application assembler may also produce new beans by combining existing beans.

Two key types of EJBs have been currently proposed, namely the:

- **session Bean**: a session enterprise bean models a connection or session with a single client. Session beans persist only for the life of the connection with the client. If the EJB server crashes, the session bean dies. When a new client references a session bean from the server, the container creates a new instance of the session bean, which is tied to the client that made the reference request through a bean object provided by the container,
- **entity Bean**: entity beans model business objects that need to persist beyond the life of a client instance. Each instance of an entity bean can be accessed simultaneously by multiple clients. Entity beans survive crashes of the server.

One of the main benefits for using EJBs is the simplicity and ease of integration, due to the usage of Java programming language and its accompanying services, web-integration, distribution (RMI), security etc (Ouzounis 98b). Another main benefit is the life-cycle operations for the creation and management of Beans. By implementing or extending only a set of well-defined interfaces and following certain instructions a bean can be easily deployed in different, native distributed platforms, like Java-RMI, CORBA-IIOP, and DCOM, due to the concept of containers. This approach enabled programmers to rapidly develop DCBS.

From the above description, it is clear that there are a lot of synergies among the OMGs Business Objects and EJBs. Particularly, the entity beans have similar concept to the common business objects while the session beans have similar ideas and mission like the process objects. In addition to that, the container model in EJBs is actually, the same concept of infobus provided by CORBA-IIOP for interoperability reasons. Actually, EJBs are considered not more than a well-specified, realistic, and standard version of OMG's Business Objects specification based on Java Framework and RMI protocol (Chung 98). Due to the industrial support of EJBs and the tools provided for developing and deploying this technology, EJBs are better positioned in the world of DCBS.

Although EJBs gained momentum in the open market, the model that is being used is the tight integration of distributed components, and thus the integration of EJBs business components across different domains can only be performed by tight integration and object binding (Hoffner 98 and 99). This is the favoured and applicable model for static Virtual Enterprises, where the

business relationships among the business partners are static, pre-determined, and fixed. This approach positions the EJB as a promising technology for intra-domain distributed component based business applications and not for dynamic VEs (Zarli 99, Tombros 00, Geppert 98).

3.2.2.3 San Francisco from IBM

IBM's SanFrancisco framework, proposed in mid 97, as OMG Business Objects and Sun's EJB, is a multi-tier Java development framework for building distributed business applications (SanFran 98). The SanFrancisco framework specifies and deploys standard business components and easy integration and customisation services with backend legacy systems. Using this framework, developers can build systems, such as order management systems, that span multiple physical nodes, integrate with the backend legacy systems at those nodes, and provide unified functionality across the diversity of different nodes and systems.

San Francisco was one of the first serious commercial attempts in the area of DCBS. The main design principles of San Francisco was the same like EJBs and business objects, i.e. to specify a set of common, persistent, objects that represent real entities in a business environment and a set of objects that deploy these "middleware" business objects to provide business processes. In that respect, San Francisco was actually IBM's view of DCBS based on the different concept emerged in OMG.

Although the framework was based on open, standard, distributed technologies like CORBA and Java, the acceptance of it was rather low (Zarli 99, Filos 00). The main reason was the deployment of non-standard and in most cases, proprietary middleware services provided only by IBM. The evolution and thus, maturity of San Francisco framework stopped when IBM announced full support of Java 2 Framework and consequently, the full adoption of EJB approach and concept.

3.2.2.4 Alliance from Extricity Software

Another related framework for inter-domain business process management is Alliance, a suite of products, aimed at business-to-business process integration (Allinace 98). Similar to the other middleware systems, Alliance provides a distributed framework that can access multiple enterprise systems and supports unified and transparent access to those systems.

However, unlike Enterprise Java Beans or SanFrancisco, which focus on data integration between systems, Alliance focuses on process integration between distributed business partners. Alliance is thus, designed from the beginning to support inter-domain business process management.

A set of business partners deploys and integrates Alliance by determining the processes that they will engage in together. The partners use the Alliance process implementation environment to model their common processes. Each partner then uses additional Alliance tools to model its private processes as they support the shared processes. Separating the definition of shared and private processes allows partners to update their private processes independently. Coordination is only needed when changing the shared portion of a cross-organization process.

At the same time, as the inter-domain shared business processes are specified, partners model the data they exchange during those processes. Alliance uses a document exchange model,

which decreases coupling of the partner information systems. The documents exchanged are defined starting from templates provided with Alliance and using information modelling tools included with Alliance. This customisation is analogous to the implementation conventions required to implement EDI solutions.

Because Alliance uses document exchange between organizations, it avoids many of the security issues that limit the applicability of Enterprise Java Beans and SanFrancisco in inter-domain deployments (Filos 00, Bolcer 99). However, other design choices in Alliance increase deployment costs and make it most appropriate for long-term, stable, closed, and closely coupled business relationships, i.e. static VEs. For example, because cross-organization business processes must be modelled across all the participating partners and then implemented in concert, partners must coordinate manually their deployments (Stricker 00). And during deployment, significant customisation and integration work is required so the system can interoperate with each of the partners' enterprise systems. This means that each new relationship takes significant work to support and thus, evolution of the VE is very difficult and time consuming task.

Alliance is thus most applicable to static VE model where business partnerships function essentially as long term partnerships of a larger virtual enterprise. These virtual enterprises can afford the deployment cost of an Alliance solution and can look for returns over long periods of time.

3.2.2.5 Distributed Component based Business Systems in the context of VEs

Distributed Component based Business Systems gained momentum in the R&D activities, as well as, in commercial systems due to the simplicity, ease of integration and deployment, high degree of distribution, standard underlying distributed protocols, like CORBA-IIOP and RMI and middleware services. However, most of these systems are inadequate for usage in a dynamic VE environment (Filos 00, Zarli 99, Tombros 00).

DCBS assume a tight coupling model (Orfali 96). This applies both to the integration with backend legacy systems and to the client applications. Backend systems and clients integrate with the distributed framework using the APIs and object models exposed by the underlying levels of the architecture. While clients are insulated from the APIs of the backend systems, they are tightly bound to the provided APIs. This design choice has two implications (Ouzounis 99b). First, by using object binding as the interaction technique, as opposed to document exchange used in EDI, DCBS applications must be adopted at once by all participants in the cross-organization relationship. And upgrades to backend systems, the component framework, and the business application must be coordinated across all participants (Spinosa 98, Hull 99). Second, because of the tight binding, security issues are a major factor. Objects running in the business components-applications at one company must be able to communicate directly with objects running in the same component model, either EJB or San-Francisco at a partner company (Thompson 99, Sheth 98). For good reason, the IT staff is reluctant to allow object access across corporate firewalls. This poses a significant barrier to adoption in cross-organization environments.

Additionally, the DCBS frameworks do not provide a complete solution, but instead serve as the starting point for developers to build applications (Orfali 96, Carr 96). By building on the

framework, developers can more quickly complete applications and leverage the code in the framework that takes care of many of the mechanical details needed for a successful distributed application. This is ideal for corporate developers who are already accustomed to doing significant custom programming.

Finally, these choices make the DCBS frameworks most appropriate for deployment inside a single company that needs to link multiple distributed divisions or sites (Redlich 98). Such a company can plan for a unified deployment and can afford the integration and customisation work. A single company can deal with the security and firewall issues internally without opening potential security hazards to the outside world (Nissen 99). Indeed, as it is stated above, the majority of DCBS deployments are taking place inside single enterprises and for intra-domain applications.

3.2.3 Messaging Systems

In contrast to classic DCBS, which seek to closely bind the enterprise systems and processes of several organizations into a single closely coordinated virtual organization, inter-domain business process management and execution systems can be built using exchange of documents, usually described in XML, to bind together multiple organizations (Ouzounis 98e, Geppert98, Hoffner 98). Ideally, such an approach would combine the strengths of EDI with the rich interactions, integration, and distribution supported by classic DCBS.

In order to support exchange of messages among different business systems and components, a distributed messaging system is needed (Sheth 98). Messaging systems have been initially deployed for interoperability reasons and easy integration among distributed, intra-domain applications. However, due to the success of the distributed object-oriented systems, messaging systems gained initially low acceptance. Recently, the new, emerging concepts of dynamic Virtual Enterprises, loosely coupled business systems, and federated business to business trading systems, brought messaging systems into light and attention again (Redlich 98, Hull 99).

In general a messaging system usually provides the following features (Ouzounis 99b, Nissen 99):

- one-one and one-many exchange of messages among different, distributed entities or clients,
- persistent storage of messages in queues for reliable and fault tolerant communication,
- store and forward operations,
- synchronous and/or asynchronous message passing to different distributed entities or clients,
- execution and communication autonomy between the senders and receivers,
- support for several open standard transport protocols like IIOP, RMI, HTTP, and TCP/IP,
- interworking among different messaging systems.

The key elements of a message are the envelope and the content (Bolcer 99). The envelop contains information like the identity of the sender and receiver, the time sent, the transport and content protocol used, the mode of communication, like synchronous or asynchronous, etc. The

content of the message describes, in a well-formatted way following a common ontology, the information that the sender sends to the receiver and the type of request or reply it sends to him.

Most of the messaging systems provide their own specialised message description language. However, due to the advent of XML, a new category of messaging systems started to appear. These systems utilise the power of XML for the description of both the envelope and content. Activities towards this direction are the proposed CORBA 3 messaging service (OMG 98), the Java Messaging System (JMS 98), the successful MQSeries from IBM (IBMMQS 98), the messaging system from BEA (BEA 99), and others.

Messaging systems are needed only for the physical transmission of messages among domains and are thus, application independent. The content of the message, the business context that is related to, and the semantic meaning of it do not influence the behaviour and functionality of the messaging system. Only the envelope of the message is necessary for the successful transmission of the message to the corresponding receiver. The business applications are responsible for specifying the ontologies, the business context, the set of legitimate messages that will be exchanged, and the applications protocols. This means that certain content description ontologies and protocols are required for the description and specification of business relationships and interactions among different, distributed, inter-domain entities (Spinosa 98, Doz 98).

In the following sections, a set of new, emerging, content specification protocols for simple electronic commerce transactions are presented. These protocols are considered the first step towards globally specified ontologies for dynamic business relationships and dynamic virtual enterprises.

3.2.3.1 Web Interface Definition Language

The Web Interface Definition Language (WIDL) has been proposed by WebMethods Inc at end of 97. WebMethods started its developments after observing that many electronic commerce websites function essentially as web-based interfaces to company business services. The websites are intended for use by humans, rather than for automated access. But if a way could be found to automatically access the services behind the website, then the site itself could be used as the basis for automated electronic business.

WebMethods's approach is called the B2B Integration Server and Web Automation Server. Developers can examine the websites of the business's partners and create WIDL descriptions of the interfaces and business functionality available on those sites (Merrick 97). After creating those descriptions, the developers use the WIDL to produce client proxies and special rules. The client proxies are callable routines that applications can use to access the services available at the remote website. The rules work in concert with the client proxies and inform the server how to process application calls to the proxies.

The developers, then, create a client application that issues calls to the client proxies generated from the WIDL description. The proxies in turn send the requests on to the Web Automation Server. The server, using the rules generated from the WIDL description, makes the appropriate web requests directly to the website in order to implement the requested operation. In this way, remote websites become available from applications, rather than requiring a human to manually interact with the site.

While the Web Automation Server enables easy access to the functionality that partners expose on their websites, it does not enable more complicated and rich interactions (Walles 99). For example, if a website does not provide an easy way to check on inventory stock, there is no way to use the web automation server to do such a check. However, because the server relied on document exchange between partners, it avoids many of the security issues and lets partners work in a loosely coupled fashion (Merrick 97). Partner system upgrades and process changes do not need to be coordinated and managed together. In fact, a business can create an application that accesses another business's services without needing any support from that business (Thompson 99).

The next step was to address the functionality limitations of the web automation server while building on the document exchange concept (Walles 99). To do this, the WIDL concept has been extended so that it could describe any available business service. Organizations can now write WIDL descriptions of the business services they provide, describing how a service is accessed, what kind of data it returns, and what parameters are needed to access the service

Despite the fact that the system uses document exchange to integrate the partner organizations, this exchange is essentially carrying object level API calls wrapped inside XML documents (Doz 98, Wognum 99). Clients at the originating site are programmed to an API generated from the definitions in the WIDL at the receiving site. And servers at the receiving site are integrated with a server API generated from the same WIDL. By using XML documents to pass information between organizations, the proposed approach avoids the security issues associated with DCBS systems. But, because the two business partners are tied together at an API level, the system does not avoid the integration and deployment costs associated with object level electronic business systems. In essence, the WIDL approach has enabled the use of XML document interchange as the distributed object system by deploying http as a transport protocol and not CORBA-IIOP or RMI (Stricker 99).

Because WIDL approach addressed the security issues and lets organizations partner in a loosely coupled manner, the system is appropriate to use in inter-domain business process management (Veloso 98, Wood 99). WIDL approach is wrapping object level API calls inside XML documents, and thus, the system can support much richer interactions than a conventional EDI-based system. At the same time, these rich capabilities mean that partner organizations must coordinate system deployment and evolution (Ciacarini 98).

Finally, WIDL remains a proprietary technology foundation (Walles 98). The founder of the approach, WebMethods, has submitted WIDL to the World Wide Web Consortium (W3C) for standardization, but the W3C has undertaken no certain activity related to WIDL (Merrick 97). Additionally, there are no other commercial solutions available from third party vendors besides the WebMethods that make use of WIDL concept. The WIDL approach is thus, appropriate for the same kinds of relationships as EDI, i.e. long-term, coordinated, closed and fixed business partnerships (Spinosa 98). It offers more extensibility and potentially lower integration costs than EDI but it lacks the business process oriented focus of Alliance (Wood 99). However, it is not appropriate for supporting more dynamic business relationships.

3.2.3.2 Common Business Library

Commerce One has taken a fundamentally different approach from EDI or the DCBS with its Common Business Library (CBL) proposed in mid 98 (CBL 98). CBL was originally developed by the non-for-profit standardisation organisation CommerceNet (CommerceNet 98). CBL uses

document exchange as the inter-domain interaction mechanism, like EDI. But instead of producing a comprehensive set of complete documents, as EDI does, or wrapping business service calls in documents, as WIDL does, CBL defines a set of basic building blocks specified in XML (Bolcer 99). These building blocks are then pulled together to make the actual documents describing the interactions between two organizations.

CBL is an application of XML. The building blocks are XML fragments and are then assembled to create complete XML documents representing an interaction, such as a purchase order, a shipping status inquiry, or an inventory stock query. The building blocks include constructs like catalogue entries, descriptions of business processes, terms of shipment and payment, etc. Where possible, the building blocks take advantage of other standards using, for example, the relevant ISO standards for dates, currencies, and names (CBL 98).

To use CBL, an organization starts by creating a CBL document describing its offer and its terms for doing business, i.e. its internal business processes and a set of interfaces for deploying these services in terms of CBL documents (Harold 98). These documents are then made available on the organization's website. Similar to WIDL, the documents describe the kinds of messages that the organization expects to receive and potential to reply with. However, these messages represent requests and responses for certain business processes, rather than encapsulating object level API calls to the services like WIDL does. This provides an additional layer of independence from the underlying services, allowing the organization to change and update its backend legacy systems without having to change the set of requests and responses that it supports (Hunt 99). After describing its offer and business processes in terms of CBL documents, the organization needs to integrate a CBL system with its backend system. This involves writing custom code that processes the expected CBL requests and makes the appropriate calls to the backend legacy systems.

Users, at other organizations, can now access the service descriptions available on the public website. This description has enough information that those users can select from among the available processes and begin sending requests to them. The requests are made by constructing appropriate CBL documents using the specified CBL building blocks, and then sending them over the Internet to the receiving organizations. The receiver's CBL system accepts the request, decomposes it, based on the contained building blocks, and processes the request. The requested domain can then return another CBL document describing when the request will be processed, how product will be shipped, etc.

CBL shares several characteristics with EDI and WIDL. Like those approaches, CBL uses document exchange as the interaction mechanism for business partners. This greatly reduces some of the integration and security costs (Veloso 98). However, document exchange can limit the kinds of interactions supported between partners (Wood 99). WIDL addresses this issue by reintroducing the tight-coupling costs associated with object-level interactions. Additionally, CBL takes a different approach to supporting rich interactions. Any CBL document is created from assembling a set of basic building blocks. These blocks are provided from the CBL framework. This gives organizations two ways to support rich interactions with their partners. First, by combining multiple building blocks, new types of documents not previously envisioned can be created. Second, because CBL is based on XML, the building blocks themselves can be extended in a safe way (Khare 99).

Since extension of CBL is straightforward, multiple competing vocabularies can be created and experimented with. Over time, the best vocabularies for various industries will be settled on and industry specific registries defining the vocabularies can be created. This lightweight and

evolutionary process will make the development of these vocabularies much easier than the corresponding EDI industry specific implementation conventions (Filos 00).

Industry specific electronic commerce protocols are similar to industry specific commerce vocabularies. CBL is also appropriate to use in defining and experimenting with these protocols. This reduces the amount of effort spent on mechanics during protocol definition and instead allows the protocol designers and developers to focus on the capabilities of the protocol (Ciacarini 98).

3.2.3.3 BizTalk Framework

The BizTalk Framework, in a similar way like WIDL and CBL, is designed to foster application integration and electronic commerce through data interchange standards based on XML (BizTalk 99). The BizTalk framework is a rather new initiative started in mid 99.

BizTalk assumes that application programs are distinct entities and application integration takes place using a loosely coupled, message-passing mechanism. There is no need for a common object model, programming language, network protocol, database, or operating system for two applications to exchange XML messages formatted using the BizTalk Framework. The two applications simply need to be able to format, transmit, receive, and consume a standardized XML message (Stricker 99).

Messages are the basis for the most significant contributions of the BizTalk Framework. A message flow between two or more applications is a means to integrate applications at the business-process level by defining a loosely coupled, request-response based, communication process. Since many business processes involve one party performing a service at the request of another party, the mapping of messages to requests is natural. Approaches making tighter integration demands, such as those based on special programming languages or shared distributed computing "platforms," are highly appropriate to tightly connected applications on single machines or in controlled environments, but they do not adequately support distributed, loosely coupled, extensible business process integration (Hamilton 98). An XML-based messaging system with open, extensible, wire formats captures the essentials of a business communication while allowing flexible internal implementations (Khare 99).

Until applications have native support for XML, these types of BizTalk Framework interchanges will require layered software that transforms native data types into XML and then performs the XML document routing. The BizTalk Framework will also provide support for schemas describing more complex interchanges involving multiple documents exchanged in a sequence. End-user companies have built these types of XML document transformers and routers in-house. Microsoft is developing a BizTalk Server that automates many of the functions required in a BizTalk Framework interchange.

3.2.3.4 Commerce XML

Commerce XML (CXML) is an industrial proposed standard for business to business systems proposed by Ariba Inc. with a set of industrial partners at early 99 (CXML 99). CXML, like WIDL, CBL and BizTalk, is an application of XML and specifies a set of messages for electronic commerce purposes. CXML shares the same design principles like the others but in a rather different way.

CXML specifies the envelope, as well as, the content of the message in XML. Both entities of the message are specified in XML. Transportation of CXML messages from one domain to another is done based on different transport protocols, like Http and TCP/IP, though the proposed CXML standard does not identify a particular one (CXML 99).

Two types of interactions between business domains identified, namely the request-response model and the one-way message. In the Request-Response, the requestor, that might be a business application, creates a legitimate[1] CXML message and sends it to another domain through the Internet. The receiver, upon request, translates the message, parses the content of the request, understands the context of the request, and invokes the corresponding back-end system or component. The back end-end system or component, that might be a DCBS, serves the request and delivers the results to the receiver. In the sequel, the receiver formulates the results in terms of a legitimate CXML message and sends it back to the original sender of the message.

In the one-way model, the sender sends a CXML message to a receiver by describing the type of message. The receiver, on the other business domain, parsers and understands the message, and invokes the corresponding service on a back end system. However, no response is generated and send back to the sender.

For the time being, only specific electronic commerce related request and response messages have been specified. In contrast to BizTalk, CXML proposed standard does not require any type of messaging system or specific transport protocol.

3.2.3.5 Messaging Systems in the context of VE

Messaging systems is an alternative technology option for dynamic VEs and inter-domain business process management. The main strengths of this approach is the differentiation among the specification of the services and the corresponding entities that provide these services (Redlich 98, Hull 99, Ouzounis 99a). This means that message based middleware systems are not based on the strong and tight integration of components, like in DCBS, and they do not require compatible middleware services, like EJBs (Spinosa 98). Additionally, messaging systems hide all the complexities of the underlying components or systems and enable true, loosely coupled, asynchronous relationships among different business domains (Thompson 99).

However, they do pose certain problems. One of the main problems is the different proposals for a message specification language (Stricker 99, Reichert 98). The previously presented protocols actually specify their own envelope in XML and their own underlying content description approach. This means that one domain that specifies its internal business processes in CBL can not interoperate with a domain that has specified its processes in CXML. In order to address this problem, certain harmonisation activities started to emerge. One of these activities is the unification and integration of CXML and BizTalk messages and specifications. It is anticipated that similar harmonisation activities will follow in order to enable semantic interoperability on different ontologies for different protocols and business sectors based on different frameworks (Wood 99, Ciacarini 98).

Another critical problem is the lack of application-independent messaging standards (Filos and Ouzounis 00b). The incompatibilities among different systems increase the problem and make

[1] Legitimate means that the message is compliant with the standard, in that case with the CXML proposed standard

the integration of business processes among different domains difficult. However, emerging standardisation activities are trying to solve the problem. These activities are the CORBA Messaging Service proposal (OMG 98) and the Java Messaging System (JMS 99) approach will probably solve, in the near future, these problems.

In addition to that, certain problems do exist on the transport protocol to be used for the exchange of messages. The existing frameworks require and deploy different transport protocols. For example CORBA message service assumes IIOP, while JMS proposes RMI and TCP/IP, BizTalk deploys the HTTP and MQSeries and BEA's system uses the TCP/IP. As the popularity and penetration of HTTP increases most of the developers and researcher agree to deploy open, Internet standards like TCP/IP and HTTP since existing middleware protocols like IIOP and RMI are natively based on TCP/IP. However, no certain adoption has been made so far.

Finally, one of the biggest problems in this area is the specification of certain ontologies or standard business entities for different business processes and sectors (Ciacarini 98, Spinosa 98, Stricker 99, Nissen 99). Standard ontologies and globally specified business process templates will enable the rapid integration and deployment of loosely coupled messaging systems for dynamic VEs. In order to achieve this, a standard open content description language is needed (Georgakopoulos 98, Fielding 98). XML seems to be the preferred option that will enable the solution to problem. However, XML is a generic meta-language that can be used for the specification of any type of ontologies and thus business sector activities are required towards this direction. The previously described frameworks, which are strictly related to business to business electronic commerce transactions, are the beginning towards this direction.

In general, messaging systems pose certain benefits over existing DCBS in the context of dynamic VEs due to the loosely coupled approach, the global ontologies, and independency among the interfaces of the components and the implementation of the components. However, before full deployment of messaging systems is done, critical issues, dealing basically with standardisation and XML acceptance, need to be solved.

3.2.4 Intelligent Mobile Agents

The success story of agents started in the early nineties with the parallel appearance of different agent concepts and technologies. These technologies can be roughly separated into intelligent agents and mobile agents (Guilfoyle and Warner 94, ComACM 94, Maes 94). In fact the term agent was used as a buzzword in these days, since not everything what was labelled as an agent, was based on agent concepts and agent technologies as we understand it today. The term "agent" has also been used for many years in the field of distributed computing, where it refers to specific client or server entities used in solving specific tasks in a distributed computing system, e.g. directory system agents, mail user agents, management agents, etc.

The interest in intelligent agents was coined by the increasing notion of Multi Agent Systems (MAS) and Interface Agents in the early nineties, driven by the Distributed Artificial Intelligence (DAI) research community (Wooldridge and Jennings 95). The multi agent system concept is largely conceived upon the idea that complex activities can be split into smaller activities and every small activity can split into smaller ones, until a primitive set of activities can be found. Every primitive activity can be provided from a special purpose software agent. Each agent co-operates with other agents inside the community to solve a particular complex problem. Therefore, a multi agent system may be defined as a set of agents that interact with each other and with the environment to solve a particular problem in a co-ordinated, i.e.,

behaviourally coherent, manner (Breugst 98, Choy 99, Magedanz 97). Therefore, agent communication and co-operation represents a major issue for this type of agents.

Another type of agents proposed so far was the Personal Assistants that support human users during their daily work. The major goal of these agents is to collaborate with the user, and hence the main emphasis of investigations clearly lies in the field of user/agent interaction. Some of these agents "locate" the behaviour of the user during his daily operation and can reveal intelligent behaviour. Agents of this type were the "smart mailboxes" and the "smart search engines". These agents do not only provide an intelligent interface to the user, but also make extensive use of the various services available in the network. In contrast to smart mailboxes, that perform advanced mail filtering based on users preferences, search engines collect knowledge available in the network on the user's behalf. This type of agent has also been called "KnowBots" or "Softbots" (Etioni 94).

Probably the most important boost in creating and establishing awareness for the term "agent", was the appearance of the mobile agent concept, coined mainly by a new technology called TeleScript developed by General Magic in 1994 (White 94). It was the time, where scripting languages, such as the Tool Command Language (TCL) and its derivative SafeTcl (Borenstein 94) gained much attention, since they enabled rapid prototyping and generation of portable code. The concept of smart messaging (Reinhardt 94), based on the encapsulation of SafeTcl scripts within emails (Borenstein 92), made new mail-enabled applications possible. The concept of mobile computing has gained increasing importance, enabled through mobile agent technology (Chess 95). Furthermore, the telecommunications domain has been considered as a main application area for mobile agents (Magedanz, 1996a). Last but not least, it was the beginning of the Java-age. Java is today the basis for most of the mobile agent platforms and systems.

TeleScript (General Magic 94), introduced as the "PostScript language for the network", was more than just a language, as it provided a complete mobile agent execution environment. This environment was designed to enable the implementation of the concept of remote programming, which was proposed as an alternative approach to the Remote Procedure Call (RPC).[2] The main idea was to ship small piece of code to the data and not the large amount of data to the code. Whereas at this time Safe-TCL and Java are primarily used to enable asynchronous operation and remote execution of "mail-enabled" and "WWW-enabled" applications within the Internet, the metaphor used within Telescript was the "electronic market place". Within this market, agents asynchronously perform tasks on behalf of users. They may communicate with the user, the services available in the network and other agents. In order to perform specific tasks, the agents migrate through a network to visit remote sites. This means that during its execution, a mobile agent may move from node to node in order to progressively accomplish its task. In other words, agents are capable of suspending their execution, transporting themselves, i.e., program code, data, and execution state, to another node, and resuming execution from the point at which they were suspended. However, due to its closed architecture and the coincident increasing acceptance of Java as the universal programming language, TeleScript has been abandoned in 1998 and has been replaced by a Java-based agent platform, called Odyssey, with less impact and success. Nevertheless, a lot of the concepts originating from TeleScript are still present in existing mobile agent systems and standards.

[2] The PostScript language can be considered a rudimentary form of the more general idea of sending programs to and executing them at a remote site as it involves sending a print programs to a remote processor in a printer.

Today there has been a lot of developments and general excitement in the area of mobile agent technology, much of which has evolved from the platform independence of the Java language with its inbuilt object serialisation and network communications support (Bellifernine 99).

Due to the mobility aspects, the intelligent capabilities and the autonomicy characteristics, agents become a fashionable and promising technology for the research and development community. However, this created also confusion, since there was a lack of common definitions and standards, resulting in various concepts, languages, architectures, technologies and terminologies. But this situation has changed a little with the establishment of common agent platform standards like OMG-MASIF (OMG 98), FIPA (FIPA 98), and Agent Communication Languages (ACL) like FIPA-ACL (FIPA 98) and KQML (Finin 94 and 95).

The term "Software Agent" (Bradshaw 97) has been adopted as the most general phrase to describe the concept of an autonomous software entity that automates some of the tasks of a human or another software process that have been delegated to it. An agent is an encapsulated software entity with its own state, behaviour, thread of control, and an ability to interact and communicate with other entities - including people, other agents, and legacy systems[3], in an autonomous, intelligent and proactive way. This definition puts an agent in the same family, but distinct[4] from, objects, functions, processes, and daemons. The agent paradigm is different to the traditional client-server approach. Agents can interact on a peer-to-peer level, mediating, collaborating, and co-operating to achieve their goals and objectives.

However, there is no unique definition of a Software Agent (Franklin 96) This reflects the diversity of theories, languages, architectures, technologies and standards. Nevertheless, today we can simplify our considerations on agents by reducing the whole spectrum of available concepts to two main categories, namely mobile agents and intelligent agents (Fuggetta 98, Choy 99, Zhang 98).

Mobile agents embody the ability to migrate seamlessly from one platform to another whilst retaining state information, typically with limited intelligence. By contrast, an intelligent agent is an agent that exhibits 'smart' behaviour. 'Smart behaviour' can range from very primitive operations achieved through following user-defined scripts, to adaptive behaviour of neural networks or other heuristic techniques. In general, intelligent agents are not mobile since the larger an agent is the less desirable it is to move it (Fünfrocken 98). By incorporating intelligent behaviour into autonomous agent requires usage of artificial intelligence techniques that result into an entity with undoubtedly bigger size.

Until recently, there has been a distinct line drawn between these paradigms and the two research domains have been focussing on quite different problems. However, the two research areas have been relying on a similar conceptualisation, i.e. that of utilising separate software processes. However, these processes may be implemented as procedure calls, a thread or several threads, but look and behave conceptually as autonomous processes for dealing with automating control tasks. It is also evident that the more specific attributes of these entities are merging towards a unified approach. This means that Mobile Intelligent Agent platforms, that enable and support agents with both characteristics, are starting to emerge.

[3] Not necessarily all of these for any one instance of an agent.

[4] An agent is at a higher level of abstraction.

In the following section the main characteristics of both types of agents will be discussed in order to clearly identify their major characteristics and features.

3.2.4.1 Intelligent Agents

Probably the most important attribute that distinguishes this type of software agents from other types of software processes is their ability to co-operate in some manner with other agents, human or software, either to gain a local advantage themselves or to add to the global 'good' of the system in which they operate via collaboration and coordination. This type of agents is sometimes called Intelligent Co-operative Agents (Barbuceanu 95, Bellifernine 99).

The abilities of software agents have been described eloquently by Wooldridge and Jennings (Wooldridge 95) and can be classified into the possession of Social Ability, Autonomy, Reactivity, Adaptability. Wooldridge and Jennings (Wooldridge 95) provide a diverse review of intelligent software agent research the interested reader is directed to. In the following, only the aspects of agent communication and cooperation will be discussed and presented. [5]

Communication enables agents in a multi-agent environment to exchange information on the basis of which they originate their action sequences and co-operate with each other. For example, to allow an agent to inform another agent about its current beliefs, amount of resources available, additional information about its environment, etc. Software agents generally communicate with other agents in order to work more flexibly. In order to achieve this level of co-ordination, the agents should interact and exchange information. This is done by means of an Agent Communication Language (ACL), which is a language with precisely defined syntax, semantics and pragmatics.

Agent communication is accomplished through the use of three components: the ACL, content language, and ontology. An ontology enumerates the terms comprising the application domain and is not unlike a data dictionary in a traditional information system. The content language is used to combine terms in the ontology into sentences, which are meaningful to agents who have committed to this ontology. Sometimes the ontology and content language are so tightly integrated that they become the same thing i.e., a list of sentences is the content language, which represent the ontology. Finally the ACL acts as a protocol, enabling the development of dialogues containing sentences of the content language between agents and defining certain semantics for the behaviour of agents participating in such dialogues.

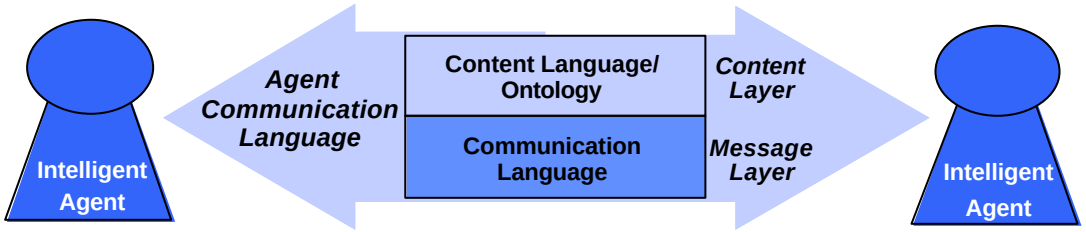

Figure 4: Agent Communication Entities

[5] A comprehensive definition and introduction to intelligent agent technology and the related AI, DAI background is beyond the scope of this chapter and this thesis. Therefore the interested readers should refer to (Wooldridge 95) for aspects of Intelligent Agent theory, languages and architectures and (Nwana 96) for a detailed discussion on background and the application domains of software agents.

Most ACLs derive their inspiration from speech act theory (Searle 69), which was developed by linguists in an attempt to understand how humans use language in every day situations, to achieve everyday tasks, such as requests, orders, promises, etc. It is based on the idea that with language the speaker not only makes statements, but also performs actions. A speech act can be put in a stylised form that begins "I hereby request …" or "I hereby declare …". In this form the verb is called the performative.

The probably most prominent ACL, representing the defacto standard before FIPA ACL (FIPA 98b) became available, was the Knowledge Query and Manipulation Language (KQML) (Finin 94 and 95) which defines a framework for knowledge exchange. KQML focuses on an extensible set of performatives or message types, which define the permissible operations that agents may execute on each other's knowledge and goal stores. As the content of the messages was not part of the standard, the Knowledge Interchange Format (KIF), a formal language was defined based on first-order predicate calculus for interchanging knowledge among disparate computer programs (Ginsberg 91). KQML and FIF have been developed in the context of the DARPA Knowledge Sharing Effort (Ginsberg 91).

Some of the benefits cited in the literature for intelligent agents, i.e., multi agent systems, can be summarised as to:

- address problems that are too large for a centralised single entity due to resource limitations, robustness concerns, or fault recovery,
- enable the reduction of processing costs. It is less expensive, in hardware terms, to use a large number of inexpensive processors than a single processor having equivalent processing power,
- improve scalability and adaptability. The organisational structure of the agents can dynamically change to reflect the dynamic environment, i.e., as the network grows in size the agent organisation can re-structure by agents altering their roles, beliefs, and actions that they perform;
- provide solutions to inherently distributed problems, i.e. where the expertise is distributed.

At a high level, the multi-agent systems approach is intuitively simple. Developers can draw on their experience in solving problems in real world co-operation with others. Therefore, the MAS paradigm is inherently scalable due to modularity, loose coupling between interacting elements, and higher levels of design abstraction because the level of abstraction is greater than that of the object level.

3.2.4.2 Mobile Agents

Mobile agents, also referred to as transportable agents or itinerant agents, are based on the principle of code mobility. Mobile code enhances the traditional client/server or Remote Procedure Calls (RPC) paradigm[6] by performing changes along two orthogonal axes:

[6] In the client/server paradigm the server is defined as a computational entity that provides a set of services. The client requests the execution of these services by interacting with the server. After the service is executed the result is delivered back to the client. The server therefore provides the knowledge of how to handle the request, as well as, the necessary resources.

- Where is the know-how of the service located?
- Who provides the computational resources?

Three main paradigms for mobile computations have been identified (Fugetta 98). These are: Remote Evaluation, Code On Demand, and Mobile agents. These paradigms differ in how the know-how, the processor, and the resources are distributed among the components of a distributed system. The know-how represents the code necessary to accomplish the computation. The resources are located at the physical node that will execute the specific computation.

In the *Remote Evaluation (REV)* paradigm (Stamos 70) a component A sends instructions specifying how to perform a service to a component B. The instructions can, for instance, be expressed in Java bytecode. B then executes the request using its resources. Java Servlets are an example of remote evaluation.

In the *Code on Demand (CoD)* paradigm the same interactions take place as in remote evaluation. However, the difference is that the component A has the resources collocated with itself but lacks the knowledge of how to access and process these resources. It gets this information from the component B. As soon as A has the necessary know-how, it can start performing its operations. Java Applets fall under this paradigm.

The *Mobile Agent (MA)* paradigm is an extension of the remote evaluation paradigm (White 94 and 97). Whereas the latter focuses primarily on the transfer of code, the mobile agent paradigm involves the mobility of an entire computational entity along with its code, state, and potentially, the resources required to solve a problem to fulfil a goal.

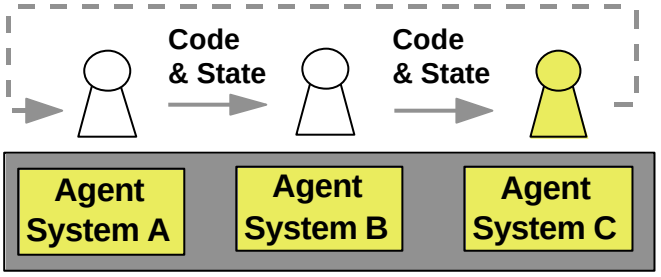

Figure 5: Mobile Agent Approach

By adopting the Mobile agent paradigm, the component A has the know-how capabilities and a processor, but lacks the resources. The computation associated with the interaction takes place on the component B, which has a processor and the required resources. For example, a client owns the code to perform a service, but does not own the resources necessary to provide the service. Therefore, the client delegates the know-how to the server where the know-how will gain access to the required resources and the service will be provided. An entity encompassing the know-how is a mobile agent. It has the ability to migrate autonomously to a different computing node where the required resources are available.

This means that a mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another. The ability to travel permits a mobile agent to move to a destination agent system that contains an object with which the agent wants to interact. Moreover, the agent may utilize the services of the destination agent system. When an agent travels, its state and code are transported with it. In this context, the

agent state can be either it's execution state, or the agent attribute values that determine what to do when execution is resumed at the destination agent system. The agent attribute values include the agent system state associated with the agent.

The mobile agent paradigm provides two alternative programming approaches, namely:

- remote execution: an agent is sent to a remote location before its activation. The agent remains in this location during its entire life-cycle
- migration: an agent is able to change its location during its execution. A mobile agent can start its execution in location A then moved into location B invoke a service in this domain and return back to the original location.

The Mobile agent paradigm is important for network-centric systems because it represents an alternate, or at least, a complementary solution to traditional client/server approaches (Chess 95). Such solutions may contribute to a reduction of the overall communication traffic in network.

Mobile agents are typically developed by means of machine-independent programming languages. The initial pioneering machine-independent languages, such as Save-Tcl and Telescript, are today mostly replaced by Java, due to its inherent portability and platform support. Nevertheless, the native capabilities of Java are not yet sufficient for implementing right away mobile agents. Extra functionality has to be implemented for realising a mobile agent that support agent transport, mobility management, and security. For that reason, mobile agent platforms have been developed in order to provide the required functionality for implementing mobile agents. These platforms provide value added services for migration of agents and thus, enable rapid development of applications with mobile agent features.

The following benefits have been most often cited (Harrison 95, Chess 98):

- asynchronous/autonomous task execution: after the injection of an agent into the network environment the user can perform other tasks,
- reduction of network traffic and client processing power: since massive data exchanges are handled locally at the nodes hosting the data and client computers could concentrate on performing only limited local tasks,
- increased robustness and reduction of dependence of network availability and client/server availability: once the agent arrived at a target system the client may crash or the network may become unavailable without any drawbacks on the task processing,
- automation of distributed task processing: agents have itineraries that determine what tasks they have to perform where without any user interaction,
- decentralised and local task processing: agent cloning enables the automated distribution of formerly centralised programmes,
- flexibility: On-demand software distribution / service provisioning – service software within mobile agents can be instantly downloaded to client and server nodes.

The mobile agent paradigm provides flexibility by re-distributing intelligence inside a distributed network environment, particularly for reducing network load and optimising service performance. Due to the above stated benefits, various problems and inefficiencies of today's client/server architectures can be handled by means of this new paradigm.

The down side of this technology is in fact the security risks introduced. An agent may be attacked, modified or deleted by a hostile agent platform. Another obvious concern related to mobile agents is the question, if agent migration is always of advantage in contrast to message passing. For example, it is probably better to interact for small information exchanges by message passing in case the agent code is bigger than the expected data volume to be exchanged.

In summary, agent technologies have a lot of appealing advantages compared to traditional technologies for solving specific problems. But they imply the introduction of new agent platforms enabling mobility and/or advanced inter-agent communication in the target environment and require the adaptation of existing interfaces to that new agent environment. Adopting a more general view, legacy technologies, including distributed object technologies, have also advantages in specific environments and more importantly a big installed base. For several applications, Remote Procedure Calls still represent a powerful and efficient solution. Thus, an integrated approach is desirable, combining the benefits of both client/server and agent technology and on the other hand, minimizing the problems that rise if one of these techniques is used as "stand-alone" solution (Guilfoyle 94). Therefore, an integration of agent technologies with existing technologies represents the best solution to combine their advantages (Choy 99, Harrison 95, Karmouch 98).

3.2.4.3 Intelligent Mobile Agent in the context of VEs

Intelligent Mobile agents provide significant benefits in relation to traditional distributed object oriented approaches. Some of the major benefits emerged from the usage of intelligent, mobile agents are: autonomy and flexibility due to the co-operation aspects among agents, scalability due to the migration capabilities, adaptability due to intelligent behaviour, and integration with existing technologies due to the object oriented concepts used to implement agent platforms and agents (Krause 96 and 97, Fuggetta 98).

Agents seem to provide certain benefits for the development of dynamic VEs. Taking under consideration the key requirements of VE in relation to agent characteristics, it becomes evident that this technology might offer significant benefits to the development of inter-domain business process management. However, this area is totally unexplored and further research is needed (Choy 99, Martesson 98).

Intelligent mobile agents can be used in different ways to solve effectively VE problems. One way is to use an agent based business process management systems that control and co-ordinate in a distributed, autonomous, and flexible way the execution of VE business processes (Filos 00). Another way is to use agents is the negotiation and partner selection phase among different VE partners prior to contract establishment. The autonomy and intelligent characteristics of agents can significantly automate the negotiation process (Kraus 98). Agents can also be used to manage and co-ordinate the provision of matchmaking services, like virtual marketplace services (White 94, Ouzounis 98e). Potential supplier agents can migrate to the marketplace and register their capabilities by communicating with the marketplace agents. In the sequel, consumer of services can visit the virtual marketplace, locate the best suppliers, negotiate about certain attributes, and select the best ones. Additionally, agents can migrate to different physical locations, where business logic exists, and deploy business services by reducing the network load and traffic (Ouzounis 99b).

Although intelligent mobile agents are a very good and promising technology and paradigm for the development of dynamic VE systems, they do introduce some problems as well (Lange 96,

Lin 96). One of the key issues is the requirement for a mobile agent platform that will enable agent life-cycle operations and migration services. Certain mobile intelligent agent platforms have been developed so far and a lot of standardisation activities have been emerged, like OMG-MASIF and FIPA that try to deal with all these problems (Ciacarini 98). Another issue was the lack of a standard Agent Communication Language (Bellifernine 99, Borghoff 97). In that respect, FIPA-ACL, and its predecessor KQML, is an emerging proposed standard that can be used to enable inter-operation and communication among different MAS prototypes and systems (FIPA 98 and Finin 94).

In addition to the benefits coming from the usage of agent concepts, agents seem to combine all the benefits offered by the messaging systems and DCBS (Choy 99, Breugst 98). Agents communicate by exchanging ACL messages in an asynchronous and loosely coupled way using underlying messaging systems. However, agents deploy the concept of ontologies that make them more flexible, pro-active, intelligent, and autonomous. Agents are deployed within a distributed object oriented platform, like CORBA or Java Framework, and thus can access any type of standard business component. Additionally, agents can migrate to different physical locations where business components exist and thus preserve the network resources. Finally, agents have the ability to execute, manage, and co-ordinate complex business processes, in a similar way like workflow management systems (Barbuceanu 95, Cai 96). Due to the high degree of distribution, autonomy, co-operation and coordination mechanisms, intelligent, autonomous agents are perfectly positioned to manage and execute complex business process in a dynamic manner, across-domain boundaries. The execution of the process is not controlled in centralised and static way, like in conventional workflow management systems, but on the contrary, the agents themselves are co-operating in a flexible, autonomous, and dynamic way to execute the process (Chess 95).

It seems that intelligent mobile agents satisfy most of the key requirements of VEs (Camarinha-Matos 99, Filos and Ouzounis 00a). However, due to the fact that this technology is rather new, no significant research efforts have been done so far in the area of agents, business process execution and management across-domain boundaries and dynamic VEs. One of the key objectives of this thesis is to explore the way that intelligent mobile agents can be used in the context of dynamic VEs.

3.2.5 Workflow Management Systems

Workflow management is one of the areas that, in recent years, has attracted the attention of many researchers, developers and users. Concepts and systems such as computer supported cooperative work, paperless office, form processing, cooperative systems, and office automation, have been delayed decades, in some cases, for the technology and know-how required to implement real systems.

Workflow management systems (WFMS) are used to coordinate and streamline business processes (Adams 97, Georgakopoulos 95 and 98). Typical business processes are loan approvals, insurance claims processing, and billing. These business processes are represented as workflows, i.e., computerized models of the business process, which specify all the parameters involved in the completion of these processes. Such parameters range from defining the individual steps, to establishing the order and conditions in which the steps must be executed, including aspects such as data flow between steps, who is responsible for each step, and the applications to use within each activity.

A Workflow Management System is (WfMC 98, Grefen 99, Miller 98, Lee 93) the:

- set of tools used to design, define, and specify business processes utilising a business process definition language, widely known as business process modelling tools,
- environment or workflow engine in which these processes are executed and managed, widely known as workflow engine, and
- set of interfaces to the users and applications involved in the workflow process, widely known as application interfaces and tasklists.

There are many parameters involved in the specification of a workflow system. In spite of the efforts of different standardisation bodies, the term workflow is still very fuzzy and used in many different contexts. Workflows usually associated with the concept of business processes, which is also not very precise. The reference model of Workflow Management Coalition (WfMC) (WfMC 98) defines a business process as "a procedure where documents, information, or tasks are passed between entities of the workflow according to defined sets of rules to achieve, or contribute to, an overall business goal". In general, a workflow is a representation of the business process in a machine-readable format. Hence, a workflow management system is "a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic" (WfMC 96).

Workflow systems can be categorised based upon the way that they execute and manage business processes. Two approaches have been proposed so far, namely the centralised and distributed one (Khare 99, Martesson 98). In the centralised approach, there is only one workflow engine. This engine controls the execution of different business process instances and has full control upon each instance. The external applications can be located either on the same system or on different systems. It is obvious that centralised systems have several drawbacks like load balancing and scalability problems in comparison to distributed ones (Grefen 99).

In the distributed approach, there are more than one workflow engines that partially execute instances of a business process instance. One process instance can start in one engine and parts of it can be executed in another one (Eder 95). This is a more flexible and distributed solution. However, due to the distributed execution of business processes from different workflow engines, some synchronisation problems might be created.

Workflow Systems can also be categorised based on the underlying communication mechanisms used to support the interfaces between the process engine and the external applications (Georgakopoulos 95). Four key categorise have been emerged so far.

The conventional client/server workflow management systems or 2-tier systems were the first approach widely adopted. In that case, the process engine and the external applications and instances use conventional client/server concepts like TCP/IP and relational databases. The interfaces among the entities are static, programming language dependent, and network protocol specific. These systems are in general closed, not adaptable, and inflexible. One of the major drawbacks of these systems is the low degree of distribution and autonomy (Miller 98). These systems deployed for intra-domain business process execution and management where the interfaces of the external applications are well known, centrally specified, and static. These systems are considered the first generation of the workflow management systems (Filos 00).

The object-oriented workflow management systems or n-tier systems were the second approach used. In that case, there are more than one workflow engines that co-operate with the external

entities by deploying distributed object-oriented platforms, like CORBA, RMI, or DCOM (Adams 97). The communication approach among the involved entities is remote procedure calls. These systems characterised by adequate degree of flexibility and distribution. There are static and well-defined interfaces for the integration of legacy applications and systems. However, these systems do not provide the appropriate degree of autonomy, intelligence, and dynamic behaviour (Alonso 95). The business process execution and management is static, pre-defined and no alterations during process execution can be easily done. Additionally, external applications and services need to implement special, static, and pre-defined interfaces, something that it is not always feasible especially in the case of different administrative domains (Bolcer 99). Finally, as in the first case, most of the OO-workflow systems are deployed for intra-domain business processes, the semantic of activities and sub-processes is well known, pre-defined, and centrally specified (Georgakopoulos 95).

The message-based workflow management systems, or n-tier message clients, were another approach proposed (Eder 95 and 96). According to this method, the workflow engine and the external entities co-operate by the exchange of messages with well-defined format, syntax, and semantic. Every activity within the workflow system is considered a message client that can send and receive messages from others. The underlying communication system is a messaging system that undertakes the responsibility to forward the messages to the corresponding receivers (Alonso 95). The communication model can be either synchronous or asynchronous, and point-to-point and/or multi-point. A special category of message-based workflow management systems is the event-driven ones. In these systems the different entities of the workflow system co-ordinate their activities with the exchange of events that have specific format (Geppert 98, Grefen 98, Tombros 99). These systems depend on the underlying event management system and the syntax and semantic of event description language. Initially, message-based workflow systems used proprietary content description languages usually specified in ASCII format. For that reason, these systems can be characterised as closed. These systems are suffering by the same problems as the previous ones, i.e. limited degree of autonomy and flexibility and low level of adaptability. More specifically, event driven systems can be not explicitly used for data passing among workflow entities. On the contrary, they are mostly used for notification of process status, i.e. events can be used as a coordination mechanism (Grefen 98). Finally, message-based workflow systems have been basically deployed for intra-domain purposes. With the advent of XML, the syntax and semantic of the exchanged messages among the entities of the workflow system can be described in a flexible and open way. Recently, different emerging workflow management standardisation committees are trying to integrate XML semantics into their specifications (WfMC 98).

The agent-based workflow management systems are rather new category of systems. According to these systems, the execution and management of business processes is performed by a set of autonomous, intelligent agents that co-operate, in a distributed manner, to execute the process and thus, to reach a business goal (Barbuceanu 95, Borghoff 97, Bellifernine 99). The autonomous agents can execute and manage either parts of processes or tasks of processes. In any case, the autonomous agents communicate through special message exchanges utilising special co-ordination protocols, like FIPA-ACL or KQML. The content and semantic of the messages are being described in open, globally-specified ontologies that the agents can understand. The agent-based workflow management systems are more adaptable and dynamic due to the open ontologies used and the autonomous and adaptive characteristics of agents. This category is considered new scientifically and not so much research has been conducted due to the rather new concept of agent technology. The main reason for this delay was the lack of

mobile agent platforms, standard agent communication languages, and globally-specified ontology specification frameworks.

From the above categorisation and discussion, it is clear that all of the approaches proposed so far have significant benefits and drawbacks regarding their deployment in the context of inter-domain business process execution and management (Filos 00). However, the acceptance of agents, as an implementation and communication paradigm, the extra capabilities that they offer, like mobility, autonomy, intelligence, adaptability, in conjunction with emerging state of the art agent platforms, like FIPA and OMG-MASIF, standard distributed platforms, like CORBA and RMI, flexible content description languages for globally specified ontologies, like XML, emerging XML-based workflow standards, like WfMC, and platform independent programming language, like Java, can provide the basic technological building blocks for the new generation of open, flexible, autonomous, adaptive, and distributed workflow management systems for dynamic VEs.

In the following sections, the basic standardisation activities in the area of WFMS are presented and further analysed. In the sequel, the applicability of the workflow management concept in relationship to intelligent agents and dynamic VE concepts is presented.

3.2.5.1 Workflow Management Coalition

The Workflow Management Coalition (WfMC) was founded in August 1993 by workflow vendors, users, and analysts with the aim of promoting the use of workflow technology through the establishment of standards for terminology, interoperability, and connectivity between different vendor products (WfMC 98).

Initially, the WfMC specified a set of definitions for the workflow management systems, namely terminology and glossary document. Most of these definitions are used now days by most of the workflow vendors and researchers. These definitions established consensus among the developers and clarified a lot of open issues and fuzzy terms. The evolution of these definitions is necessary for the development of future workflow systems.

Another key contribution of WfMC was the identification of key modules of a workflow management system and to propose a Reference Architecture for interoperable workflow management systems. The main objective of WfMC was not to standardise the functionality of the different modules of the architecture but the interfaces among them. To enable interoperability, the WfMC proposes standardisation of five major interfaces and certain data interchange formats.

The interface around the workflow enactment service is called the WAPI - Workflow APIs and Interchange formats, and is seen as a unified service interface to support the five types of workflow interfaces[7]. The WAPI is defined as a common set of API calls and interchange formats with specific extensions, where necessary, to cater for the specifics of the five interfaces. For every interface a working group has been created. The WAPI enables services of the workflow system to be accessed and can control the interaction of the workflow system components. More detail about the use of WAPI within the Workflow Enactment Service and over each of the five interfaces can be found in (WfMC 94-98).

[7] A number of the functions within each of the five interfaces are common.

Based on this concept the WfMC proposed the following key interfaces:

- **Interface 1** The Process Definition Tools Interface, a standard interface between business process definition tools and workflow engines.
- **Interface 2** The Workflow Application Client Interface, a standard interface for communication between a workflow engine and a client interface, that is the interface through which work items are presented to the user.
- **Interface 3** The Invoked Application Interface, a standard interface that allow a workflow engine to invoke a variety of external applications.
- **Interface 4** The Workflow Interoperability Interface, standards that will allow workflow systems produced by different vendors to inter-operate by pass work items between each another.
- **Interface 5** The Process Auditing & Administration Interface, a standard interface between monitoring applications and workflow engines thus allowing one vendors monitoring application to work with another's workflow engine.

To date the WfMC has published standards for all interfaces. It has also published an OMG IDL binding for its Client Application Programming Interface (Interface 2) and is has been developing a similar binding for the Interoperability Interface. These standard APIs should ensure openness between various workflow products and indeed the WfMC has already demonstrated interoperability between workflow products.

Independent of the success of the WfMC and the conceptual framework and interfaces that have been proposed, most of the commercial workflow systems do not offer open interfaces between the different functional components. Systems often combine several functional components as a single logical entity with the embedded interfaces and they utilise different underlying technologies like messaging systems, TCP/IP, or object oriented middleware platforms like CORBA.

In September 1999 the Workflow Management Coalition (W*f*MC) launched a draft specific ation of Wf-XML, which seeks to provide XML-based workflow standards. The specification builds on the foundation of W*f*MC's earlier work, providing an evolution of the existing workflow standards into XML-based exchanges between workflow systems.

The W*f*MC initiative has brought together the work originated in the OMG jointFlow submission (see next section) and the initial proposals from the IETF sponsored Simple Workflow Access Protocol (Bolcer 99) (see next section). Wf-XML is an XML-based variant of the W*f*MC Interoperability Interface that can work with HTTP or a number of other transport mechanisms email, direct TCP/IP connection, or messaging systems.

The specification, currently at draft level, includes a definition of the basic DTDs defining the XML encoding of workflow messages to support interoperability. The intention is to extend the specification to include workflow operations from other W*f*MC interfaces and to form a complete XML-based specification for all workflow functions. As part of this work, the interoperability specification has been implemented as a prototype on two different workflow systems and has been successfully tested in different interoperability trials. During the second quarter of 2000, the W*f*MC intends to refine and release the WF-XML specification, as a full standard and continue to invite input and comment from other industry groups including the IETF and W3C.

The intentions of Wf-XML were not towards the specification of a inter-domain workflow management standard. This version does not bring any new concepts, but it is actually a direct translation of the existing interfaces to XML format with some minor improvements, like session management. Critical open issues, like inter-domain workflow execution and management, business process specification for inter-domain business processes, dynamic selection of workflow providers during process execution, autonomous and intelligent behaviour with emerging agent concepts and ideas, are not discussed at all.

3.2.5.2 OMG's Workflow Management System

The Object Management Group initiated a similar activity to standardise workflow systems in late 1997 (OMG 98). The standardisation activity was in the context of business objects activity and the main emphasis was the CORBA-based communication and interoperation among different workflow modules.

The OMG has issued a Request For Proposal (RFP) for its Workflow Management Facility in mid 1998. Four initial submissions were made. The accepted submission was the jFlow that has been made by a consortium influenced by WfMC members. The accepted proposal is based on the WfMC concepts and standards and thus, WfMC backs the standardisation activities. It is the intention of this consortium and the WfMC to have one workflow management Facility specification endorsed by both WfMC and OMG.

The currently accepted standard addresses the following interfaces:

- WfRequester, WfProcessMgr, for workflow execution control, monitoring and interoperability in a similar context as WfMC. This interface corresponds to the Interface 2,3, and 4 of the WfMC.
- WfProcess, WfActivity, WfExecutionObject, for business process definition and execution. This interface corresponds to the Interface 1 and 2 of the WfMC.
- WfAssignment, for worklist management. This interface corresponds to the Interface 3 of the WfMC.

In addition to that, OMG recently announced an RFP for Resource Management and Assignment Facilities to allow facilities that perform the assignment of resources to be interfaced to multiple workflow management systems. The RFP covers issues like resource assignment management, resource selection based on criteria and policies, and resource specification and it is due to April 2000.

OMG Workflow Management Facility is aligned to some degree with the standards of the WfMC. In this case a set of IDL interfaces have been emerged corresponding to the interfaces defined in the WfMC reference model. This would mean that applications could be workflow enabled by augmenting them with the appropriate IDL interface. The application could then be used with any workflow component compliant with the standard.

The current proposed OMG standard are not directly dealing with cross-organisational business process execution and management. As in the case of WfMC, critical open issues, like inter-domain workflow execution and management, business process specification for inter-domain business processes and dynamic selection of workflow providers during process execution are not discussed at all. Additionally, as has been explained in the DCBS section, the deployment of CORBA as a mechanism for autonomous inter-domain business process execution and

management is rather problematic and in general, inflexible due to the tight coupling approach. Therefore, it is anticipated that a message-based approach with corresponding XML message requests and responses would have been better since the degree of autonomy and flexibility is increasing.

3.2.5.3 Simple Workflow Access Protocol

The Simple Workflow Access Protocol (SWAP) is a simple, lightweight proposed protocol for communicating information about long-lived workflow activities and processes over the Internet and especially though the HTTP (Bolcer 99). The main motivation behind the proposal was the ability to integrate workflow providers and workflow performers to support asynchronous services across Internet, intranet, and extranet. In that way, multiple clients can use business processes from different business domains utilising standard Internet protocols.

The proposal has been submitted in IETF by Netscape, HP, and Sun in August 98 and it is under public discussion and debate. SWAP uses, as underlying Internet protocol, the HTTP and, as content description language, the XML. The SWAP protocol uses a simple, though extensive request-response model. Every workflow related request is a HTTP packet-request, while the requested process, the input parameters and the values for these parameters are XML content. Every response is an HTTP packet-response that contains results formatted in XML. This approach is very strong one, since it is dependent only on the transport protocol used, the XML content used to describe the requests and responses, and the protocol used, i.e. the SWAP. However, how these requests will be serviced on intra-domain level is left open for the workflow management providers and developers.

The main operations that SWAP supports are:

- **initiate**: create remotely set-up and invoke a business process,
- **monitor**: check the business process instance current status, get a history of the execution, or find out the current and possible states of a running process,
- **manage**: read and set the running state of remote, generic, asynchronous workflow services, pause or resume and executing one or terminating it when no longer needed,
- **notify**: send appropriate notifications of status changes to interested observers during normal execution or when exception occur.

SWAP is not a complete workflow standard. It addresses only the interface 4 of WfMC, i.e. the workflow to workflow interoperation with major emphasis on Internet based business process execution and management. It uses HTTP as transport protocol and XML as a content language and has a set of different semantics for methods and data structures, as WfMC and OMG's jointFlow have. In that respect, SWAP has not clear relationship to existing workflow standard community, however, it addresses a serious problem, i.e. the deployment and usage of workflow management systems through Internet standards, something that the other two standardisation committees failed to do consistently (Bolcer 99). OMG and WfMC have strong interests to incorporate XML technology and solve problems related to asynchronous execution and management of business process across the Internet. In that respect, SWAP is a good possibility towards this direction. However, explicit steps towards harmonisation of SWAP ideas in interface 4 of WfMC and OMG standards have not been published.

3.2.5.4 Workflow Management Systems in the context of VEs

Workflow management systems are used to specify, execute, manage, co-ordinate, and streamline business processes. However, most of the workflow concepts have been used only on an inrta-domain level, i.e. for business processes that are totally controlled by only one organisation. Due to the emerging models of dynamic VEs, a clear need for distributed, inter-domain workflow management systems, has been emerged (Grefen 99, Miller 98).

Workflow management systems feature a set of good attributes for deployment within the context of a VE. The shared business processes among the VE members can be described by deploying a business process specification language. In that way, shared business processes can be easily developed. For example a shared process can start in one domain and then, can be continued in another domain-partner, by utilising remotely a sub-process. The WFMS will undertake the responsibility to control and manage the execution of the shared business process in a distributed and systematic way (Hoffner 98, Ouzounis 99b).

Although traditional WFMS systems have significant benefits, they do have certain drawbacks in relation to the VE concept. One of the main issues is the limited autonomy and flexibility that they have. WFMS execute upon well-defined business processes specified in a specification language that the system could understand. So far there are no certain extensions to the existing business process specification languages towards the direction of cross-organisational business processes (Filos 00). Additionally, remote invocation of business processes, provided by different business domains, should follow access control, authorisation and contract checks. Current workflow systems do not provide such mechanisms (Klingemann 99). Finally, in current prototypical workflow systems, shared business processes are being specified statically in relation to remote processes, i.e. the VE partners that will provide them specified statically (Tombros 99, Geppert 98). This approach is suitable for static VEs and not for dynamic ones, where the partners that can provide parts of the shared business process are not known in advance. On the contrary, the remote domains are being selected dynamically after negotiation and during the business process execution. This means that for the same business process specification different instances might exist. For every instance a set of different partners might be selected according to the needs and requirements of the process (Ouzounis 99b).

Current proposed standards are not directly dealing with cross-organisational business process execution and management. Critical open issues, like inter-domain workflow execution and management, business process specification languages for inter-domain business processes, and dynamic selection of workflow providers during process execution are not discussed at all (Bolcer 99). Additionally, as has been explained in the DCBS section, the deployment of tight-couple communication mechanisms, like CORBA, as a mechanism for autonomous inter-domain business process execution and management is rather problematic and in general, inflexible. Therefore, it is anticipated that a message-based approach with corresponding XML message requests and responses would have been better since the degree of autonomy and flexibility is increased (Georgakopoulos 98, Miller 98).

Agent-based workflow management systems seem to be in position to solve some of these problems. For example, the execution of the shared business processes could be controlled by agents that can migrate to the remote business domain, invoke the required business processes, by identifying themselves, and return back to continue their normal execution. The physical location of the remote domain should not be known in advance, but can be found after selection and negotiation with potential providers in a virtual marketplace. The remote domain can also

authenticate and authorise the requesting agents based on electronic contracts that have been established during the negotiation phase.

However, most of the issues related to agent-based workflow management systems, cross-organisational business process execution, and dynamic selection of partners are under investigation and certain solutions and concepts are required (Tombros 00).

One of the key issues of this thesis is to investigate and explore how intelligent mobile agents and workflow management systems can be used in an effective, flexible, and dynamic way to provide a complete solution for dynamic VEs.

3.2.6 Virtual Marketplaces

Virtual marketplaces are a third-party virtual environment where buyers and sellers can meet and engage themselves in electronic commerce activities, like buying and selling of products and services. Due to the advent of electronic commerce, several types of virtual marketplaces have been proposed and developed so far with emphasis on different aspects (Bichler 98, Guttman 98). However, the general features of the virtual marketplace remain the same.

When a seller is in position to provide a service or a product, he registers his service offerings in a virtual marketplace in relation to a generic service or product template. The service or product template actually determines the characteristics and properties that the service or product has. When a buyer would like to buy a service or a product, he conducts the marketplace and retrieves all the potential sellers that can provide the service or product. The selection process is done based on some criteria or constraints that the buyer specifies. After the potential buyer has identified a set of potential sellers, a negotiation process might occur. However, the negotiation process is an activity related to the buyers and sellers and not directly with the virtual marketplace functionality as such (Beam 96, Parsons 99). In that respect, a virtual marketplace is a matchmaking service and resembles a directory service with some additional customised services for electronic commerce purposes.

In general, virtual marketplaces offer three types of services:

- administration of service or product templates, like creation, deletion, and modification of service types,
- management of registration of service/product providers, like register, deregister, modify service offers related to specific service types,
- management of seller selection requests based on some constraints, like select all offers related to a specific service type and satisfy some constraints.

Different approaches have been proposed so far for the realisation of such services. The most influential ones are, the Open Distributed Processing (ODP) Trader and the subsequent compatible standard of Object Management Group (OMG) Trader, the ISO's X.500 directory service, the Internet directory services like Domain Name System (DNS), and Lightweight Directory Application Service (LDAP), and the recently proposed de facto standard Java Naming and Directory Interface (JNDI). Each one of these services has gained success and penetration in different technical areas. The OMG-Trader is deployed in the area of distributed object oriented platforms, LDAP and DNS in Internet related services, and JDNI is an emerging de facto standard in Java based applications.

Though these services are being used in different platforms and for different purposes, they do have the same techniques for the definition of service templates. In general, a service template has a name and a set of named properties, which are actually (name, value) pairs. A service template can extend other existing templates by using the concept of inheritance. All the above-mentioned standard directories or matchmaking services are providing all three key types of services required for the realisation of a virtual marketplace or a matchmaking service (Filos and Ouzounis 00).

However, most of the above services have been used for the dynamic location and utilisation of resources, objects, and services within a distributed, homogeneous and intra-domain environment and not for electronic commerce purposes. An emerging number of research activities are concentrating now on the development of third-party marketplaces for electronic commerce purposes (Sierra 97, Wurman 99). The main emphasis on these activities is not only on the development of matchmaking mechanisms but also on the provision of mechanism that will enable negotiation between the buyers and sellers (Sandholm 95).

In the following section, an analysis and assessment of the currently proposed methods and techniques regarding negotiation are presented.

3.2.6.1 Negotiation

Negotiation is a process by which a joint decision is made by two or more parties. The parties first verbalize contradictory demands and then move towards agreement by a process of concession making or search for new alternatives (Bichler 98). Negotiation in electronic commerce can be defined as the process by which two or more parties multilaterally bargain resources for mutual intended gain using tools and techniques of electronic commerce (Beam 96, Zeng 96). In general, the negotiation process can be categorised in two major areas, namely the cooperative negotiations and the competitive negotiations (Bichler 98). In cooperative negotiations, different entities are negotiating to achieve mutual gains, i.e. this is a win-win scenario. In competitive negotiations the involved parties are totally autonomous, non cooperative, and they are trying to maximise their individual gain, i.e. this is a win-lose scenario (Milgrom 82). In the context of this thesis, the competitive negotiations will be considered and further analysed (Ouzounis 99a).

The negotiation process can be either automatic, semi-automatic, and human based (Beam 96, Sierra 97, Wurman 99). Automated negotiations take place when the whole negotiation function is performed by autonomous software entities without human intervention (Beam 96). On the contrary, in human based negotiations humans are taking decisions and influence the whole process. Finally, in semi-automated negotiations, the whole process of negotiation is supported by autonomous entities and when an agreement should be reached, human operators intervene. In the context of this thesis, only the automated negotiation will be considered and further analysed[8].

Furthermore, the negotiation process can be either, single-issued, or multi-issued. In the single issued case, only one issue or property is the context of the negotiation. All the efforts and interactions among the involved entities are concentrated in the maximisation or minimisation of

[8] Interesting readers about semi-automatic and human based negotiations should refer to [Beam97, Perkins96, Raiffa82, Sycara96]

this issue. Most of the research prototypes and systems today are concentrated on the single-issue negotiation. Especially in the electronic commerce field, the price of the product or service is usually the issue under negotiation (Sandholm 95). In multi-issued negotiation, more than one issues or properties are the context of the negotiation. Multi-issued negotiations are more complex, require more advanced decision and strategy techniques, and negotiation protocols. Multi-issued negotiations are a very active research field especially for business to business electronic commerce (Bichler 98).

Several forms of negotiations have been proposed so far. However, "the basic finding of the negotiation science is that there is no single negotiation protocol and schema for all possible negotiation situations" (Sandholm 95). Automated negotiations are still an open research field due to the fact that several critical issues should be taken under consideration. Some of the most critical issues involved are the:

- number of participants involved in the negotiation, i.e. buyers and sellers. The different possibilities are one seller and multiple buyers, or multiple sellers and one buyer, or multiple buyers and sellers,
- type of interaction, i.e. whether the interaction among the entities are private and committed bids, like in Sealed-bid auction, or publicly available to all involved entities, like in English and Dutch auction,
- number of negotiation rounds, i.e. whether the entities can improve their bids by counter-proposing, like in bargaining model, or it is one round negotiation, like in bidding,
- negotiation protocol used, i.e. the set of messages that can be exchanged and the different states that the involved entities can be during the negotiation process. Different protocols have been proposed, like Contract-Net protocol, English, Dutch, and Sealed-bid auction protocols,
- message description language, i.e. the language used for the description of messages exchanged among the involved entities, like KQML, FIPA-ACL for intelligent mobile agents, or any other proprietary one,
- ontology used, i.e. the way to categorise objects and entities so that they are semantically meaningful to software modules, like knowledge Interchange Format (KIF), XML, etc.
- strategy used during the negotiation for the counter proposals and decision making.

The most simple and utilised negotiation models proposed so far are the bidding, bargaining and auctioning methods.

In the bidding approach, the buyer specifies the product or service that wants to buy and generates a Call For Proposals (CFPs) related to that request. Potential sellers, check the CFP, generate corresponding proposals, and send them privately to the buyer. After all the proposals have been received or a timeout has passed, the buyer selects the best proposal based on some selection criteria, i.e. applies it own strategy. No counter proposals can be resubmitted. This type of negotiation model resembles to some extent the Sealed-bid where the same steps occurred. The negotiation protocol used in that case is the Contract-Net specified by Davis and Smith (Davis 80).

The bargaining model is similar to the bidding one except that there is not only one negotiation round but several ones until a selection can be done. During this process, the different sellers improve their proposals based on different strategies in order to be selected by the seller. The

protocol used in that case is the iterated Contract Net protocol (Davis 80, Kraus 98), i.e. modified versions of the original version of the Contract Net protocol. The bidding and bargaining model has been applied in both single, as well as, in multi issued negotiations. However, most of the prototypes and commercial systems developed so far provide only single issue negotiations.

In the auction model, a seller starts the negotiation process and requests bids or proposals from potential buyers. A potential buyer generates a proposal and announces it publicly to the whole group of buyers. The seller usually continues the negotiation process until certain criteria will be satisfied. Different models have been proposed so far, like the English and the Dutch auction. In the case of English auction model, the winner buyer is the remaining participant bidding the highest price. In the Dutch model, the price at which an item is offered for sale starts from a high level and declines steadily until one of the buyers stops the clock and buys the good at that price. The auction model has been applied only in single issued negotiations. However, there is a clear need to extend the model for multi issued negotiations, e.g. for procurement auctions (Bichler 98, Wurman 99).

Although negotiations have the purpose of restricting the possible courses of negotiation, they must obviously leave the alternatives for participating parties to choose from. In choosing between or proposing protocol-compliant alternatives, each participant follows its own negotiation strategy, which is normally not disclosed to other parties (Beam 96). Thus, in order to enable automated negotiations using autonomous entities, it is necessary to equip each entity with a formalised strategy to compute actions and offers corresponding to the role it takes in the negotiation. In general, two major schools of thought have been proposed and deployed so far, namely the analytical and the evolutionary approach.

In the analytical approach, the negotiation participant should be initially created with its complete set of strategies in place. In other words, the participant should have a large memory containing detailed instructions for each possible situation. The complete set of instructions, that determine the behaviour of the entity during negotiation, are based on static mathematical models and equations. Several analytical strategies have been proposed so far. Sandholm (Sandholm 82) discuss a so-called self-interest agent to design an optimised evaluation/decision function and suggests several elements that should be considered in negotiation: commitment level, local deliberation, and linking negotiation elements. Using ideas borrowed from game theory, Zlotkin (Zlotkin 97) treat negotiation as a type of interaction among distributed systems. In order to make the overall system more efficient, interaction rules, called negotiation mechanisms, are followed by each component system. Koistinen () uses a service constraint satisfaction technique and a worth-based evaluation function to determine the final deal in a quality-of-service negotiation. Guttman and Maes have created Kasbah (Maes 94, Guttman 98), a marketplace for negotiating the purchase and sale of goods using intelligent software agents. The agents, in their words, are “not tremendously smart”, nor do the agents use any machine learning technique or AI techniques, nor do agents attempt to encompass abstractions, such as user goals or preferences. Rather, the Kasbah software agents receive their complete strategies through the Web from the users, who specify the way in which the acceptable price can change over time, and retain final control over the agents at all times.

In the evolutionary approach, the negotiation participants should be able to learn. Rather than having a large memory, they should have the ability to acquire experience from previous negotiations they have conducted. The evolutionary approach makes use of very dynamic computing techniques, which are based on evolution principles such as selection, recombination,

and mutation. With evolutionary approaches, the learning effect is generally greater and also has a different dimension, since not only the data basis can evolve, but also the algorithms operating on these data themselves. Thus, evolutionary strategies are principally much more creative and self-adaptable than those based on analytical models. However, there only exist a few implementations of simple, data oriented, evolutionary negotiation strategies (Oliver 96). The main disadvantage of the evolutionary approach is that the resulting mechanisms always need certain initial phase to adapt so they are not immediately ready for effective operation. On the contrary, Oliver shows that any pre-programmed negotiation strategy will not be effective in real negotiation cases and shows that a system of artificial adaptive agents using a genetic algorithm can learn strategies that enable the system to effectively participate in business negotiations. However, Beam et. al. point out that genetic programming requires too many trials to obtain the good negotiation strategies. Zeng and Sycara (Zeng 96) present Bazaar, an experimental system for updating negotiation offers between two intelligent agents during bilateral negotiations. It explicitly models negotiation as a sequence decision-making task and user Bayesian probability as the underlying learning mechanism (Zeng 96). This technique demonstrates that although the computing model is static, a learning effect can be achieved by using some knowledge base that is updated dynamically during negotiation, so that every negotiation can take a different course.

Several prototypes of agent-based marketplaces have been developed and proposed the last years. PersonalLogic, Firefly (Firefly homepage), and Tete-a-Tete are agent based shopping assistants that help customers to narrow down the products that best meet their needs by guiding them through a large feature space. Andersen Consulting's BargainFinder (BargainFinder homepage) and Jango (Jango homepage) were the first shopping agents for online price comparisons. All of these systems do not provide any type of negotiation feature. They only enable customers to find and assess products and services that exist on different merchant sites. Agent-based marketplace systems with extra negotiation features based on auction model are actually the commercial sites of OnSale (Onsale homepage), Ebay (Ebay homepage), Cathay Pacifc, and Koll-Dove. However, the negotiation process is fully human-driven, i.e. the user needs to make the decisions and determine the strategy that he should follow. More advanced auction-based marketplaces are the Kasbah system from MIT and the AuctionBot from Michigan University. For a extensive analysis on these systems, the interesting reader should refer to (Maes 99, Beam 96).

3.2.6.2 Virtual Marketplaces in the Context of VE

Virtual marketplaces are a central part for electronic commerce transactions and especially for dynamic VEs. Virtual marketplaces provide third party matchmaking services that enable service providers and service users to find each other (Filos 00, Camarinha-Matos 99). Though the main objective of virtual marketplaces is on provision of matchmaking services, several additional services can be considered, like negotiation.

In automated matchmaking and negotiation process the involved entities should be autonomous, software modules that can locate potential suppliers and start negotiate without human intervention by exchanging messages that follow specific communication and negotiation protocols (Maes 94, Wurman 99). Intelligent mobile agents seem to be an ideal technical solution for such kind of problems due to the autonomy, adaptability, and learning characteristics that they reveal (Sierra 97). Actually, the mobile intelligent agent area has been

emerged as a technology to solve such kind of problems and has been ever since deployed in different other scientific areas (Maes 94).

The above described virtual marketplaces concepts and systems, especially the ones that are focusing on the negotiation aspects, do not clarify the way that different autonomous negotiation agents initially find them selves (Wurman 99). Existing matchmaking services, like the ones that previously described, can be deployed in that case. However, how the integration and deployment of these services is done is not clearly provided in most of the current proposals. In a multi-agent environment, where autonomous agents communicate with other agents by exchanging messages, the matchmaking services should be provided by dedicated agents that wrap the functionality of standard component or services and provide generic ontologies and standard messages for interaction with other agents (Guttman 98). Additionally, most of the current approaches are not taking under consideration the emerging agent communication standards, like FIPA, FIPA-ACL, and FIPA protocols, and how existing standard matchmaking services can be used in a multi-agent environment (Ouzounis 98e). This is an issue that current agent-based virtual marketplace systems do not extensively address.

Furthermore, the above described negotiation approaches, techniques and models have basically concentrated in the area of business to consumer and consumer to consumer electronic commerce and they are not addressing the needs of business to business marketplaces and especially, the needs of dynamic VEs (Filos and Ouzounis 00). Though some of the above techniques can be extended for the dynamic selection of partners in VEs, this area is considered new and further research is needed. Certain key issues like the agent communication language, the ontology, the negotiation protocol, and the internal strategy need to be clarified and extended for the case of dynamic VEs. Existing results of other research areas in negotiation, like consumer to consumer and business to consumer negotiation methods, can be deployed and extended (Beam 96, Guttman 98, Wurman 99).

Though different negotiation models have been proposed so far, not all of them are adequate for dynamic VEs. In that case, one business domain called the business process requestor, that would like to find another domain that can provide a specific business process, conducts the virtual marketplace and locates all the potential partners called the business process providers. Then, the business process requestor starts a negotiation process by issuing a CFP message. The potential business process providers respond with different proposals and the negotiation process continues accordingly. From this scenario, it is obvious that the bidding and bargaining model are the most favourable ones. In the auction models, not the requestors but the providers start the negotiation process and different requestors are responding with proposals. The existing, publicly available, research activities in the area of dynamic VEs are not covering this issue using open standard agent standards and techniques (Ouzounis 99b, Zarli 99).

Another interesting area that has been not covered significantly is the combination of negotiation interaction with the mobility aspect of agents (Choy 99). All of the above systems consider only persistent autonomous agents that interact with the exchange of messages. However, the benefits or drawbacks that mobile agents bring to the negotiation process are an unexplored field (Ciacarini 98). This is mainly because the intelligent autonomous agents, that take part in negotiation process, are rather "big" software entities and the migration of them across different physical location will probably take more time and will waste more network resource than to send simple FIPA-ACL messages. However, this is a quantitative remark that has not been proved yet nor certain alternatives heuristic mechanisms have been proposed.

Additionally, the usage of negotiation mechanisms during business process execution and management for the dynamic selection of partners is an very active research field (Tombros 99, Geppert 98). Most of the current workflow management systems use a static and pre-defined mechanism to relate business processes with different business domains. The deployment of negotiation mechanisms for the dynamic and automated selection of business process providers during process execution is a key issue for dynamic VEs (Grefen 98). This means that certain techniques for the integration of workflow management systems with autonomous, intelligent, agents and virtual marketplaces is an unexplored scientific area. Critical issues that should be taken under consideration are the integration of workflow management systems with negotiation agents, the suitable communication mechanisms, protocols, and ontologies used among the negotiation agents, the usage of virtual marketplaces by the negotiation agents, and the execution and management of inter-domain business processes (Ouzounis 99b, Filos and Ouzounis 00). Enabling the dynamic selection of business process providers based on automated negotiation is a key requirement for dynamic VEs and a significant improvement for inter-domain workflow management systems. As has been already stated, current workflow management systems and standards do not effectively address or solve these issues.

Finally, most of the above presented negotiation systems and models are concentrating on the provision of a coherent and generic strategy for automated negotiations from theoretical point of view and not on the provision of the communication and negotiation protocols and ontologies based on emerging agent standards (Sierra 97). Actually, the above presented systems, though they claim that they deploy intelligent agent concepts, they are not using emerging agent communication languages like FIPA-ACL, open flexible ontologies based on XML, and standard negotiation protocols like FIPA-Contract Net.

Autonomous intelligent agents that automate the process of matchmaking and negotiation for dynamic VEs are a very important and rational technological choice. Existing approaches and technologies for the matchmaking and negotiation process can be investigated, adopted, or extended for deployment in the case of dynamic VEs.

3.3 Limitations of existing Technologies in the context of VEs

A rapidly increasing number of projects and R&D activities worldwide are addressing different technical and business aspects of virtual enterprise technologies and infrastructures. Several technologies have been proposed so far, like Electronic Document Interchange (EDI), Distributed Component-based Business Systems (DCBS), Messaging Systems (MS), Workflow Management Systems (WFMS), Intelligent Mobile Agents (IMA), and Virtual Marketplaces (VMP) and Negotiation. In this section, a summary of the limitations that these systems and technologies have in relation to the dynamic VE requirements are discussed and presented.

In the case of EDI, the enterprise systems at the different partner organisations do not need to directly tightly couple their internal systems. Instead, all interactions between inter-domain business processes are accomplished via standard document exchange and message passing mechanisms. Though this method is very suitable for autonomous, asynchronous, and loosely coupled execution of shared business process across different domains, the currently provided format and syntax of EDI messages is static and rather limited (Gibon 99). Due to the fact that the scope and context of EDI documents is relatively restricted to a well-defined set of E-commerce transactions, it is difficult to use EDI as the basis for general purpose, inter-domain

business process execution and management approach (Lomet 93). The EDI standards provide initial definitions of common business documents, but historically, these have been inadequate for actual use. To address this issue, the EDI standard organisations, like EDIFACT and ANSI X.12, have undertaken an effort to standardise sets of documents for various industries and business sectors. Using these industry specific document definitions, the customisation required per business relationship can be reduced, though in general, per-relationship integration and customisation work is still required. Given the set of tradeoffs involved in the usage and deployment of EDI, it is best suited for long-term and stable business relationships between organisations that can make significant investments to support their relationship (Lee 98, Srinivasan 93). Business processes that do not related to electronic commerce, such as supply chain optimisation or product design, are best done outside the EDI context. In general, each new EDI relationship requires new customisation and integration work. These relationships are thus not entered into easily and return on EDI investment is gained over long periods of time and not over short-term relationships (Bolcer 99, Doz 98).

Distributed Component based Business Systems gained momentum in the R&D community due to the simplicity, ease of integration and deployment, high degree of distribution, standard underlying distributed protocols, like CORBA-IIOP and RMI, and middleware services. In principle, most of these systems are inadequate for usage in a dynamic VE environment mainly due to the fact that DCBS assume a tight coupling model (Zarli 99, Tombros 00). Backend systems and clients integrate with the distributed framework using the APIs and object models exposed by the underlying levels of the architecture. While clients are insulated from the APIs of the backend systems, they are tightly bound to the provided APIs (Orfali 96, Spinosa 98). This design choice has two implications. First, by using object binding as the interaction technique, DCBS applications must be adopted at once by all participants in the cross-organization relationship. Upgrades to backend systems, the component framework, and the business application, must be coordinated across all participants. Second, because of the tight binding, security issues are a major factor. Objects running in the business components-applications at one company must be able to communicate directly with objects running in the same component model at a partner company. This poses a significant barrier to adoption in cross-organization environments (Redlich 98). Additionally, the DCBS frameworks do not provide a complete solution, but instead serve as the starting point for developers to build applications (Carr 96, Nissen 99). By building on the framework, developers can more quickly complete applications and leverage the code in the framework that takes care of many of the mechanical details needed for a successful distributed application. Finally, these choices make the DCBS frameworks most appropriate for deployment inside a single company that needs to link multiple distributed divisions or sites. Such a company can plan for a unified deployment and can afford the integration and customisation work. Indeed the majority of DCBS deployments are taking place inside single enterprises and for intra-domain applications (Sheth 98).

Messaging systems is an alternative technology option for dynamic VEs. The main strengths of this approach are the differentiation among the interface of the services and the corresponding modules that provide these services. This means that messaging systems are not based on the static and tight couple model of components, like in DCBS, and they do not require compatible middleware services, like EJBs (Filos 00, Stricker 00). Additionally, messaging systems hide all the complexities of the underlying components or systems and enable true autonomous, asynchronous, and loosely coupled relationships among different business domains. This is very important issue for dynamic VEs, where the management of business processes is done by different domains that have been selected during the process execution (Reichert 98). However,

messaging systems do have certain problems. One of the key problems is the different proposals for a message specification language, i.e. envelope and content of the message. The previously presented protocols actually specify their own envelope in XML and their own underlying content description approach. Another critical problem is the lack of generic messaging standards. The incompatibilities among different systems increase the problem and make the integration of business processes among different domains difficult (Georgakopoulos 98). In addition to that, certain problems do exist on the transport protocol deployed for the exchange of messages. The existing protocols and frameworks specify different transport protocol like CORBA-IIOP, TCP/IP, or HTTP. Finally, one of the biggest problems in this area is the specification of certain ontologies for different business sectors (Spinosa 98). Standard ontologies will enable the rapid integration and deployment of messaging systems for dynamic VEs. In order to achieve this, standard, open, content description meta-languages are needed. XML seems to be the preferred option that will solve the problem (Ouzounis 99a). However, XML is a newly adopted standard and it will take some time to establish acceptance before such activities will start. In general, messaging systems pose certain benefits over existing DCBS in the context of dynamic VEs due to the asynchronous and loosely coupled approach, the global ontologies, and the independency among the interfaces of the components and the components.

Intelligent Mobile Agents provide significant benefits in relation to traditional distributed object oriented approaches. Some of the major benefits emerged from the usage of intelligent, mobile agents are autonomy and flexibility, due to the communication and co-operation models among agents, scalability, due to the migration capabilities, adaptability, due to the intelligent behaviour, and integration with existing technologies, due to the object oriented concepts used to implement agent platforms and agents (Krause 96 and 97, Lecihsering 98). Intelligent agents seem to provide certain benefits for the development of dynamic VEs. Intelligent mobile agents can be used in different ways to solve effectively VE problems (Kraus 98). One way is to use an agent based business process management system that control and co-ordinate in a distributed, autonomous, and flexible way the execution of VE business processes (Bellifernine 99). Another way is to use agents for the dynamic selection of partners and the negotiation phase among different VE partners. The autonomy and intelligent characteristics of agents can significantly improve and automate the selection and negotiation process (Borghoff 97). Agents can also be used to manage and co-ordinate the provision of matchmaking services, like virtual marketplace services. Additionally, agents can migrate to different physical locations where business logic exists and deploy business services by reducing the network load and traffic (Choy 99).

Although intelligent, mobile agents seem to a very good candidate for the development of dynamic VE systems, they do have some problems as well. One of the key issues is the requirement for a mobile agent platform for the provision of agent life-cycle and migration management services (Martesson 98). Certain mobile, intelligent agent platforms have been developed so far and a lot of standardisation activities have been emerged, like OMG-MASIF and FIPA, which try to deal with these problems. Another issue is the lack of standard Agent Communication Languages (ACLs). In that respect, FIPA-ACL (FIPA 98), and its predecessor KQML (Finin 95), is a proposed standard that has been used by many Multi Agent System (MAS) prototypes.

In addition to the benefits coming from the usage of agent concepts, agents seem to combine all the benefits offered by the messaging systems and DCBS. Agents communicate by exchanging ACL messages in an asynchronous and loosely coupled way using underlying messaging systems (Breugst 98). However, agents deploy the concept of ontologies that make them more flexible and autonomous (Kraus 98). Agents are deployed within a distributed object oriented

platform, like Corba or Java Framework, and thus can access any type of standard business component. Additionally, agents can migrate to the physical location of business components preserving the network resources. Finally, agents have the ability to execute and co-ordinate complex business processes, in a similar way like workflow management systems. However, the execution of the process is not controlled in a centralised way by the workflow engine, but the agents themselves are co-operating in a flexible and autonomous way.

Workflow management systems are used to specify, execute, manage, co-ordinate, and streamline business processes. Workflow management systems feature a set of good attributes for deployment within the context of a VE (Grefen 99). The shared business processes among the VE members can be described by deploying a business process specification language (Hoffner 98, Georgakopoulos 98). For example a shared process can start in one domain and then, can be continued in another domain-partner, by utilising remotely a sub-process (Miller 98). The workflow management system will undertake the responsibility to execute and manage the execution of the shared business process in a distributed and systematic way (Geppert 98).

Although traditional WFMS systems have significant benefits, they do have certain drawbacks in relation to the VE concept. One of the main issues is the limited autonomy and flexibility that they have (Miller 98). WFMS execute upon well-defined business processes specified in a specification language that the system understands. So far there are no certain extensions to the existing business process specification languages towards the direction of cross-organisational business processes (Bolcer 99). Additionally, remote invocation of business processes, provided by different business domains, should follow access control, authorisation and contract checks. Current workflow systems do not provide such mechanisms (Klingemann 99). Finally, in current prototypical workflow systems, shared business processes are being specified statically in relation to remote processes, i.e. the VE partners that will provide them are being specified statically. This approach is suitable for static VEs and not for dynamic ones, where the partners, that can provide parts of the shared business process, are not known in advance. On the contrary, the remote domains can be selected dynamically, after negotiation and during the business process execution (Ouzounis 98d, Tombros 00). This means that for the same business process specification different instances might exist and thus, different constellations of VEs. For every instance a set of different partners might be selected according to the needs and requirements of the various partners.

Current proposed standards are not directly dealing with cross-organisational business process execution and management (Bolcer 99). Critical open issues, like inter-domain workflow execution and management, business process specification languages for inter-domain business processes, and dynamic selection of workflow providers during process execution are not discussed at all. Additionally, as has been explained in the DCBS section, the deployment of tight-couple communication mechanisms, like CORBA, as a mechanism for autonomous inter-domain business process execution and management is rather problematic and in general, inflexible (Miller 98). Therefore, it is anticipated that a message-based approach with corresponding XML message requests and responses would have been better since the degree of autonomy and flexibility is increased (Ouzounis 98c, Geppert 98, Grefen 98).

Agent-based workflow management systems seem to be in position to solve some of these problems. For example the execution of the shared business processes can be controlled by agents that can deploy them remotely or migrate to the remote business domain, invoke the required business process by identifying themselves, and return back to continue the normal execution of the process (Ouzounis 98b, Tombros 99). The physical location of the remote

domain should not be known in advance, but can be found after selection and negotiation with potential providers in a virtual marketplace. The remote domain can also authenticate and authorise the requesting agents based on electronic contracts that have been established during the negotiation phase. However, most of the issues related to agent-based workflow management systems, cross-organisational business process execution, and dynamic selection of partners are under investigation and certain solutions and concepts are required (Tombros 00, Ouzounis 99b, Georgakopoulos 98).

Finally, virtual marketplaces are a central part for dynamic VEs because they provide dynamicity, flexibility, and evolution to the VE models (Camarinha-Matos 99). Though the main objective of virtual marketplaces is on provision of matchmaking services, several additional services can be considered like negotiation. Intelligent mobile agents seem to be an ideal technical solution for such kind of problems due to the autonomy, adaptability, and learning characteristics that they reveal (Magedanz 99). The previous described virtual marketplaces concepts and systems, especially the ones that are focusing on the negotiation aspects, do not clarify the way that different autonomous negotiation agents initially find them selves. Existing matchmaking services, like the ones that previously described, can be deployed in that case. However, how the integration and deployment of these services can be done is not clearly provided in most of the current proposals (Zarli 99, Hoffner 98). In reality, most of the approaches are not taking under consideration the emerging agent standards like FIPA-ACL and FIPA protocols, and how existing standard matchmaking services can be used in a multi-agent environment. Furthermore, the above described negotiation approaches, techniques, and models have basically concentrated in the area of business to consumer and consumer to consumer electronic commerce and they are not addressing the needs of business to business marketplaces and especially the needs of dynamic VEs. Although some of the above techniques can be extended for the dynamic selection of partners in VEs based on service templates, this area is considered new and further research is needed (Guttman 98). Certain key issues like the agent communication language, the ontology, the negotiation protocol, and the internal strategy need to be clarified and extended for the case of dynamic VEs (Wurman 99). Additionally, though different negotiation models have been proposed so far, not all of them are adequate for dynamic VEs. The bidding model and bargaining model are the most favourable ones in comparison with the auction models (Bichler 98). In general, autonomous intelligent agents that automate the process of matchmaking and negotiation for dynamic VEs is a very important area of work that needs significant research and development work (Zarli 99, Filos 00, Ciacarnini 98).

Additionally, the usage of negotiation mechanisms during business process execution and management for the dynamic selection of process providers is also a very active research field. Most of the current workflow management systems use a static and pre-defined mechanism to relate business processes with different business domains (Tombros 99, Grefen 99). The deployment of negotiation mechanisms for the dynamic and automated selection of business process providers during process execution is a key issue for dynamic VEs. This means that certain techniques for the integration of workflow management systems with autonomous, intelligent agents and virtual marketplaces are required. Critical issues that should be taken under consideration are the integration of workflow management systems with negotiation agents, the suitable communication mechanisms, protocols, and ontologies used among the negotiation agents, the usage of virtual marketplaces by the negotiation agents, and the execution and management of inter-domain business processes. Enabling the dynamic selection of business process providers based on automated negotiation is a key requirement for dynamic VEs and a significant improvement for inter-domain workflow management systems. As has been already

stated, current workflow management systems and standards do not effectively address or solve these issues.

In Table 2, a summary of the limitations that the existing technologies have in relation to the dynamic VE concepts is presented. In principle, The acceptance of intelligent and mobile agents, as an implementation and communication paradigm, the extra capabilities that they offer, like mobility, autonomy, and intelligence in conjunction with emerging state of the art agent platforms and standards, like FIPA and OMG-MASIF, and standard distributed platforms, like Corba and Java-RMI, flexible content description languages for globally specified ontologies, like XML, emerging XML-based workflow standards, like WfMC and SWAP, and platform independent programming language, like Java, can provide the basis for the new generation of open, flexible, autonomous, and distributed systems for the management and execution of shared business processes in the context of dynamic VEs.

	EDI	**DCBS**	**MS**	**WFMS**	**IMA**
Communication Model and Synchronous vs Asynchronous Communication	Message exchanges, asynchronous	Tight couple, basically synchronous	Message exchanges, asynchronous	Both Message and Tight couple, both synchronous and asynchronous	Both Message and Tight couple
Transport Protocol	VPNs, recently secure TCP/IP	Corba-IIOP, Java-RMI	Corba-IIOP, Java-RMI, TCP/IP, recently Http	Corba-IIOP, Java-RMI, TCP/IP,	Standards: Corba-IIOP and FIPA-ACC, non-standard Java-RMI
Autonomy	High, limited set of messages	Medium to low, dependency on standard interfaces	High, generic set of messages	Medium, workflow management specific messages	High, general messages related to ontologies
Mobility	No	No	No	No	Yes
Abstraction Level	EDI-message	Object	Message	Message/Object	Agent
Application Specific	Yes, E-commerce	No	No	No	No
Flexibility	No, due to pre-specified messages	Yes but on intra-domain level	Yes, on both intra- and inter-domain	Yes, but mostly on intra-domain level	Yes, but mostly on the intra-domain level
Customisation and Integration	Rather high	Medium	Medium	Medium	Medium
Openess and Standards	Low due to different EDI specification and standards	Medium due to the dependencies on the interfaces, the programming language and the tight couple model	Medium due to the lack of standards in Message Descriptions and deployment of different transport protocols	Medium due to the lack of well-accepted standards for Internet-based workflow systems and business process definition languages	Medium due to the lack of well-accepted standard in the Mobile Agent Platforms and differences in Agent Communication Languages

Limitations for deployment in the Context if VEs	restricted to E-commerce, impossible to extend, high integration and customisation costs, no access control and authorisation mechanisms	synchronous communication model, tight couple, security and access control problems, enforcement of certain technologies	interoperability in message specification, security and authorisation problems, business oriented ontologies, enforcement of certain technologies	lack of business process definition languages for shared processes, inter-domain workflow management, access control and contracting issues,	enforcement of a mobile agent platform, lack of standard communication languages, lack of certain ontologies for inter-domain business process execution
Suitable for	Static VEs	Static VEs	As basis for both static and dynamic VEs	Static VEs. If they address inter-domain workflow management are suitable for DVEs	Dynamic VEs if they address workflow and virtual marketplace concepts

Table 2: Limitation of Existing Technologies

The main objective of this thesis is to analyse, design, develop, test, validate and assess a platform for the management of dynamic virtual enterprises that will support the whole life cycle model (see 2.4 section) and will be based on standard FIPA intelligent mobile agent concepts, emerging agent-based workflow management concepts for cross-organisational business process execution and management, and virtual marketplaces with emphasis on OMG Trader integration and automated negotiation for dynamic partner selection.

More particularly, this thesis will define, develop and validate the following entities:

- a XML-based virtual marketplace ontology for business process registration, management of offers, and dynamic partner selection in virtual marketplaces,
- an agent-based FIPA compliant virtual marketplace and integration with the standard OMG Trader,
- a negotiation ontology and protocol for dynamic partner selection based on FIPA-Contract Net protocol,
- an XML-based business process definition language for the specification of business processes in the context of dynamic virtual enterprises and a business process repository for the storage of business processes,
- a distributed, agent-based, FIPA compliant, workflow management system for the execution and management of shared business processes across different organisational boundaries,
- an XML-based intra- and inter-domain ontology for cross-organisational agent-based business process execution and management,
- provision of shared business processes to the web by integrating the agent-based workflow management system with standard web integration technologies,

In the following chapters the above-mentioned entities are further analysed, designed, tested and validated.

3.4 Summary

This chapter presents an exhaustive analysis of the state of the art in both technical and functional issues. The chapter starts with an analytical description and assessment of the current projects and academic and scientific results in the area of VEs in relation to the requirements presented in the previous chapter. The presented analysis concludes with a set of open technical issues that needs to be addressed and solved. In the sequel, an assessment of the technologies proposed and deployed so far in the area of VEs is given. The traditional Electronic Document Interchange (EDI), the Distributed Component Business Frameworks (DCBS), the Messaging Systems (MS), the emerging Intelligent Mobile Agents (IMA), the Workflow Management Systems (WFMS), and Virtual Marketplaces and Negotiation (VMP) are assessed and evaluated. For all these technologies, an extensive individual assessment regarding their applicability to the dynamic VE concepts and requirements is done. Based on the academic state of the art and the assessment of the proposed and deployed technologies, the problem statement follows, which is an agent-based platform for the management of dynamic Virtual Enterprises. In the problem statement, the main objectives of the thesis in relation to the most adequate technologies are described.

Chapter 4: An Agent-based Platform for the Management of Dynamic VEs

4.1 Introduction

A platform for the management of dynamic VEs should provide all the basic services required for both phases of the life-cycle model (see section 2.4), namely the Business Process Specification and Registration and Business Process Management.

During the Business Process Specification and Registration Phase a VE candidate partner *specifies* his local and remote business processes. The specification of local business processes is done using a *business process definition language*. For every business process, the input parameters, the output parameters, the sub-processes, the tasks and the conditions among the sub-processes and tasks are being specified. Additionally, every sub-process is specified as local or remote process. Local processes are the processes that can be fully provided by this domain while remote processes are the processes that can be provided only by remote domains. Furthermore, for every specified task the associated business objects, that will be deployed, are also specified. In this way, autonomous agents can easily deploy legacy services provided by existing distributed objects that physically located in different network locations within the domain.

In the sequel, every administrative domain that would like to participate in dynamic VE relationships registers its processes in the virtual marketpla ce. The business process *registration* performed by deploying the existing service types provided by the marketplace. If there is no associated service type for a particular process, a new one is being created by possibly inheriting existing service types. This process can be done either automatically or manually through the virtual marketplace administrator. During the registration process, certain values for certain attributes related to the service type, like location, quantity, etc., are specified. These attributes are usually related to the provision of the process to remote administrative domains. In addition

to the service provision related attributes, a set of attributes that will influence the negotiation process is also specified e.g. price. These attributes might include the low price that can be negotiated upon, the maximum quantity that can be offered, the best and worst delivery dates, etc.

During the Business Process Management Phase a VE partner provides business processes to customers or other VE partners by deploying the dynamic model of the virtual marketplace. Initially, when a customer requests a business process by a VE Representative a process instance is being created, i.e. the process description for this process is retrieved, interpreted, and the execution of the process is started. The instantiation, interpretation, and execution of the business process are done by a set of autonomous agents that co-operate to provide the requested business process. The initial request of the customer for a business process execution is served from these autonomous agents. The co-ordination of the autonomous agents during the execution of a business process is performed by deploying the *intra-domain ontology*. The intra-domain ontology is the set of messages that the different agents exchange during the execution and management of the local business processes. If one of the sub-processes of the main process has been specified as remote, then a suitable partner for this sub-process should be located. For that reason, the virtual marketplace is conducted and several potential VE candidate partners are being selected. Upon request, the virtual marketplace informs the initial domain about all the registered domains that can provide this sub-process, i.e. all the potential VE candidate partners. In the sequel, the negotiation process is initiated by conducting all the VE candidate partners. The negotiation process is performed by using a *specialised negotiation protocol* and *ontology*. The result of this negotiation process is the selection of the best VE candidate domain that satisfies certain classification criteria. This agreement is being described in terms of a "technical", electronic *contract* that regulates the agreement.

As soon as a VE partner has been selected for a particular remote process, the VE representative conducts the selected VE partner domain and requests the execution of the business process by referring to the contract id that has been signed during the negotiation process. The VE partner domain checks the list of existing contracts and starts the execution of the requested process if a legitimate contract has been found.

During the execution of the main process, the customer can manage the execution of the main business process. The main operations that can be performed are suspension, resumption, or termination of the execution of the process. Every customer request is served initially from the VE representative domain. All the agents related to the execution of this business process instance are suspended, resumed, or terminated accordingly. In addition to that, all the remote processes that have been previously requested should also be suspended, resumed or terminated. Therefore, similar requests are issued and sent from the VE representative to the corresponding VE partners. Whenever a request to suspend, resume or terminate an existing local business process arrives, the agents check the contract id and serve it accordingly. In that way, unauthorised requests for process suspension, resumption, or termination are not served. The co-ordination among the different autonomous agents during the execution of remote business processes is performed by deploying the *inter-domain ontology*. The inter-domain ontology is actually the set of messages that the agents exchange during the execution and management of remote business processes. Additionally, the customer can always ask about the current status of the business process. In a similar way, the VE representative requests from all involved autonomous agents associated with this process, local or remote, to declare their current status. When a process finally completes its operation, the VE representative partner informs the

customer by posting to him the output results of the process and other statistical information like the time of completion.

Furthermore, if during the execution of a process a fatal problem occurs, then the corresponding agent, responsible for this process instance, informs the system that the execution of this process can not be continued and thus, the agent needs to abort himself. The agent informs on-demand, either the customer or the associated VE partner about this event and stops the execution of the business process.

In the following sections the analysis approach used for designing a complex system like this, the business domains and roles involved, the operations that they perform, a layered architecture of the system, and finally details about the agents and the services they offer are presented and further discussed.

4.2 Analysis and Specification Approach

In order to analyse and specify a complex system, like the one that has been previously described, a consistent and coherent methodology is needed. The approach that will be used for the analysis and design phase in this thesis is the Unified Modelling Language (UML) approach. UML is a standard, consistent methodology for analysing, specifying, and developing complex distributed systems. The methodology consists of the following phases (UML 98):

- **Business Domain Analysis and Specification**: In this phase the identification of the different administrative domains and the specification of the relationships that these domains have among each other is performed. In principle, an administrative domain has a relationship with another domain when a user, agent, or service deploys or communicates with a user, agent or service provided by another domain. Additionally, in each business domain the identification of the key human roles, the responsibilities that they have, and the basic operations that they perform is done. A human role normally deploys a service or an agent in a certain way by deploying certain operations provided by the agent or service. All the ways that a human user deploys one service constitute a use case. These requirements on the usage of an agent or service are the key requirements for the design of each individual entity that will be performed in the next phases,
- **Architecture Specification**: In this phase the specification of the basic layered architecture of the system, the sub-layers, and the underlying supporting middleware services within each layer is performed. Additionally, the services or the agents that will be deployed, as such, and the services or agents that will be further specified, designed, and developed is done. These agents or services are the key entities for further analysis and specification that will be performed in the next phase,
- **Agent Specification**: In this phase, every agent or service that has been identified in the previous phase, is being specified. The specification of the agent includes the external interface, i.e. the services provided to other agents, the internal architecture, and the internal entities of the agent in terms of UML class diagrams. Additionally, for all the internal components of an agent the specification of the interface and the relationships that they have is done. In the sequel, the external operations of an agent are specified in terms of UML sequence diagrams involving all the internal modules of the agent. These sequence diagrams determine the way that the internal modules of the agent are being deployed for the provision of the key operations of the agent.

In the following sections, the first 2 phases are presented analytically and certain decisions are being taken. The agent specification phase is primary concerned with the specification of individual agents involved into the system and will be provided analytically in subsequent chapters.

4.3 Business Domain Analysis and Specification

Business domain analysis and specification is concerned with the identification and specification of the different administrative domains, the relationships, the human roles involved and the responsibilities that they have. This phase consists of the following key steps:

- Business model and relationships specification,
- Role and responsibilities specification.

In the following sub-sections the previous mentioned steps are further discussed and analysed.

4.3.1 Business Model and Relationships

The business model is specified as a set of different administrative domains having specific relationships. The business model for the agent-based platform for the management of dynamic VE consists of the following administrative domains:

- **Customer**, is the domain that has subscribed to the services provided by the VE and is allowed to use them,
- **VE representative**, is the domain that represents the VE to the outside world, i.e. to the customers, and provides the VE services to them. The VE representative is the responsible domain where the end-users are logged in, deploy and manage the provided services, i.e. the business processes,
- **VE Candidate/Partner** is the domain that registers its service offerings into the marketplace and negotiates with other VE partners on-demand to establish business relationships. When a successful negotiation has been achieved between a VE partner and a VE candidate partner, then this domain acquires the status of the VE partner. The VE partner authorises and authenticates business process requests, based on the negotiated contracts, and executes and manages the business processes on behalf of other domains,
- **Virtual Marketplace**, is the domain that provides registration and selection services for VE candidate partners. This domain is responsible for administrating service types and service offers and for managing the daily operations of the marketplace.

The logical relationships among these domains are depicted in the following picture.

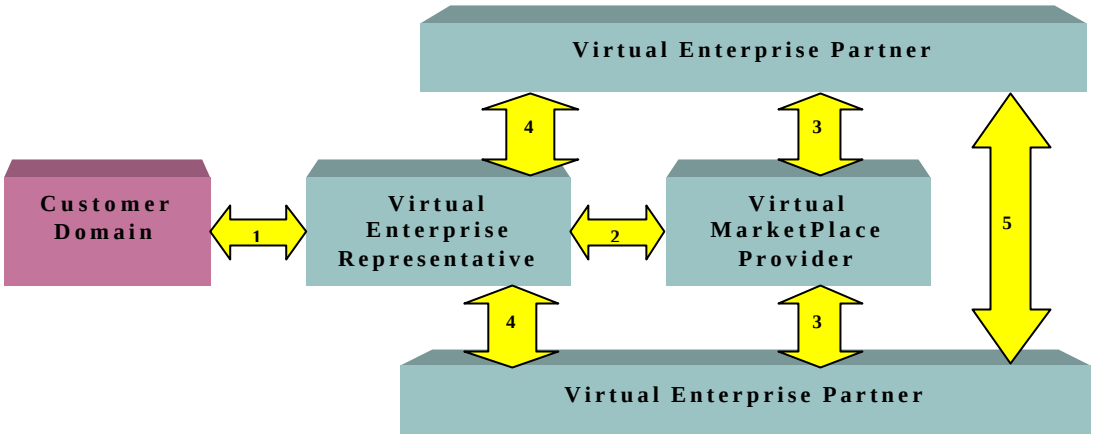

Figure 6: Business Model and Relationships

It should be noted that the VE representative domain and the VE Candidate/Partner provide similar services to each other and they deploy the same mechanisms. The main different is that the VE representative actually represents the VE to the external world, i.e. to the customers. Otherwise, the services, roles, responsibilities and internal components of both domains are identical from technical point of view. This means also that one domain can be VE representative for one VE and a normal VE partner for another VE, i.e. the roles of the VE representative and VE partner are symmetrical and independent of the underlying services.

Based on the previous business model, the following logical relationships among the domains can be specified:

- **Customer - VE representative (1):** the customer domain deploys all the provided services from the VE representative in a transparent way, i.e. the customer does not know the existence of the VE partners. The customer can log in into the system using a standard web browser and can start a business process, get a status report about a running processes, and manage existing processes, i.e. suspend, resume or terminate a process. Finally, when a running process completes, the result of the process is returned to the customer.
- **VE representative - Virtual marketplace (2):** the VE domain uses the marketplace to register local business processes and to search for potential VE candidate partners that can provide a service. More specifically, the VE representative can register, de-register or modify an existing business process offer stored in the virtual marketplaces. Additionally, the VE representative can search the virtual marketplace based on some constraints and get a list of potential VE candidate partners that can provide a specific service. The requirements imposed by this relationship are reflected into the specifications of the *virtual marketplace ontology*.
- **VE partner - Virtual marketplace (3):** the meaning of this relationship is the same like the previous one except the fact that the VE partner performs these operations. Additionally, in this relationship only the registration and administration of business process offers into the virtual marketplace is provided and not the searching operations. The requirements imposed by this relationship are reflected into the specifications of the *virtual marketplace ontology*.

- **VE representative – VE Candidate/Partner (4):** the VE representative starts a negotiation process with a VE candidate partner. This negotiation process results into an electronic contract that regulates the co-operation among the domains. As soon as the contract has been established, the VE representative can start the agreed remote process, resume, suspend, or terminate it upon request of a customer. The requirements imposed by this relationship are reflected into the specifications of the *inter-domain* and *negotiation ontology*.
- **VE Partner – VE Candidate/Partner (5):** the meaning of this relationship is the same like the previous one. In that case, the VE partner negotiates with one or more VE Candidate providers and selects one as VE partner. Then, the execution and management of remote business processes can be done. This relationships enables the dynamic creation of complex VEs where the partner outsource some of their business processes on-demand to other capable providers. The requirements imposed by this relationship are reflected into the specifications of the *inter-domain* and *negotiation ontology*.

It should be noted that the relationships 2 and 3 are similar in the sense that the technical realization and the required specification is the same. However, in the case of relationship 2, the domain that deploys this relationship is the VE Representative, while in the case of relationship 3, the domain that deploys this relationship is the VE Partner.

In the same way, the relationships 4 and 5 are also similar in the sense that the technical requirements imposed by them are the same. However, the difference is only semantically and it is related with the names and business position of the domains that deploy these relationships, namely the VE Representative and VE Partner.

4.3.2 Roles and Responsibilities

Having specifying the key business domains of the platform and the key relationships that they have, the individual human roles that exist in every domain can be specified.

In the VE representative and VE Candidate/Partner domain the role of the *Business Process Analyst* exists. This person is responsible for the specification of the business processes of this domain by deploying the business process definition language. The analyst also specifies which processes will be provided by this domain, i.e. local processes and which processes will be deployed remotely and dynamically by other partners, i.e. remote processes. Additionally, the analyst specifies the terms and conditions concerning the offering of local processes to potential partners. Based on these terms and conditions, the registration of local business processes to the virtual marketplace is done. The terms and conditions are actually logical constraints that relate process properties with certain min or max values. More specifically, this role performs the following operations:

- creation, modification, and deletion of business processes using the business process definition language,
- specification, modification, and deletion of certain terms and conditions related to the provision of local business processes to potential partners. These terms and conditions will be used during the negotiation process with potential partners,

In the Virtual Marketplace domain the role of the *administrator* exists. This person is responsible for specifying and managing the service types that have been created in the virtual marketplace. The main operations that this role performs are:

- creation, modification, list, and deletion of a service type,

In the Customer domain the role of the *end-user* exists. This person initially subscribes to the services provided by this domain and gains access to them. This person has no particular responsibility except to log in and use the provided services. The main operations that this role performs are:

- log into the system by using a standard web browser,
- initiation of a business process and monitor the status of an existing business process,
- suspension, resumption, or termination of a running business process provided by different VE partners.

Recalling the life-cycle model of dynamic VEs, the distribution of the above roles in each phase is the following:

- In Business Process specification and Registration phase, the Business Process Analysts for both the VE representative and the VE partner participates in the specification of the business processes, administration of the registration of local processes into the marketplace, and specification of the terms and conditions of the negotiation process for each local process.
- In Business Process management phase, no human role is involved. On the contrary, the different autonomous agents undertake the responsibility to execute business processes, to search for potential partners dynamically, to negotiate for the selection of the best partner, and to authorise the usage of processes based on the established electronic contracts. The only human role involved is the end-user, from the customer domain, that can query the status of a process, suspend, resume, or terminate a running process.

4.4 Architecture Specification

In general an architecture expresses a fundamental structure of the system under analysis and design. The architecture defines a set of functional components, sub-systems or modules described in terms of their behaviour and interfaces into which the system is divided. It defines also how these components interact or interconnect to fulfil the goals of the system. Thus, an architectural description is primarily concerned with the structure of the system provided by the specification of the functions and the responsibilities of the functional components. In principle, the term component includes functional components that can be either distributed objects or autonomous agents. This is strongly influenced by the object-oriented paradigm where data and behaviour are not separated entities. This is shown in the specifications of the different entities in the following sections and chapters by identifying specific operations. Taking these definitions into account, the architecture is described, according to the UML approach, with interfaces, operations, use case diagrams, and sequence diagrams. The interfaces can be defined either in Java programming language or in XML (XML 98, Harold 98).

By specifying and using a layered architecture for the development of a complex, distributed, autonomous system provides a number of benefits (Barry 98, Ceri 96):

- **Understanding of System Structure**. Architecture and architectural descriptions characterise a system's structure in terms of high-level computational elements and their interactions. That is, the architecture frames its design solution as a configuration of interacting components,
- **Rich abstractions for interaction**. Interactions between architectural components provide a rich vocabulary for system designers. It also separates the functionality of components from the concerns of interaction between them. This allows a modularisation of the system components that facilitates the evolution of the functionality of the each component,
- **Software development economics**. Software architectures support and facilitate the re-use of itself from system to system and of its sub-components whenever these are clearly defined and documented.

The layered architecture of the agent-based platform for the management of dynamic virtual enterprises consists of three respective layers. These are:

- **Agent-based Business Process Specification, Registration and Management System** and **Agent-based Virtual Marketplace System** that provide the basic operations for the specification of inter-domain business processes, the registration of them in the virtual marketplace, the selection and negotiation of partners, the access control and authorisation of process requests, and the execution and management of business processes.
- **Mobile Agent Platform (MAP) and Supporting Services** that provide the basic agent life-cycle services, migration services, messaging services, and access to services provided by the underlying distributed processing environment like XML parsers, legacy systems, etc.
- **Distributed Processing Environment (DPE)**: that supports the key operations for object life-cycle management and distributed services like Remote Method Invocation, access to persistent repositories, deployment of existing legacy systems, enabling services like vectors, etc. The distributed processing environment that will be deployed is that of the Java Framework with support of CORBA middleware services in order to enable interoperable access to distributed objects and components.

This architecture is presented in the following Figure 7.

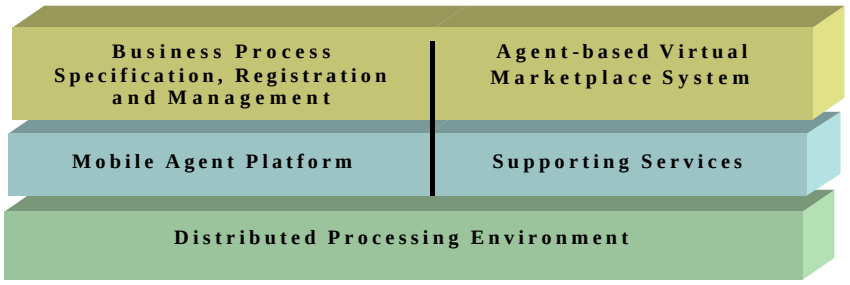

Figure 7: Overall System Reference Architecture

The last two layers are the basic infrastructure where the agent-based platform for the management of VE has been developed. The different services and components provided by these two layers will be directly used by the upper layer, i.e. from both the agent-based business process specification, registration and management system and the agent-based virtual marketplace.

Additionally, the first layer of the architecture is split into the business process specification, registration and management sub-layer and the virtual marketplace sub-layer. However, every administrative domain deploys this architecture in a rather different way. More specifically, the virtual marketplace domain uses only the agent-based virtual marketplace sub-layer, the VE representative and Candidate/Partner domain uses the business process specification, registration and management, while the customer domain deploys only a standard web browser for accessing the business processes. In both cases the two lowest layers deployed are the same, i.e. the Distributed Processing Environment and the Mobile Agent Platform and Supporting Services. In the following figure, the domain specific architecture is depicted.

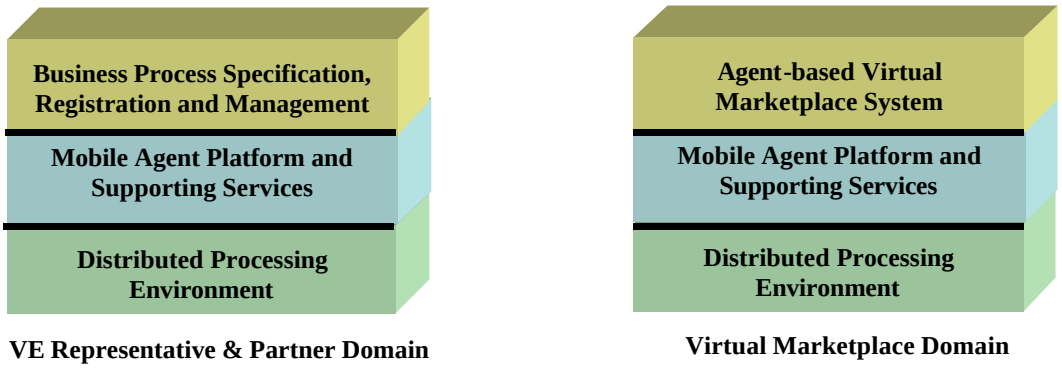

Figure 8: Business Domain Specific Reference Architecture

In the following sections each one of the following layers is analysed and more information regarding the specific components of each layer is provided. More emphasis is placed on the upper layer where the appropriate components will be specified and analysed.

4.4.1 Agent-based Virtual Marketplace

A Virtual Marketplace is a third party administrative domain that provides matchmaking services to the VE partners. The Virtual marketplace enables *VE Candidate Partners* to register and administer service offers in relation to certain service types and *VE Representatives* to search for potential partners that can provide particular business processes associated with existing service types.

Every registered business process in the virtual marketplace is associated with a service type. In general, service types describe in a consistent way the interface of business processes. For every service type, the name of the process and a set of named properties are specified. The name of the service type is the name of the business process, while the input and output parameters of the process are named properties of the service type. Additionally, extra properties, related with the negotiation process, are also included into the service type. For every property (name, value) pair is associated. Service types managed by the virtual marketplace administrator. The service type management includes creation, deletion, modification and retrieval of service types.

VE candidate partners that want to register their process offerings in the marketplace should always create a service offer in association with an existing service type and register it to the virtual marketplace. A service offer is actually an instance of a service type where certain properties have given certain values. Service offers managed individually by each domain in a

private manner. The management of service offers includes the registration of an offer, the withdrawal, and the modification of it.

Finally, VE representatives or partners that want to find suitable partners that can provide a particular process retrieve from the marketplace all the registered offers that satisfy certain constraints. The service offer retrieval management process actually includes the retrieval of offers that satisfy certain constraints.

Therefore, the basic services provided by the marketplace are service type management, service offer management and service offer retrieval management. Each one of these operations provided by individual, FIPA compliant, autonomous agents. More specifically, the:

- **Service Type Agent (STA)** is responsible for the management of service types and more specifically for the addition, removal, listing, and modification of a service type,
- **Service Offer Agent (SOA)** is responsible for the management of service offers and more specifically for the registration, withdrawal, description, and modification of a service offer,
- **Service Offer Retrieval Agent (SORA)** is responsible for the retrieval of offers associated with a service type based on some constraints.

Other administrative domains are using the service provided by the three virtual marketplace agents by exchanging messages. The messages are being described in FIPA-ACL format while the content of the message is specified in XML following the virtual marketplace ontology. Therefore, the virtual marketplace ontology is the set of FIPA ACL/XML requests and responses that autonomous agents can exchange with the virtual marketplace agents. The communication protocol used for this interaction is the FIPA compliant request-response protocol.

In addition to the above stated virtual marketplace agents, the following main non-agent components are specified:

- **Service Type Repository (STR)** responsible for the storage and management of services types in a persistent way,
- **Service Offer Repository (SOR)** responsible for the storage and management of offers associated with service types in a persistent way.

In the following chapters, the marketplace concept and marketplace agents are extensively analysed and full specifications of the agents and the key entities are provided.

4.4.2 Agent-based Business Process Specification, Registration and Management

The business process specification, registration and management layer provides the basic infrastructure for the specification, interpretation, execution, and management of business processes in the context of dynamic VE. This layer supports both, the business process specification phase, and the business process execution and management phase, i.e. the two key phases of the VE life-cycle model.

In the first phase, the business process analyst in every VE domain specifies the business processes by using the *Business Process Definition Language* (BPDL). The BPDL is an XML-based language enabling the specification of complex processes. The language has been

specified and designed for the purposes of dynamic VEs and enables the utilisation of remote business processes in an easy and flexible way. The business processes of each domain are stored into the *Business Process Repository* (BPR). The BPR is a persistent system that stores business process. Additionally, the BPR provides services for the interpretation of processes from XML format into a specialised model that can be easily deployed by the autonomous agents.

In the second phase, the execution and management of business processes in the context of dynamic virtual enterprises is performed. The execution and management of processes is done by a set of autonomous, distributed, inter-domain agents that co-operate among each other to fulfil their mission. The following FIPA compliant agents have been identified for the provision and management of VE processes:

- **Personal User Agent (PUA)** is responsible for managing the requests of the end-users coming from standard web browsers and is located on the VE representative domain. These customer requests can be to start, to resume, to suspend, to terminate a process or get the status of one or more processes. Every request is checked for authorisation and then is forwarded to the Domain Representative agent (DR),
- **Domain Representative (DR)** is responsible for managing the requests of the PUA, if the domain plays the role of the VE representative, and the requests of the remote domains, if the domain plays the role of the VE partner. In both cases, the DR authenticates the requests by conducting the contract repository. If the request is an authorised one and is related to the instantiation of a process, the DR creates a Workflow Provider Agent (WPA) that will serve the request, otherwise the corresponding existing WPA is located and the request is forwarded to him,
- **Workflow Provider Agent (WPA)** is responsible for executing and managing an instance of a process or sub-process. The WPA replies to requests coming from the DR or informs the DR about the status of the process that it executes. Additionally, the WPA co-operates in an autonomous way with other WPAs by exchanging messages specified in the intra or inter-domain ontology during the execution of business processes. Finally, the WPA controls the execution of tasks involved into the business process by invoking, requesting, or informing different Resource Provider Agents (RPA),
- **Resource Provider Agent (RPA)** is responsible for carrying out one specific task of the business process. One task is a simple elementary processing unit that can be included into one or more business processes. An RPA agent always deploys existing resources, business objects, or legacy systems provided by the domain in a distributed and interoperable way.
- **Requestor Negotiation Agent (RNA)** is responsible for managing the partner search, negotiation, and selection process. When a WPA realises that a remote process is required for the continuation of the currently executed business process, it creates automatically a RNA agent. This agent migrates to the virtual marketplace, selects the potential VE candidate partners, based on some constraints, and starts a parallel negotiation process with them. The result of the negotiation is an electronic contract that regulates this agreement.
- **Provider Negotiation Agent (PNA)** represents a VE candidate domain during the negotiation process and is responsible for the automatic negotiations with other RNAs. Additionally, PNAs manage the business process registration to the virtual marketplace and update the contract repository when a negotiation process has been successfully ended, i.e. a contract has been agreed upon.

In addition to the above entities related to business processes, the following internal components have been identified. These are:

- **Inter-domain ontology** is the set of messages exchanged among the autonomous agents located in *different* administrative domains during the *remote* business process execution and management. The specification of the ontology has been done in XML, the format of the messages is based on FIPA ACL-XML, while the protocol used is the standard FIPA request-response protocol,
- **Intra-domain ontology:** is the set of messages exchanged among the autonomous agents located in the *same* administrative domains during the *local* business process execution and management. The specification of the ontology has been done in XML, the format of the messages is based on FIPA ACL-XML, while the protocol used is the standard FIPA request-response protocol,
- **Workflow Engine (WfE)** is the intelligent unit of the WPA agent that controls the status of a running process, evaluates the conditions of the related sub-processes, triggers the creation, suspension, abortion, and termination of the related running agents, and checks whether a process or sub-process has been completed,
- **Offer Repository (OR)** is responsible for the storage of offers and constraints related to the negotiation process. For every local process that has been registered on the marketplaces, an offer is specified into the OR. These offers regulate and drive the negotiation process during the partner selection process,
- **Contract Repository (CR)** is responsible for storing the contracts that have been established between this domain and other remote domains. The contract database is updated automatically when an agreement has been reached after a negotiation process among a RNA and a PNA agent. It is always local to each individual domain.

In the following chapters, analysis and design of the agent-based business process specification, registration, and management system is provided and further details regarding how the agents co-operate to manage business processes in the context of dynamic VE are provided.

4.4.3 Mobile Agent Platform

The emerging research activities related to mobile agent platforms started in the mid nineties, motivated by several advantages promised by this new technology, e.g., asynchronous task execution, reduction of network traffic, robustness, distributed task processing, and flexible on-demand service provision (Chess 98). In course of time, several fundamental requirements and services have been identified due to experiences that have been made during research and development activities (Breugst 98a). These services have to be provided by the mobile agent platform and the requirements that they impose have to be fulfilled by any state of the art mobile agent platform. These services are:

- **Agent Execution Support**: An agent platform must provide the capability to create mobile agents, taking into account agent-specific requirements regarding the runtime environment. Before the creation, the platform has to retrieve the agent's code that may either be delivered with the creation request or downloaded separately from an external, network location code base.

- **Management Support**: It is necessary for agent administrators to be able to monitor and control their agents. The control aspect comprises among others the temporary interruption of an agent's task execution, its premature termination, or the modification of its task. The monitoring of an agent is associated with its localisation in the scope of the whole distributed environment. Regarding an agent system, all hosted agents as well as the occupied system resources have to be monitored and controlled by the system administrator.
- **Mobility Support**: A special mobility support must be provided by the platform, supporting remote execution as well as migration. Note that the mobility aspect cannot be sufficiently handled without regarding the security support mentioned above.
- **Support for Unique Identification**: Mobile agents, as well as, agent systems have to be uniquely identifiable in the scope of the entire agent environment. Thus, special support is required for the generation of unique agent and agent system identifiers.
- **Communication Support**: Agents should be able to communicate with each other, as well as, with platform services. Several mechanisms are possible, such as, messages, method invocation or blackboard mechanisms. Communication through messages may be done point-to-point, by multicasting, or broadcasting. Furthermore, agent communication includes support for semantic analysis. Additionally, standardised agent communication languages and content languages including compatible interpreters should be provided
- **Security Support**: Important aspects are authentication, i.e., the determination of an agent's or system's identity, and access control of resources or services provided by an agent or agent system. To guarantee privacy and integrity, important information such as code and state of a migrating agent should by encrypted before transfer.

Further functional requirements may rise depending on concrete applications. However, these additional features should be separated from the basic, inevitable capabilities mentioned above. These enhanced services should be handled as add-ons that can be "plugged" into a core system in order to individually enhance its functionality. Apart from these functional requirements, various generic demands have to be regarded, such as performance, efficiency, portability, and support for the integration/wrapping of legacy components. One particular legacy technology of pivotal importance is distributed object technology!

The following figure shows the structure of a core agent system comprising several services in order to fulfil the basic functional requirements identified above. Note that some of the services provide remote interfaces in order to be accessible by external actors, such as, other agent systems, agents, or human users.

Thus, in the course of time it become clear that only an integration of agent technology with Distributed Processing Environment (DPE) will provide the prerequisite for the full acceptance of agent technologies. In fact, both, distributed processing environment and agent technologies are complementary. DPE enables the interoperability and reusability of distributed heterogeneous service, or components, i.e., distributed intelligence, whereas Mobile Agent Technology (MAT) allows the dynamic provisioning and extensibility of components, i.e., task delegation and intelligence on demand. Furthermore, intelligent agent technology enables more abstract communication between distributed components via a high level Agent Communication Language (ACL) and thus, more advanced co-operation. As a consequence, an integrated middleware would provide ultimate flexibility for implementing applications in accord to the varying requirements of many different environments.

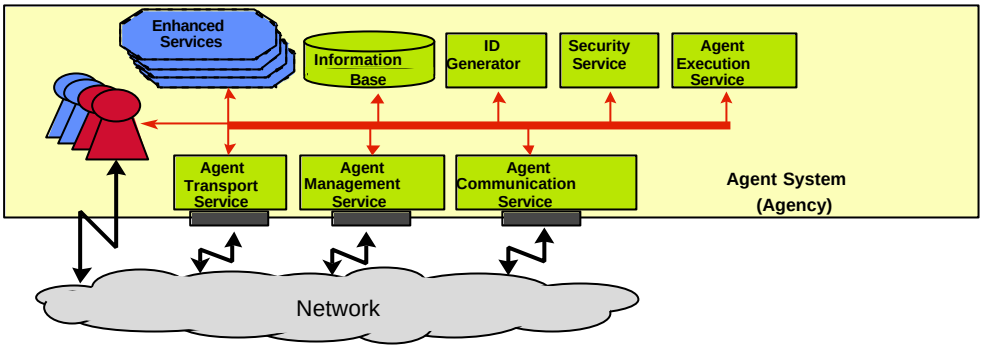

Figure 9: Basic Capabilities of Mobile Agent Platforms

As an example, for some applications it may be desirable to maintain certain service capabilities in a centralised way, i.e., to interact with these remotely, whereas other service capabilities may be realised in a distributed way more effectively. In order to combine these two approaches, a distributed agent environment (DAE) can be established upon a Distributed Processing Environment (DPE), such as CORBA or Java Framework. In this way, a unified environment is built, combining DPE and MAT.

Good examples for existing state of the art mobile agent platforms are Aglets Software Development Kit from IBM (Lange 96), Grasshopper from IKV++ (Breugst 98a), Odyssey from General Magic (USA), Voyager from ObjectSpace (USA), April from Imperial College (UK), D'Agents from Dartmouth College (USA). The first four platforms are Java based while the two following ones non-Java-based.

In the context of this thesis the Grasshopper mobile agent platform with FIPA add on capabilities will be used. The main reason for this decision is actually the fact that Grasshopper is only standard Mobile Agent Platform that supports both FIPA and OMG-MASIF standards. Grasshopper has been developed by GMD FOKUS and IKV++ GmbH and it is a mobile agent development and runtime platform that is built on top of a DPE.

An analytical description of the Grasshopper platform with the FIPA add-on capabilities and details regarding how to implement FIPA compliant agents is provided in the following chapter 5.

4.4.4 Supporting Services

In addition to the underlying Distributed Processing Environment and the Mobile Agent Platform, a set of specialised supporting services will be used. These supporting services are:

- **Extensible Markup Language (XML)** will be used as a content description language for the specification and interpretation of agent messages, as well as, for the specification of business processes. The messages exchanged by agents are described in FIPA-ACL/XML. FIPA ACL is a standard Agent Communication Language based on the speech-act theory. FIPA ACL does not specify the deployment of any specific content description language. The content of the messages and the semantic meaning of requests and responses will be described in XML. Due to the extensible characteristics of XML as a meta-language, business processes will also be described in XML in an flexible and dynamic way,

- **Java Expert System Shell (JESS)** is a Java-based "mini" expert system that allows description of facts and rules related to specific knowledge domains. Jess is a tool for building a type of intelligent software called Expert Systems. Jess uses a special algorithm called Rete to match the rules to the facts. Rete makes Jess much faster than a simple set of cascading if.. then statements in a loop. Jess was originally conceived as a Java clone of CLIPS, but nowadays has many features that differentiate it from its parent. Jess has been developed by Ernest Friedman-Hill at Sandia National Laboratories as part of an internal research project. JESS will be used in the context of this thesis for the specification and evaluation of scheduling conditions of business processes, as well as, for the decision making process during negotiation among agents,
- **OMG Trader** is an OMG standard matchmaking service that enables distributed service providers to register offers in a centralised repository in relation to specific standard service types. Clients, that would like to locate potential providers, conduct the Trader and select a set of potential providers based on some selection constraints. The OMG Trader has been specified and deployed as a mechanism to locate objects in a distributed intra-domain environment. In the context of this thesis, the OMG Trader will be used as a legacy system supporting the basic operations of the agent-based virtual marketplace.

The following figure presents the underlying architecture of the proposed platform. This architecture actually specifies all the services that will be used for the design, development and testing of the agent-based virtual marketplaces and the agent-based business process specification, registration and management systems.

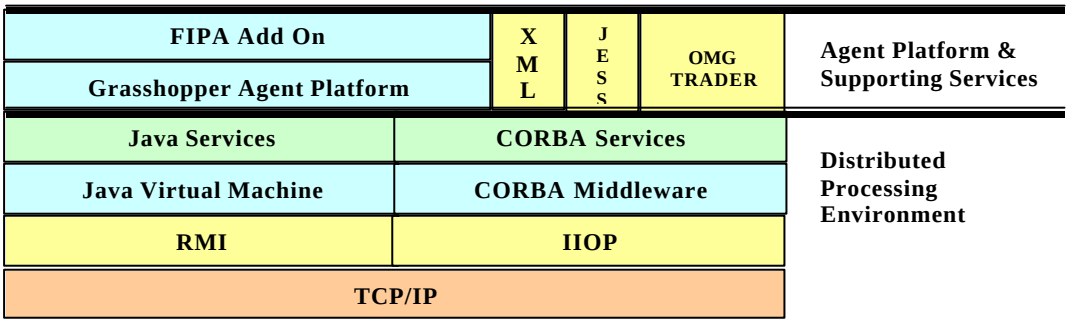

Figure 10: Layered Architecture of Mobile Agent Platform and Supporting Services

4.4.5 Distributed Processing Environment

On the distributed processing environment layer, the basic services offered by the Java Framework (Java) and the CORBA Framework will be used. The main reason for selecting both Java and CORBA frameworks is that they offer the basic means for true interoperable distributed applications. In the following section the rational behind the selection of these frameworks and the key services that will be deployed are explained.

4.4.5.1 The Java Framework

The Java programming language and Framework developed by Sun Microsystems is based on two concepts that appear to make it particularly suitable for the development of multi-agent systems, namely a network-based concept and a platform independent development language.

In traditional programming languages, a compiler or a runtime interpreter is used to convert the program source code into system-specific binary code. Java adopts another approach. The Java compiler does not directly translate the Java source code into binary code but into a so-called Java byte-code. This byte-code is platform-independent and can be executed without modification on all platforms that support Java. A Java interpreter developed for a particular platform is used to execute the byte-code on the target platform. The virtual machine is added to the existing operating system of the target computer and provides a simulated, consistent, runtime environment. Irrespective of the actual system platform, for example, UNIX, Windows or Mac-OS, the Java Virtual Machine always provides a Java program with a standardised runtime environment.

The byte-code can be either executed locally, as Java applications, or transferred over the network to a remote computer where it is executed as a so-called Java applet. In the Web, for example, the Java applet is embedded in the HTML page, transferred together with the HTML page, and executed on the target computer in a browser. The browser must provide the required Java runtime environment, i.e., a Java Virtual Machine and a Java interpreter. In a rather similar way, a java object can be sent to a remote physical address and continue execution. This leads directly to the concept of mobile agents. However, the existing Java Framework does not provide the basic services for mobility of objects. For that reason dedicated mobile agent platforms have been built upon the Java Framework that try to solve this problem and provide specific services for the migration of services.

The network-based concept of Java and, in particular, the principle of remote objects, place demands on the underlying security model that far exceed those provided by conventional programming languages. If a user loads an existing object from the network and executes it on his computer, he permits an object that he does not know to execute on his local system. Although the task of the applet may provide the user with a general idea of the concrete actions that the object performs, he can never be sure whether the applet behaves as expected or whether under certain circumstances it performs some unwanted actions.

As seen from the security model, the execution of an object can be considered from two directions. In one case, an object could have full access to the execution system. This corresponds to the traditional model in which an operating system permits the executing software programs to have access to all important system functions. This concept requires the user to have complete trust in the executing program and the operating system to perform preliminary checks on the program. Both aspects are difficult to realise for Java applets and in general objects. The actions of an object can be restricted to a specified space within the system. Java refers to this as the object's "sandbox". Although an object can perform any actions it wishes within its sandbox, it has no access capabilities to resources that lie outside the sandbox. The Java Virtual Machine provides the sandbox in practice. The security concept of Java[9]

[9] The interested reader can read more bout the security model of Java on the http://java.sun.com/

consists of various components, some of which are positioned in the Java language itself whereas other parts are in the object executing application.

Object serialisation extends the central input/output classes of Java and permits sending a Java object over a network connection. This extension plays an important role in the implementation of mobile agents under Java framework. The functionality of the object serialisation is concerned with the packaging of all the classes that belong to the object, including the classes that the object can access, into a serial data stream, the transfer of this data stream over the network, and the reconstitution of the original object at the target computer. Thus, object serialisation supports the creation of persistent objects. An object can be interrupted at any point in its execution, packaged and transferred, reconstituted at the target computer and continued at the original program location. This is also a fundamental service that is used by the mobile agent platforms.

In the context of this thesis, the Java programming language and therefore, the Java Virtual Machine will be used for developing mobile intelligent agents based on a mobile agent platform. The underlying services of Java framework, like security service, Remote Method Invocation service, serialisation service, and basic core services, like hashtables and vectors, will be deployed as basic middleware services for the required functionality within the agents.

4.4.5.2 The CORBA Framework

CORBA is a standard, defined by the Object Management Group (OMG 98), for implementing distributed applications. Currently, CORBA is integrated and supplied with all kind of tools and systems like databases, web browsers, program development environments standard object oriented analysis and design methods, e.g., UML, and support plenty of operating systems.

Basically, CORBA provides two basic benefits. First, the capability of an object to request an operation provided by a distributed object and to receive the results. Second, a set of standard distributed services, which can be accessed in the same way as other distributed objects. What makes CORBA an advantage is its support for interoperability. This means that CORBA objects can be implemented in many different programming languages and run in different environments with varying transport protocols, operating systems and hardware. This interoperability is real, and supported by almost all the significant technology providers in the IT domain.

Apart from the ORB, there is a set of standard middleware services and a component model. This means that certain standard compliant services already implemented by different vendors can be easily deployed and integrated into a distributed CORBA environment.

OMG defines a set of system-level services called *CORBAservices*. These services are of general use, such as Naming service, Trader service, Life-cycle service, Event service, Security service, etc. They are assumed to be widely available and affect the architecture of the application. *CORBAfacilities* are general-purpose services useful in many application domains, but with no relevant impact in the architecture. An example is the Printing Spooling facility. Those facilities specific to market sectors, such as financial services, manufacturing, or telecommunications are defined as *CORBAdomains*

In the context of this thesis, the CORBA framework will be used for integrating and deploying distributed business objects that can be directly used by an autonomous agent in the context of a business process. Additionally, the CORBA framework will be used for wrapping existing

legacy systems and offer their services in a distributed multi-agent environment, like the integration of OMG Trader into the virtual marketplaces.

4.5 Summary

This chapter presents a layered architecture of the system under analysis and identifies the key business domains, actors, activities and agent required. Initially, a generic description of the analysis method is presented. The analysis and specification approach used is the Unified Modelling Language (UML) an OMG standard. In the sequel, three layers are identified, namely the agent-based Virtual Marketplace and Business Process Specification, Registration and Management layer, the Mobile Agent Platform and Supporting Services layer and the Distributed Processing Environment (DPE) layer. For the first layer, which consists the core contribution of this work, a set of autonomous and intelligent agents are identified and roughly presented. These agents will be further analysed and specified in subsequent chapters.

Chapter 5: Mobile Agent Platform

5.1 Introduction

Grasshopper is a CORBA-based MASIF conformant mobile agent platform which has been enhanced recently with a FIPA add on in order to give the application developer total flexibility. This evolution of the platform is witnessing the fact that the traditional separation of mobile agents and intelligent agents is fading away as the corresponding standards bodies, i.e., OMG-Agent SIG and FIPA are aiming to develop compatible standards. Thus, Grasshopper enables its users to develop a broad range of agents, ranging from small simple mobile agents roaming the network nodes, up to static multi agent systems talking via an Agent Communication Language (ACL) for distributed problem solving.

Today Grasshopper is the agent platform of choice in multiple international research projects within the European CLIMATE *(Cluster for Intelligent Mobile Agents for Telecommunication Environments)* initiative (Climate 99). A common aspect of most of these projects is to explore the usage of agent-based middleware in particular application domains, such as service control in fixed and mobile networks, telecommunications management, electronic commerce, multimedia applications, etc.

The emerging research activities related to mobile agent platforms started in the mid nineties, motivated by several advantages promised by this new technology, e.g. asynchronous task execution, reduction of network traffic, robustness, distributed task processing, and flexible on-demand service provision. Many research labs and manufacturers were involved in the development of various platforms, built on top of different operating systems, based on different programming languages and technologies. However, within the last few years, common trends can be noticed: Interpreter-based programming languages, particularly Java, are forming the basis for most of today's agent platforms. Additionally several approaches are associated to the integration of mobile agents and RPC-based middleware like CORBA.

5.1.1 Distributed Agent Environment

In principle, Grasshopper realizes a *Distributed Agent Environment* (DAE). The DAE is composed of regions, places, agencies and different types of agents. Figure 2 depicts an abstract view of these entities.

Two types of agents are distinguished in Grasshopper: mobile agents and stationary agents. The actual runtime environment for both mobile and stationary agents is an *agency*: on each host at least one agency has to run to support the execution of agents. A Grasshopper agency consists of two parts: the core agency and one or more places. Core Agencies represent the minimal functionality required by an agency in order to support the execution of agents.

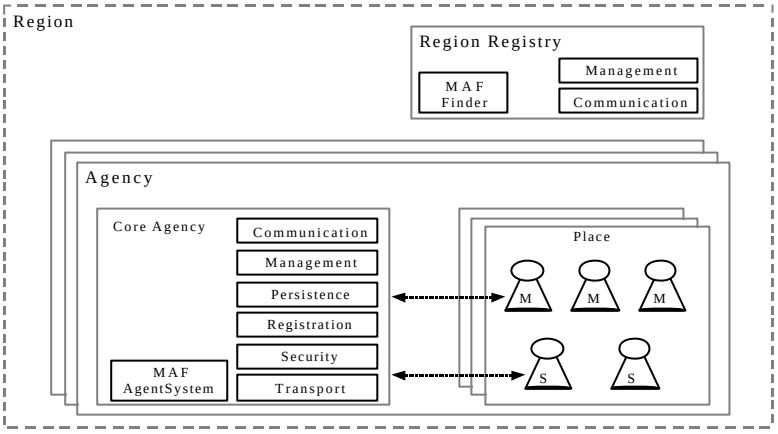

Figure 11: Grasshopper Distributed Agent Environment

The following services are provided by a Grasshopper core agency:

- **Communication Service:** This service is responsible for all remote interactions that take place between the distributed components of Grasshopper, such as location-transparent inter-agent communication, agent transport, and the localization of agents by means of the region registry. All interactions can be performed via CORBA-IIOP, Java-RMI, or plain socket connections. Optionally, RMI and plain socket connections can be protected by means of the Secure Socket Layer (SSL) which is the de-facto standard Internet security protocol. The communication service supports synchronous and asynchronous communication, multicast communication, as well as dynamic method invocation. As an alternative to the communication service, Grasshopper can use its OMG MASIF-compliant CORBA interfaces for remote interactions. For this purpose, each agency provides the interface *MAFAgentSystem*, and the region registries provide the interface *MAFFinder*.
- **Registration Service:** Each agency must be able to know about all agents and places currently hosted, on the one hand for external management purposes and on the other hand in order to deliver information about registered entities to hosted agents. Furthermore, the registration service of each agency is connected to the region registry which maintains information of agents, agencies and places in the scope of a whole region.
- **Management Service:** The management services allow the monitoring and control of agents and places of an agency by users. It is possible, among others, to create, remove, suspend and resume agents, services, and places, in order to get information about specific

agents and services, to list all agents residing in a specific place, and to list all places of an agency.

- **Security Service:** Grasshopper supports two security mechanisms: external and internal security.
  - **External security** protects remote interactions between the distributed Grasshopper components, i.e. between agencies and region registries. For this purpose, X.509 certificates and the SSL are used. By using SSL, confidentiality, data integrity, and mutual authentication of both communication partners can be achieved.
  - **Internal security** protects agency resources from unauthorised access by agents. Besides, it is used to protect agents from each other. This is achieved by authenticating and authorising the user on whose behalf an agent is executed. Due to the authentication/authorisation results, access control policies are activated.
- **Persistence Service:** The Grasshopper persistence service enables the storage of agents and places on a persistent medium. This way, it is possible to recover agents or places when needed, e.g. when an agency is restarted after a system crash.

A place provides a logical grouping of functionality inside of an agency. The region concept facilitates the management of the distributed components, i.e. agencies, places, and agents, in the Grasshopper environment. Agencies, as well as, their places can be associated with a specific region by registering them within the accompanying region registry. All agents, which are currently hosted by those agencies will also be automatically registered by the region registry. If an agent moves to another location, the corresponding registry information is automatically updated.

5.1.2 Communication Concepts

The communication facilities of Grasshopper are realised by the *Communication Service (CS)* which is an essential part of each core agency. This communication service allows location-transparent interactions between agents, agencies, and non-agent-based entities.[10]

Remote interactions are generally achieved by means of a specific protocol. The communication service supports communication via the IIOP, Java's RMI, and plain socket connections. To achieve a secure communication, RMI and the plain socket connection can optionally be protected with the SSL.

- **CORBA IIOP:** The CORBA 2.0-compliant Internet Inter-ORB Protocol can be used in all environments that support CORBA, independent of a vendor-specific ORB implementation. It uses the standard-compliant mechanism to connect to an object using a CORBA Naming Service.

[10] As an alternative to the communication service, Grasshopper can use its OMG MASIF-compliant CORBA interfaces for remote interactions. For this purpose, each agency provides the interface *MAFAgentSystem*, and the region registries provide the interface *MAFFinder*. Those interfaces are defined in the MASIF standard. Note that the following sections only describe the Grasshopper communication service. Detailed information about MASIF can be found in the standard itself.

- **MAF IIOP:** This protocol is a specialisation of CORBA IIOP developed for agent system interaction. It is introduced in the MASIF standard and provides the connectivity between agent systems of different vendors. Thus, MASIF IIOP does not use the Grasshopper communication service and connects directly to the MASIF interface of the peer agency.
- **RMI:** Java Remote Method Invocation enables Java objects to invoke methods of other Java objects running on another Virtual Machine (VM). Since this protocol is included in every JDK1.1-compliant VM, all Grasshopper agencies support this protocol by default without any further installation or configuration effort.
- **Plain Sockets:** The fastest way of remote interactions is the communication via plain sockets to a specific port of the target host. This technique is robust and avoids the overhead of a distributed object model. Plain socket communication is possible in each Internet-enabled environment and it is the default protocol used by Grasshopper agencies.
- **Plain Sockets with SSL:** Using this protocol, plain socket connections are protected by SSL. The preconditions for usage are the same as those mentioned for RMI/SSL.

Inside of a region, Grasshopper is able to determine dynamically the protocols supported by a desired communication peer and to select the most suitable protocol for the remote interactions. Since the supported communication protocols are realized via a plug-in interface, developers can easily integrate new communication protocols by writing their own protocol plug-ins. In this way Grasshopper is open for future requirements that may come up in the changing communication world.

The communication service is used internally by the Grasshopper system for the agent transport, for locating entities within the DAE, etc. Agents can use the communication service to invoke methods on other agents. Since an agent does not have to care about the location of a desired communication peer, the communication is totally location-transparent. Within the agent code, there is no difference between remote method invocations and local method invocations.

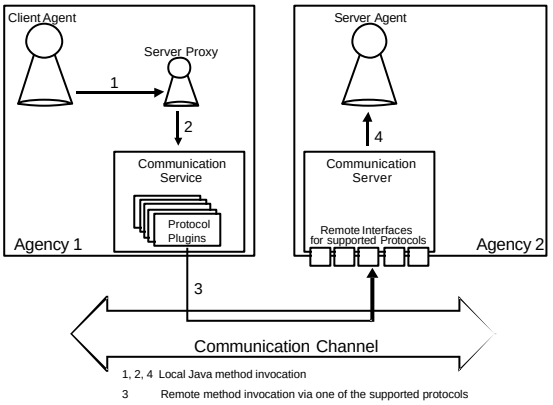

Figure 12: Location Transparent Communication

This is achieved by means of so-called *proxy objects* that are directly accessed by a client. The proxy object forwards the call via the ORB to the remote target object. In this way, these proxy objects are equivalent to the client stubs used by CORBA implementations.

Inter-agent communication within Grasshopper may be performed in several modes. Grasshopper supports the following communication modes:

- **Synchronous Communication:** Usually, when a client invokes a method on a server, the server executes the called method and returns the result to the client, which then continues its work. This style is called *synchronous* because the client is blocked until the result of the method is sent back.
- **Asynchronous Communication:** When using asynchronous communication, the client does not have to wait for the server executing the method. Instead the client continues performing its own task. There are several possibilities for the client to get the result of the invoked method: It can periodically ask the server whether the method execution has been finished, wait for the result whenever it is required, or subscribe to be notified when the result is available.
- **Dynamic Communication:** This mechanism is useful if the client does not have access to a server proxy. The client is able to construct a message at runtime by specifying the signature of the server method that shall be invoked. Dynamic messaging can be used both synchronously and asynchronously.
- **Multicast Communication:** Multicast communication enables clients to use parallelism when interacting with server objects. By using multicast communication, a client is able to invoke the same method on several servers in parallel.

5.1.3 Security Concepts

As mentioned, the Grasshopper security service supports two different kinds of security mechanisms: external and internal security.

External security protects remote interactions between the distributed Grasshopper components. The external security mechanisms are based on the use of X.509 certificates and the SSL. SSL is an industry-standard protocol that makes substantial use of both symmetric and asymmetric cryptography.

- **Confidentiality:** The whole communication between clients and servers is handled via secure sockets, encrypted with a symmetric keys and an encryption algorithm negotiated in a handshake prior to the actual SSL session. Although the IP packets can still be intercepted, encryption renders them useless for eavesdroppers. Currently, Grasshopper uses RC4 with 128 bit keys for encryption and RSA with 1024 bit keys for session key exchange,
- **Integrity:** Message Authentication Codes (MACs) can prove that a message was not modified during transportation. These MACs are calculated for each SSL packet using hash functions. Grasshopper uses MD5 in conjunction with shared secrets to generate these MACs,
- **Authentication:** The purpose of authentication is that both communication parties convince each other of their identity. During the SSL handshake, client and server exchange personal data and their public keys packaged together in the form of X.509 certificates. The authentication process requires both parties to digitally sign protocol data with their private keys. The certificate itself does not authenticate, but the combination of certificate and

correct private key does. Currently, Grasshopper uses RSA with 1024 bit keys for authentication,

Internal security protects resources of an agency from unauthorized access by agents. Furthermore, it is useful to protect agents from each other.

Regarding access control, Grasshopper is strongly oriented towards the security mechanisms of JDK 1.2. It makes use of an identity-based and group-based access control policy, which is initialized at start-up. In Grasshopper, an access control policy is an access control list comprising several entries, one for each subject treated in this policy, where a subject can be a single identity or a group consisting of 1..n members. A set of permissions is associated with each subject, granting access to all important parts of the Grasshopper agency. Each permission consists of a type, a target and optionally one or more actions.

When an agent tries to make a system access an access controller is consulted to make the access decision. In fact, each time a system access happens the access controller is invoked, but it is capable of distinguishing whether the access was made by an agent or by trusted system code, e.g. the Grasshopper core. If the access came from an agent, the access controller extracts the agent's owner from the agent itself. With this information, it contacts the Policy object, a runtime representation of the Grasshopper access control policy, to extract the set of permissions valid for this subject. If the subject is a member in one or more groups, the group permissions are added to the individual permissions. It is then checked if the permission to perform the access is contained in the set of permissions granted to the subject. If not, an access control exception is thrown.

Persistence is an important topic in the field of distributed applications. Objects are sent from one computer to another and often have an extended life span. That is especially valid for mobile agents. The following undesirable scenarios have to be taken into account:

- An agent moves from one agency to another. The transmission fails for some reason so that the agent never arrives at its destination.
- An agent is residing within an agency whose host computer crashes or shuts down unexpectedly (e.g. due to a power failure).
- There are many agents residing within an agency, with most of them waiting for external events without performing any task, thus just wasting system resources. Therefore the host computer could run out of resources (especially memory) if more agents want to migrate into that agency.

While the first scenario can be avoided by buffering the agent until its arrival has been confirmed, the remaining two need other approaches. A copy of the agent object has to be maintained on a persistent medium, e.g. a hard disk. If the agency system crashes, persistent agents can be reloaded from this medium after the agency has been restarted. Besides, idle agents, i.e. agents just waiting for an event without executing any task, do not need to remain instantiated, but could be stored permanently and then removed from the agency's memory in order to save resources. If a request for such a flushed agent arrives, it can be re-instantiated in order to handle the request.

Grasshopper provides mechanisms to handle all the topics mentioned above if the persistence service is enabled.

5.1.4 Agent Development

The functionality of Grasshopper is provided on the one hand by the platform itself, i.e. by *core agencies* and *region registries*, and on the other hand by *agents* that are running within the agencies, in this way enhancing the platform's functionality. The following possibilities regarding the access to the Grasshopper functionality must be distinguished:

- Agents can access the functionality of the *local agency*, i.e. the agency in which they are currently running, by invoking the methods of their super classes `Service`, `StationaryAgent`, and `MobileAgent`, respectively. These super classes are provided by the platform in order to build the bridge between individual agents and agencies, and each agent has to be derived from one of the classes `StationaryAgent` or `MobileAgent`.
- Agents as well as other DAE or non-DAE components, such as user applications, are able to access the functionality of *remote agencies* and *region registries*. For this purpose, each agency and region registry offers an external interface which can be accessed via the Grasshopper communication service.
- Agencies and region registries may optionally be accessed by means of the MASIF-compliant interfaces `MAFAgentSystem` and `MAFFinder`.

In the context of Grasshopper, each agent is regarded as a *service*, i.e. as a software component that offers functionality to other entities within the DAE. Each agent/service can be subdivided into a common and an individual part. The common (or core) part is represented by classes that are part of the Grasshopper platform, namely the classes `Service`. `MobileAgent`, and `StationaryAgent`, whereas the individual part has to be implemented by the agent programmer.

A Grasshopper agent consists of one or more Java classes. One of these classes builds the actual core of the agent and is referred to as *agent class*. Among others, this class has to implement the method `live` which specifies the actual task of the agent. The agent class must be derived, either from the class `StationaryAgent`, or from the class `MobileAgent`, which in turn inherits from the common super class `Service`. The methods of these classes represent the essential interfaces between agents and their environment. The following two ways of method usage have to be distinguished:

- One part of the super class methods of an agent enable the access to the local core agency. For example, an agent may invoke the method `listMobileAgents()`, which it inherits from its super class `Service`, in order to retrieve a list of all other agents that are currently residing in the same agency.
- The remaining super class methods are defined to access individual agents. These methods are usually invoked by *other* agents or agencies via the *communication service* of Grasshopper. For instance, any agent may call the method `getState()` of *another* agent in order to retrieve information about the other agent's actual state. Note that this way of access is not performed directly on an agent instance, but instead on an agent's *proxy* object.

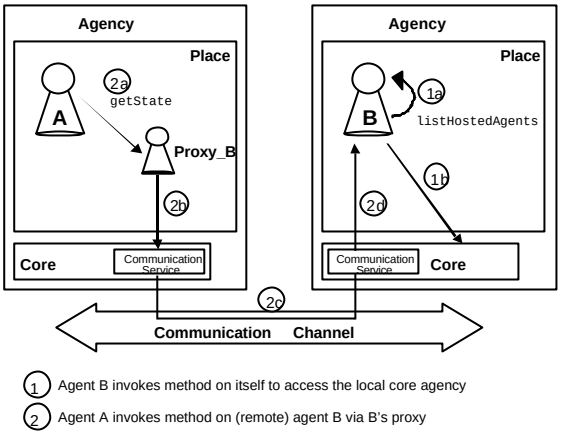

Figure 13: Access of an Agent's Methods

In order to contact a remote agency, the client (e.g. an agent, agency, or user application) must have access to an agency proxy object. The remotely accessible functionality of each Grasshopper agency can be separated into the following parts:

- **Registration functionality** offers detailed information about all places as well as agents/services that are currently hosted by a remote agency.
- **Service control functionality** enables the remote control of places and agents/services within an agency, such as agent creation, suspension, resumption, transport, and termination.
- **Persistence functionality** supports the persistent storage of agents/services and places within a remote agency.
- **Listener functionality** enables the registration and de-registration of `AgentSystemListeners` for remote agencies.

The RegionRegistration Interface

In order to contact a remote region registry, the client must have access to a registry proxy. The functionality of a Grasshopper region registry comprises the registration and de-registration of agents/services, places, and agencies. Besides, lookup methods enable the retrieval of specific information about the registered components.

The RegionRegistryListener Interface

This interface can be used to monitor the events occurring in the region registry, and to present them to a user. The listener is notified about any changes associated with the registration/de-registration of agencies, agents/services and places, and it retrieves any output from the registry. Each listener is identified by means of a unique identifier. Listener implementations may be realized by platform users, e.g. in order to create individual graphical user interfaces.

The following figure gives an overview of the remotely accessible interfaces described above.

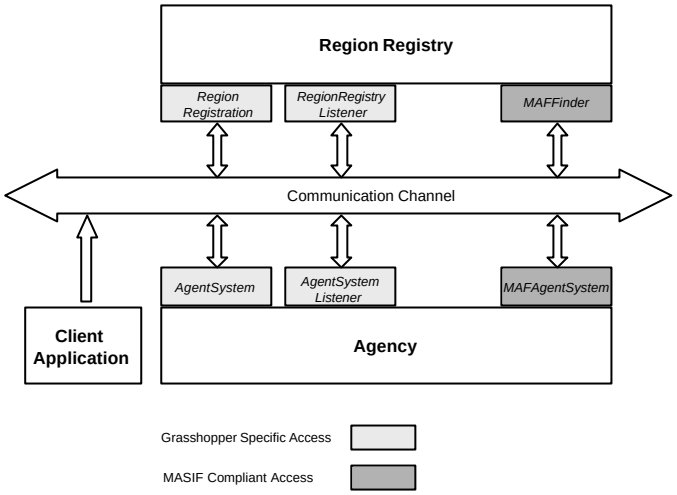

Figure 14: Remotely Accessible Grasshopper Interfaces

In the following section, the key concepts and functionality of the FIPA standards and platform are described. Emphasis is placed on how the FIPA platform is integrated into the Grasshopper agent platform and how standard FIPA agents can be developed.

5.2 Foundation for Intelligent Physical Agents

The Foundation for Intelligent Physical Agents (FIPA) is a non-profit association registered in Geneva, Switzerland. FIPA's purpose is to promote agent technology through the development of specifications that maximise interoperability across agent-based applications, services and equipment. FIPA specifies the interfaces of the different components in the environment with which an agent can interact, i.e., humans, other agents, non-agent software and the physical world. The main emphasis in FIPA is on standardising agent communication, which is a dedicated Agent Communication Language (ACL) is used for all communication between FIPA Agents.

FIPA specifications are developed in a yearly manner. In October 1997, FIPA released its first set of specifications, called FIPA `97, Version 1.0. This set of specifications comprises seven parts. The three main normative specifications (parts 1-3) are focusing on agent management define an agent communication language, and deal with agent/software interaction. These specifications have been derived from examining requirements on agent technology posed by specific industrial applications chosen by FIPA.

In 1998, FIPA started work on an enhanced set of specifications, called FIPA`98. In addition to modifications and extensions of the normative FIPA'97 specifications, FIPA'98 also contains six new parts. These are: Human Agent Interaction, Product Design and Manufacturing Agents, Agent Security Management, Agent Management Support for Mobility, Ontology Service, and FIPA´97 Developers Guide.

Looking for the definition of a FIPA agent platform, one has to look at the normative Agent Management System specification (FIPA 98a). Agent management provides the normative framework within which FIPA Agents exist and operate. It establishes the logical reference model for the creation, registration, location, communication, migration and retirement of agents

and thus is very much related to capabilities of a FIPA agent platform. The following figure is a graphical representation of the agent management reference model.

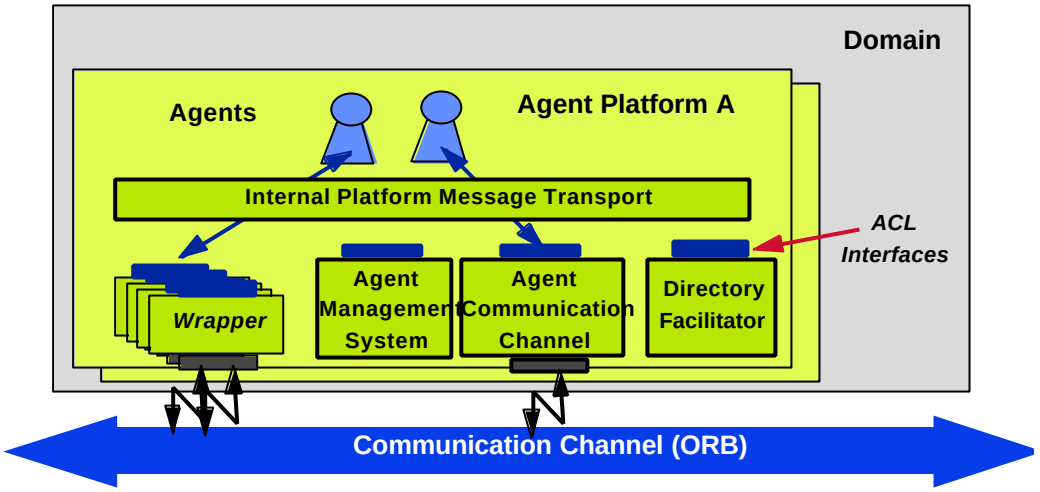

Figure 15: FIPA Agent Management Reference Model

FIPA proposes the concept of an *Agent Platform (AP)* offering three basic services. These services are namely the *Agent Management System (AMS)*, the *Directory Facilitator (DF)* and the *Agent Communication Channel (ACC)*. Agents are considered residing on a home agent platform (HAP) if they are registered on the home agent platforms' AMS. Agents may offer their services to other agents and make their services searchable in a yellow pages manner by the Directory Facilitator if they register on the Directory Facilitator. Registration on a Directory Facilitator is optionally while registering on the AMS is mandatory on an agent platform. Finally, the Agent Communication Channel is enabling agent communication between agents on a platform and between platforms by offering a message forwarding service called *forward*. Reachability between platforms is gained by making the forward service available by the OMG-IIOP.

In summary it can be recognised, that a FIPA agent platform provides the physical infrastructure in which intelligent agents can be deployed. An agent must be registered on an agent platform in order to interact with other agents on that agent platform or other agent platforms. In fact the concept of agent platform can be regarded as a refinement of the facilitator concept in the traditional intelligent agent frameworks.

5.2.1 The Agent Communication Language

FIPA provides the agent designer with speech-act-based performatives (speech act category with well-known semantics) and a standard syntax for messages. These messages are based on the speech act theory. The theory is a result of the linguistic analysis of the human communication and is based on the work of (Searle 69). The key maxim of the speech act theory is that producing language is an action. The action is performed by a speaker who intends to change the mental state of the listener. FIPA provides the agent designer with speech-act-based performatives and a standard syntax for messages.

The main structural elements of an ACL message are depicted in the following figure:

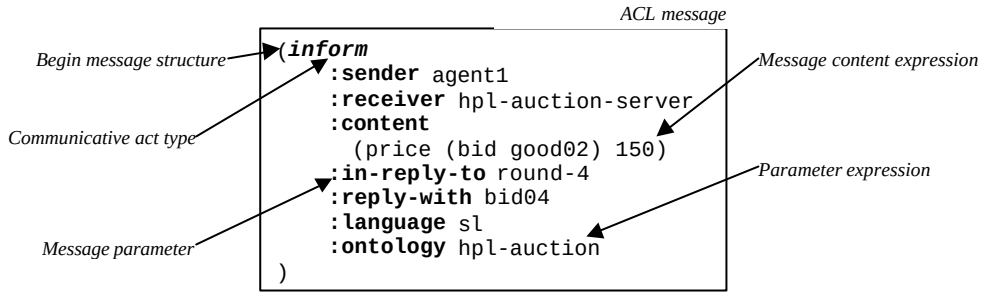

Figure 16: Structure of an ACL Message

The semantic meaning of the individual message parameters of the message is the following: (FIPA 97b):

- Sender: Denotes the identity of the sender of the message, i.e. the name of the agent of the communicative act,
- Receiver: Denotes the identity of the intended recipient of the message that might be single agent name, or a tuple of agent names. This corresponds to the action of multicasting the message. Pragmatically, the semantics of this multicast is that the message is sent to each agent named in the tuple, and that the sender intends each of them to be recipient of the CA encoded in the message,
- CONTENT: Denotes the content of the message; equivalently denotes the object of the action,
- REPLY-WITH: Introduces an expression which will be used by the agent responding to this message to identify the original message. Can be used to follow a conversation thread in a situation where multiple dialogues occur simultaneously,
- IN-REPLY-TO: Denotes an expression that references an earlier action to which this message is a reply,
- ENVELOPE: Denotes an expression that provides useful information about the message as seen by the message transport service. The content of this parameter is not defined in the specification, but may include time sent, time received, route, etc.,
- LANGUAGE: Denotes the encoding scheme of the content of the action,
- ONTOLOGY: Denotes the ontology which is used to give a meaning to the symbols in the content expression,
- REPLY-BY: Denotes a time and/or date expression which indicates a guideline on the latest time by which the sending agent would like a reply,
- PROTOCOL: Introduces an identifier, which denotes the protocol which the sending agent is employing. The protocol serves to give additional context for the interpretation of the message.
- CONVERSATION-ID: Introduces an expression, which is used to identify an ongoing sequence of communicative acts, which together form a conversation. A conversation may be used by an agent to manage its communication strategies and activities.

The following table categorizes the different key communicative acts:

Communicative act	Information passing	Requesting information	Negotiation	Action performing	Error handling
Accept-proposal			✓		
Agree				✓	
Cancel				✓	
Cfp			✓		
Failure					✓
Inform	✓				
Not-understood					✓
Propose			✓		
query-if		✓			
Refuse				✓	
reject-proposal			✓		
Request				✓	

Table 3: Categories of Communicative Acts [FIPA97P2]

The following example should give a motivation for using the FIPA-ACL. Suppose an agent, named 'Peter', requests the delivery of a parcel to a certain location ('home'), then the agent could send the following message:

```
(request
    :sender peter
    :receiver parcel-service
    :content (action parcel-service
                    (deliver parcel42 (location home)))
    :protocol fipa-request
    :reply-with order1
)
```

The receiving parcel-service agent working on behalf of the parcel-service provider can confirm the request. The parcel-service agent may have problems to understand the location 'home'. In this case, the agent can send a request message back to agent 'Peter' asking to determine the location 'home'. In the other case the parcel-service agent knows the home location of agent 'Peter', because this information is already stored in its database. Then, the parcel-service agent would send the following agree message:

```
(agree
    :sender j
    :receiver i
    :content ((deliver j parcel42 (location home))
    :in-reply-to order1
    :protocol fipa-request
)
```

The Grasshopper FIPA Add-On implementation supports the new standard language for the World Wide Web, i.e., the eXtensible Markup Language (XML) as well the FIPA ACL as of

FIPA 97 Specification (part 2). An appropriate parser for FIPA-ACL has been implemented and is provided to the FIPA agents, i.e., the FIPA platform components ACC, DF, AMS as well as to the applications. With XML, appropriate ontologies can be specified for different application domains. The communication between the FIPA agents takes place with the default communication language, either FIPA-ACL or ACL/XML. The real message content e.g., message payload can be encoded in FIPA SL1 or also in XML without any extra user intervention. To support any other proprietary content language, it is left to the users, to develop their specific own parser implementation.

5.2.2 Content of ACL Messages

According to FIPA the content of an ACL message can be encoded in any content language. A content language must be able to express propositions, objects, and actions. No other properties are required, though any given content language may be much more expressive than this. More specifically, the content of a message must express the data type of the action: propositions for inform, actions for request, etc.

In this context, a proposition can be a sentence, e.g. in predicate-logic, which can be true or false. An object represents an abstract or a concrete entity, which does not necessary appear in an object-orientated languages. An action can be regarded as activity, which can be carried out by an agent. Possible candidate content languages are:

- Knowledge Interchange Format (KIF). KIF is a prefix version of first order predicate calculus.
- Semantic Language (SL),
- Prolog,
- eXtended Markup Language (XML).

XML is regarded as the next generation of HTML, which, different from HTML, allows the definition of new tags in the documents. By deploying XML as agent communication content language is important because the presentation of agent communication messages in Web browsers and the integration with existing Web-based applications can be easily performed.

There are two possible ways for encoding the ACL messages within XML documents:

- Encode only the ACL message content as XML documents and keep the ACL container format specified in the current FIPA specifications, or
- Encode the whole ACL message both, the message layer and the content layer in a XML document.

The first approach conforms to the current FIPA specifications, which allow the deployment of arbitrary content language within the standardised ACL wrappers. The disadvantage is the complexity of the implementation of the agent platform and the applications. Typically, in this case two parsers are needed, one for parsing the ACL message and one for parsing the XML content. The second approach does not comply with the current FIPA specification and thus, will not be used in the context of this thesis.

Furthermore the knowledge and semantic meaning of messages exchanged between agents should be described in an ontology using also a content language. An ontology describes the

meaning of symbols and expressions in a domain. This description is expressed in the appropriate content language of an agent. The ontology assigns a constant symbol in an agent message a well-understood meaning. This ensures that agents with the same ontology have the same semantic understanding of a message. In most cases, the agent programmers specify the ontology. The applications developer should agree on a specific content language, e.g. XML, and specify the common ontology that will be used. Based on these observations, the content of the ACL messages will be fully encoded in XML and while the specification of the ontology for these messages will be done in XML-DTDs.

5.2.3 FIPA Protocols

Ongoing conversations between agents often fall into typical patterns. In such cases, certain message sequences are expected, and, at any point in the conversation, other messages are expected to follow. These typical patterns of message exchange are called protocols. A designer of an agent system has the choice to make the agents sufficiently aware of the meanings of the messages, goals, beliefs and other mental attitudes the agent possesses. This, however, places a heavy burden of capability and complexity on the agent implementation, though it is not an uncommon choice in the agent community at large. An alternative, and very pragmatic, view is to pre-specify the protocols, so that a simpler agent implementation can nevertheless engage in meaningful conversation with other agents, simply by carefully following the known protocol. FIPA specifies a number of protocols, in order to facilitate the effective inter-operation of simple and complex agents.

The FIPA-request-inform protocol simply allows one agent to request from another agent to perform some action. The receiving agent should first perform the action and then reply with an inform message or explicitly specify that it can not perform it. Based on this protocol, the requestor agent sends a request ACL message and the receiver of the message can reply with an inform ACL message stating that the request has been performed.

In the FIPA-query-response protocol, the receiving agent is requested to perform some kind of query action. Requesting by an agent to query results in the generation of an inform message with the results of the query. There are two query-acts: query-if and query-ref. Both acts may be used to initiate this protocol. If the protocol is initiated by a query-if act, the responder will plan to return the answer to the query with a normal inform act. If the request has been initiated by query-ref, it will instead be an inform-ref.

The FIPA contract-Net Protocol is a version of the widely used *Contract Net Protocol*, originally developed by Smith and Davis (Smith & Davis 80). FIPA-Contract-Net is a minor modification of the original contract net protocol in that it adds rejection and confirmation communicative acts. In the contract net protocol, one agent takes the role of a manager. The manager wishes to have some task performed by one or more other agents and further wishes to optimise a function that characterises the task. This characteristic is commonly expressed as the price, in some domain specific way, but could also be the fastest time to completion, fair distribution of tasks, etc. The manager solicits proposals from other agents by issuing a *call for proposals*, which specify the task and any conditions the manager is placing upon the execution of the task. Agents receiving the call for proposals are viewed as potential contractors and are able to generate proposals to perform the task as propose acts. The contractor's proposal includes the preconditions that the contractor is setting out for the task, which may be the price, time, completion time, etc. Alternatively, the contractor may refuse to propose. Once the manager

receives back replies from all of the contractors, it evaluates the proposals and makes its choice of which agents will perform the task. One, several, or no agents may be chosen. The agents of the selected proposal(s) will be sent an acceptance message while the others will receive a notice of rejection. The proposals are assumed to be binding on the contractor, so that once the manager accepts the proposal the contractor acquires a commitment to perform the task. Once the contractor has completed the task, it sends a completion message to the manager. In the case that a contractor fails to reply with either a propose or a refuse, the manager may potentially be left waiting indefinitely. To guard against this, the *cfp* message includes a deadline by which replies should be received by the manager. Proposals received after the deadline are automatically rejected with the given reason that the proposal was late.

Other standardised FIPA protocols are: Iterated-Contract-Net, Request-when Protocol, Auction-English Protocol, Auction-Dutch Protocol.

In the context of this thesis the FIPA request-response, FIPA query-response, and FIPA Contract-Net protocols have been developed and used. The particular usage of these protocols in the context of this thesis is further explained in the subsequent chapters.

5.2.4 Agent Management System Agent

The Agent Management System is the core of any agent platform. It is responsible for registering agents on their HAP. An agent is residing on an agent platform which is then its home agent platform, if and only if it is registered with the agent platform`s AMS. Registering on an AMS is done by calling the AMS' *message* method with a request to register encoded in an ACL string conforming to the FIPA ACL definition. Alternatively registration can be done by calling the request method of the AMS with a data structure equivalent to an ACL message containing such data as sender, receiver and content of the request, the content in this case containing the agent description data. Other functionality offered by the AMS is deregistering of agents, modification of the agent description and modifying the agents life cycle state.

The AMS is responsible for managing the operation of an AP. These responsibilities include creation of agents, deletion of agents, deciding whether an agent can dynamically register with the AP and overseeing the migration of agents to and from the AP. Since different APs have different capabilities, the AMS can be queried to obtain a profile of its AP. A life-cycle is associated with each agent on the AP.

All these features are already provided by the Grasshopper Mobile Agent Platform. Thus, the *AMSAgent* provides a FIPA compliant interface to the Grasshopper agent management facilities. Additionally, the AMS represents the managing authority of an agent platform. If the agent platform has multiple machines the AMS represents the authority across all machines. An AMS can request an agent to `quit` (i.e. terminate all execution on its AP). The AMS has authority to forcibly terminate an agent if such a request is ignored.

The AMS possesses a table of all agents, which are currently resident on the platform. The table maps the agents Globally Unique Identifier (GUID) and their associated transport address. The GUID identifies the agent in the whole agent universe. All agents have a unique GUID. The GUID consists of the Home Agent Platform (HAP) address and the agent's name, which should be unique within the HAP. In general, the GUID looks as follow *<agent-name>@<HAP-address>*. The HAP-address can consist of the logical name or IP address of the host, on which the ACC runs, and of the port-number, at which the ACC listens.

The AMS functionality is based on the agent management functionality of Grasshopper agent platform. The key services that the AMS offers are:

- Register: When an agent is created on the agent platform, it is automatically registered in the Grasshopper Region Registry under an agent platform identifier. The AMS associates the agent platform identifier with the agent's name and saves it in it's registration base.
- De-register: When an agent is removed from the agent platform, it is automatically de-registered from the Grasshopper Region Registry. The AMS removed the agent's entry from it's registration base.
- Authenticate: The Authentication functionality is covered by the Grasshopper agent platform.

5.2.5 Directory Facilitator Agent

The Directory Facilitator is offering services similar to those of the AMS but offers additionally a search functionality. Thus the DF acts as a yellow pages directory, where agents willing to offer their services in a dynamic manner to other agents may register with a DF. The registration is done in the same way as with the AMS. Agents can deregister with the DF by calling the *deregister* service.

The DF may restrict access to information in its directory, and will verify all access permissions for agents that attempt to inform it of agent state changes. The DF does not control the agent platform life-cycle of any agent. Agents may register their services with the DF or query the DF to find out what services are offered by which agents. DFs can register with each other. Similarly, the AMS, and ACC can register with a DF.

Besides the agent platforms, FIPA has also defined the logical concept of *domains*. An agent domain is a logical grouping of agents and their services, defined by their membership in a DF. Each domain has one and only one DF, which provides a unified, complete, and coherent description of the domain. The DF lists all intelligent agents in the domain and advertises the intelligent agents existence, services, capabilities, protocols etc. An agent may be present in one or more domains via registration in one or more DFs. A domain in this context can have organizational, geo-political, contractual, ontological, affiliation or physical significance.

In Grasshopper, this functionality is already realized from the *Region Registry*. Thus, the DF realises a wrapper of these features.

Agents willing to offer their services in a dynamical manner to other agents may voluntarily register with a DF. An agent can de-register with the DF by calling the de-register service. The registration is done in the same way as that on the AMS, and the data being contributed to registration is identical to the data given to the AMS. All data delivered is then searchable by other agents by calling the DF's search service. Thus the DF acts as a yellow page directory.

The DF functionality consist of:

- Register: When it receives a *register* request, the DF will try to create an entry in the registration base with the provided agent information. Therefore, it first checks if there is already an entry for the given `:agent-name`. If an entry exists, the registration will fail and an `agent-already-registered` failure message will be send back to the agent which sent the request. If no entry exists, a new entry with the `:agent-name` as key will

be created and a confirmation message will be sent back.

- De-register: When it receives a *deregister* request, the DF will try to remove the entry corresponding to the provided agent information. If there is an entry with the given `:agent-name`, it will be removed and a confirmation message will be send back. Else, the de-registration will fail and the DF will send back an `unable-to-deregister` failure message.
- Search: search requests are handled differently, according to the provided parameters. If no `:df-search-depth` is specified or it's value is 1, the search request will be categorized as *simple* and the search will be restricted to the local registration base. If the `:df-search-depth` is grater than 1, the search request will be considered a *federated* one and the DF will try to involve also other DFs for fulfilling the request. If the `:df-description` contains the `:agent-name`, the search request will be categorized as *upon-name*. In this case the search will finish when an entry with the given key was found. If the `:df-description` contains only other search constraints as `:services`, `:type` etc., the search will continue after a matching entry was found, and all the matching entries will be returned.

5.2.6 Agent Communication Channel Agent

The Agent Communication Channel (ACC) realizes the messages exchanged among agents, including the DF and AMS. The ACC provides the default communication channel between FIPA agent platforms. An agent invokes the forward function of the ACC for exchanging the ACL messages.

The ACC component is also implemented as a Grasshopper agent. Inheriting from the class FIPAAgent it provides the message and send methods. The ACC agent is responsible for receiving FIPA ACL messages from agents and forward them to the destination agent. If the destination agent is residing on the same agent platform, the message is send to it directly. If the destination agent resides on a remote platform, a forward FIPA ACL message, addressed to the ACC from the remote platform, is built and sent to it. The ACC functionality consist of:

- Receive Messages over IIOP - The ACC provides an external IIOP interface for being able to receive string messages from other ACC over IIOP. The ACC can receive forward requests or IOR informs. When the ACC receives an IOR inform message it will save it in it's URL-IOR base using the `:sender` as key, so that it will be able to get the IOR for a given agent address.
- Forward Message: When the ACC receives a forward request from another ACC it will try to extract the contained message and send it to the destination agent. If the content is not a valid forward message, the ACC will send back a not-understood message. If the content represents a valid forward message, it will extract the contained message and check the receiver address. If the address is the same with the own address, it will try to send the contained message to the local agent. If the address is not a local address, the ACC will create a new forward message from the received message and will try to send the message to the appropriate remote ACC.
- Forward Message Local: For sending a message to a local agent, the ACC will contact the AMS in order to get the local agent platform identifier for the given agent name. If there is

no agent with the given name on the local platform, the ACC will send back an `agent-not-registered` refuse message. If the ACC can obtain the agent identifier, it will try to send it the message using the Grasshopper communication mechanism. If an error occurs, the ACC will send back a `no-communication-means` refuse massage.

- Forward Message Remote: In order to send a message to a remote agent, the ACC will first try to get the IOR of the appropriate ACC. Therefore, it will analyse the receiver address (GUID) and will extract the address of the platform. If the platform address is not already an IOR, the ACC will check it's URL-IOR base in order to get the corresponding IOR. If no entry for the given platform URL was found, the ACC will check the known IOR locations, i.e. a list of Web servers read at start-up and will try to get the IOR from there. If there is no entry for the given URL or the locations are not accessible, the it will send back an `agent-not-registered` refuse message. If the ACC obtained the IOR of the remote ACC, it will try get the ACC's reference and send it the message using the CORBA messaging mechanism. If an error occurs, the ACC will send back a `no-communication-means` refuse massage.

An agent, which intends to send a message to another agent on a different platform, has two possibilities. On the one hand, the agent can contact the local ACC. Then, the local ACC routes the messages to the remote ACC (residing on the target platform), which delivers the message to the target agent. On the other hand, the agent contacts the target ACC directly, which will then route (forward) the message to the target agent. With it, the sender agent avoids the usage of the local ACC. The address of the target platform, which is necessary for contacting the corresponding ACC can be ascertained from the target agent's identifier (GUID). The first part of the FIPA specification (FIPA97P1) specifies how forwarding is done and how logical addresses are mapped to physical addresses.

Fundamentally, the FIPA communication concept bases on the exchange of messages by ACC's. An ACC has to wait only for incoming messages, which then have to be forwarded to the addressed agents. Therefore, the forward action is the only service or method to be implemented by an ACC.

How the implementation of local communication is realized is free to the developer of the agent platform. However, it is obvious, that the simplest solution for local communication between local agents is the usage of platform native communication protocols. Accordingly, for the implementation of the FIPA platform components, Grasshopper's communication service can be applied.

FIPA prescribes the support for inter-platform communication, i.e. the communication among agents on different, possibly heterogeneous agent platforms. It is mandatory to realize the forward service supporting IIOP. The most convenient method is to use a common ORB implementation for offering this service. Thus, the ACC-object offering the service should be connected to an ORB and offers in this way automatically and transparently it's services via IIOP. Figure 17 illustrates the inter platform communication:

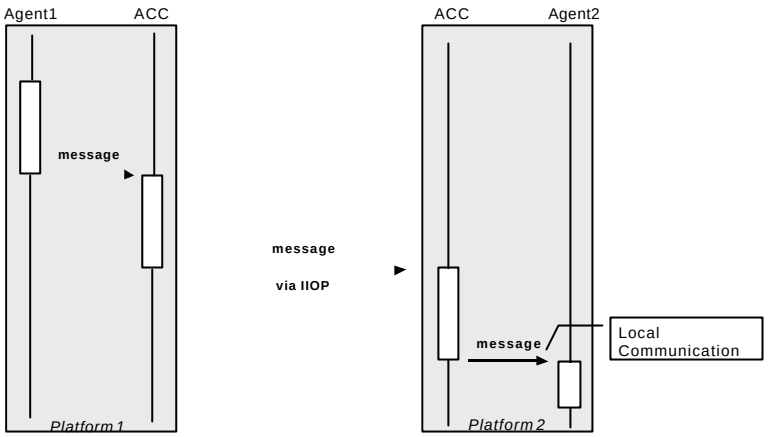

Figure 17: Inter-Platform Communication

5.3 Implementing FIPA Agents on top of Grasshopper

Due to the increasing acceptance of the FIPA standards and the resulting increased demand for FIPA conformant agent platforms, Grasshopper has been extended by a corresponding package, referred to as "*FIPA Add On*".

As described above, the main components of a FIPA compliant platform are the AMS, DF and ACC. FIPA proposes to realize these components as agents. Appropriately, for the realization on Grasshopper each of them is implemented as a single stationary Grasshopper agent respectively. To support platform interoperability as required by FIPA the ACC has to support IIOP. Grasshopper supports this protocol since the Java JDK contains a complete ORB including IIOP.

The class *FIPAAgent*, which forms the basis for the FIPA platform components, extends Grasshopper's class *StationaryAgent.* Agent developers have also to use this class as basis for their agent application. The class FIPAAgent offers two methods: *send()* and *message().* With them, the ACL-message exchange over the ACC is realized. The method send() requires a parameter of the type ACLMessage and server for dispatching of messages, whereas the message method serves for retrieval of messages.

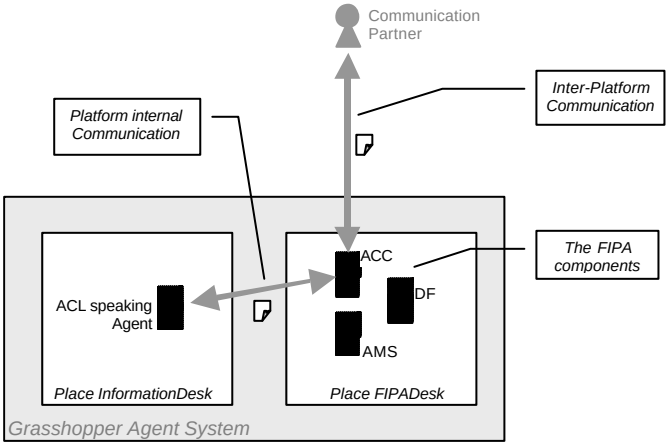

Figure 18: Realization of the FIPA Platform Components on Top of Grasshopper

The following class diagram shows the implementation of the FIPA platform on top of Grasshopper. The class FIPAAgent which will be the basis for the FIPA platform components, extends the FIPARegistrableAgent class, which itself inherits from Grasshopper's class MobileAgent. Agent developers have also to use this class as basis for their agent application. The FIPARegistrableAgent offers methods for automatically registration and deregistration with the local DF.

The class FIPAAgent offers two methods: send() and message() for the ACL-based communication with other agents. The method send() requires a parameter of the type FIPAACLMessage and server for dispatching of messages, whereas the message method serves for retrieving of messages. FIPA recommends that for asynchronous communication among agents a message queue is applied. Such message stack can be considered as a enhancement of class FIPAAgent in future versions.

To implement an agent, which intends to act as a FIPA agent, the agent class has to inherit from the class FIPAAgent by means of writing a Java class extending FIPA.FIPAAgent. The class FIPAAgent is itself extending the class de.ikv.grasshopper.agency.StationaryAgent, thus being a regular grasshopper stationary agent. The extension of the stationary agent mainly consists of two methods. These are:

- public void message(FIPAACLMessage msg) —This method has to be overwritten by a FIPA agent in order to be able to receive FIPA ACL messages sent by other agents through the ACC.
- public void message(String msg) – this method is the standard communication interface of all FIPA agents. FIPA agents receive ACL messages encoded in strings through this method. For the definition of the agent's behavior this method has to be overwritten
- public void send(ACLMessage msg) – this method simplifies agent communication over the ACC. Calling this methods results in establishing a connection to the local ACC by means of grasshopper communication and sending a forwarding request with the message *msg* to the agent addressed in the message. In this way the communication is completely ACC transparent to the agent developer. Naturally, the communication with the ACC and other agents can be done completely with native grasshopper's communication service. Then, the method *send* does not have to be used.

Apart from the regular Grasshopper agent methods, which have to be implemented for each Grasshopper agent such as the *live* method, the agent developers have to implement or overwrite the *message* method, whereas the *send* method can be simply used.

Additionally, standard FIPA agent should be in position to send and receive ACL/XML messages. For that reason, when a message has been sent to an agent, the agent first needs to parse the incoming ACL/XML message by deploying a standard ACL parser. For that reason a FIPA ACL Parser is provided. The parser gets as input an ACL/XML message string, parses it and produces a query object called FIPAACLMessage. The ACLMessage class provides operations for getting and setting the type of message, the sender, receiver, content, etc.

As soon as the incoming messages have been parsed from the ACL parser, the content of the message, which is described in XML, should also be parsed. For that reason a specialised XML parser has been developed (see next chapters). The XML parser provides all the necessary operations for retrieving information that has been formed in terms of XML content. Further

details about the XML ontologies specified and used in the context of this thesis are provided in subsequent chapters.

Additionally, when an agent wants to send an ACL/XML message to another agent he should always compose an ACL/XML message. The message composers are tightly coupled with the ontologies and will be explained in the subsequent chapters. In general, the responsibility of message composer is to produce a legitimate ACL/XML string. The agent can then send the message to another agent by utilising the send operation provided by the core FIPA agent class.

In the following UML class diagram the main classes involved in the development of a standard FIPA agent are provided.

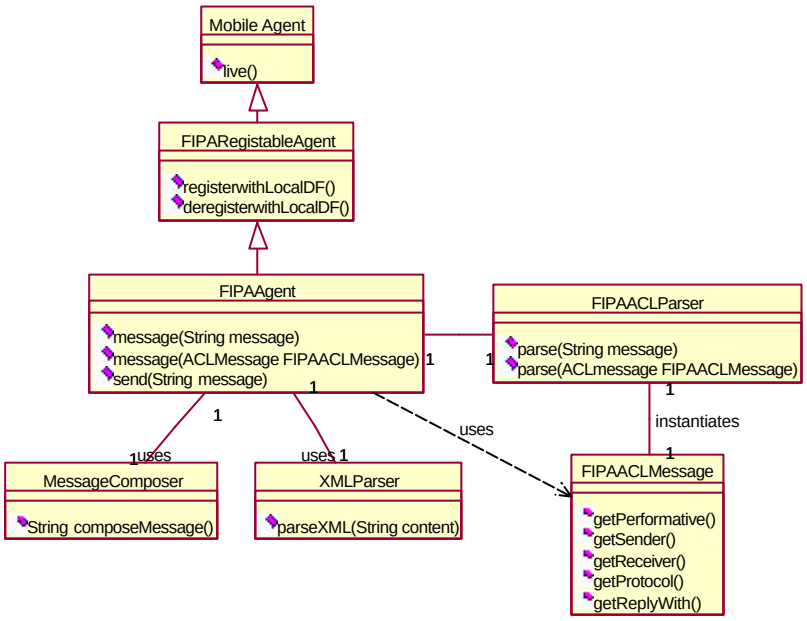

Figure 19: FIPA Agent Class Diagram

In the following chapters, certain standard FIPA agents are specified. The internal class diagram of this agent is based on the above described class model.

5.4 Summary

This chapter presents extensively the Agent Platform layer and the services that will be deployed. More specifically, this chapter is split it into two parts the core agent management platform which the Grasshopper Agent Platform, and the FIPA add on services. In both cases, an analytical description of the services and capabilities offered are presented. Due to the fact that all the agents under design and analysis are FIPA compliant agents, certain details concerning the implementation of them in the Grasshopper agent platform are presented.

Chapter 6: Virtual Marketplaces

6.1 Introduction

A Virtual Marketplace is a third party administrative domain that provides matchmaking services to the VE partners. The Virtual marketplace enables *VE candidate partners* to register and administer service offers in relation to certain service types and *VE representatives* to search for potential partners that can provide particular business processes associated with existing service types.

Every business process registered in the virtual marketplace is associated with a service type. In general, service types describe in a consistent way the interface of business processes. Service types mainly managed by the virtual marketplace administrator. The *service type management* includes creation, deletion, modification and retrieval of service types.

VE candidate partners that want to register their process offerings in the marketplace should always create a service offer in association with an existing service type and register it to the virtual marketplace. A service offer is actually an instance of a service type where certain properties have given certain values. Service offers are managed individually from each domain in a private manner. The *management of service offers* includes the registration of an offer, the withdrawal, the listing of offers and the modification of an offer.

Finally, VE representatives or partners that want to find suitable partners that can provide a particular process retrieve from the marketplace all the registered offers that satisfy certain constraints. The *service offer retrieval management* process actually includes the retrieval of offers that satisfy certain constraints.

Therefore, the basic services provided by the marketplace are *service type management*, *service offer management*, and *service offer retrieval management*. Each one of these services are provided by individual FIPA compliant agents, namely the:

- **Service Type Agent (STA)** responsible for the management of service types and more specifically for the addition, removal, listing, and modification of service types,
- **Service Offer Agent (SOA)** responsible for the management of service offers and more specifically for the registration, withdrawal, description, and modification of service offers,
- **Service Offer Retrieval Agent (SOR)** responsible for the retrieval of offers associated with a service type based on some constraints.

These three agents are FIPA compliant agent, i.e. they communicate with other agents by exchanging standard FIPA ACL/XML messages. The content of these messages is described in XML and it follows the Virtual Marketplace Ontology. The Virtual Marketplace Ontology describes the set of input and output messages that the marketplace agents can exchange with other agents. The whole specification of the Virtual Marketplace Ontology is given at the end of the thesis in Annex. Agents that want to communicate with the STA, SOA, or SOR agents should formulate and understand messages based on this ontology.

The layered reference architecture of the virtual marketplace is depicted in the following figure. It is actually the specialisation of the overall reference architecture described in previous chapters for the virtual marketplace.

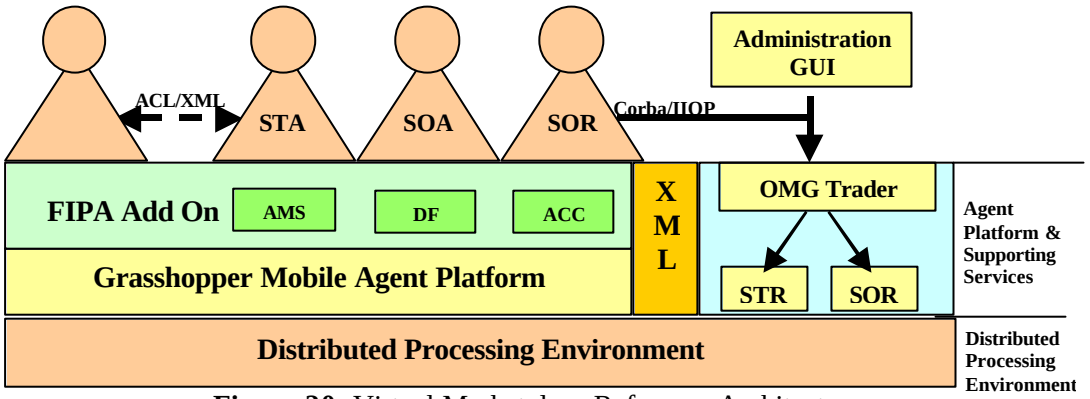

Figure 20: Virtual Marketplace Reference Architecture

The key design principles of the agent-based Virtual Marketplaces are being influenced by the corresponding concepts of the OMG Trader. This is actually the objective of the thesis, i.e. to develop an agent-based virtual marketplace by integrating a standard OMG Trader. For that reason, a standard OMG Trader, as a basis for the development and testing of the virtual marketplace, has been used. In principle, the OMG-Trader is a CORBA object and could not be used directly by the different FIPA compliant agents in an autonomous and message-based way. Due to this reason, the specialised STA, SOA, and SOR FIPA agents, as well as, the virtual marketplace ontology have been specified and developed. These agents are actually offering the basic functionality of the OMG-Trader in a FIPA compliant way to other agents.

The general format of a standard FIPA ACL/XML request based on the virtual marketplace ontology is the following:

```
(request
   :sender Provider Negotiation Agent
   :receiver STA or SOA
   :content ( <VMPMessage>   // this is a VMP message
              <STAMessage or SOAMessage>     //this is a message for the STA
             agent
              <STARequest or SOARequest RequestId="abc"> // id of the request
              <command type>  // the command requested;
               .......          //
              </command type>//
              </STARequest or SOARequest>
             </STAMessage or SOAMessage>
             </VMPMessage>
             )
   :protocol fipa-request
   :reply-with inform
)
```

The general format of a standard FIPA ACL/XML inform response to a particular request is the following:

```
(inform
   :sender STA or SOA
   :receiver Provider Negotiation Agent
   :content ( <VMPMessage>   // this is a VMP message
              <STAMessage or SOAMessage >     //this is a message for the STA
             agent
              <STAResonse or SOAResonse RequestId="abc"> // id of the request
               <command type>  // the command requested;
               .......          //
              </command type>//close everything according to the DTD file
              </STARequest or SOARequest >
              </STAMessage or SOAMessage >
             </VMPMessage>
             )
   :protocol fipa-request
   :reply-with inform
)
```

In general, VE candidate partners that want to create or administer new service types in the marketplace should always refer to the appropriate service type name. In that case, a suitable agent migrates to the virtual marketplace, composes a FIPA ACL/XML request, and sends it to the STA agent. The communication protocol among the agents is based on the standard FIPA-Request-Response protocol (FIPA98). The STA receives the request, parses it from the ACL and XML parser, checks the type of the request and decides which action is required. In the sequel, the STA performs the request by deploying the Service Type Repository, generates an ACL/XML inform message and sends it to the requestor agent. The requestor agent, upon receipt of the message, parses it first from the ACL and XML parser, checks the response and migrates back to the VE candidate domain to inform its domain.

In a similar way, VE candidate partners willing to register new service offers in the marketplace or administer them should always refer to the appropriate service type name. In that case, in a similar manner like the STA agent, an instance of a suitable agent migrates to the marketplace, composes a FIPA ACL/XML request and sends it to the SOA agent. In the sequel, the SOA receives the message, parses it from the ACL and XML parser, checks the type of the request and decides which action is required. Afterwards, the SOA performs the request, by deploying

the Service Offer Repository, composes an ACL/XML inform message and sends it back to the requestor agent. The requestor agent receives the message, parses it first from the ACL and XML parser, checks the reply of the SOA and migrates back to the VE candidate partner to inform its domain.

In addition to the generic entities that a FIPA compliant agent has, the internal architecture of a virtual marketplace agent consists of the following key components:

- **VMP XML Parser**: responsible for parsing the content of the FIPA ACL messages based on the virtual marketplace ontology,
- **VMP Message composer**: responsible for composing the appropriate response FIPA ACL-XML messages that will be sent back to the requestor agents. The structure of the message is based on the marketplace ontology,
- **Decision manager**: responsible for controlling the basic operations of the agent, communicating with the STR, SOA and the OMG-Trader. According to whether the agent is STA, SOA or SOR this module is different in order to perform the appropriate operations.

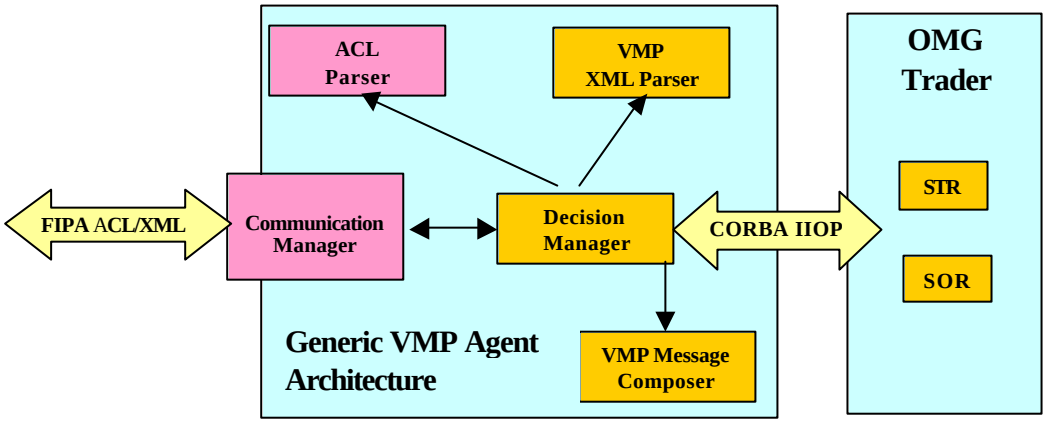

Figure 21: Generic Virtual Marketplace Agent Internal Architecture

In the following sections, the STA, SOA, and SOR agents are further analyzed and more details about their functionality and structure are provided.

6.2 Service Type Management

One of the basic concepts of the Virtual Marketplace is the service type. A service type has a unique service type name and one or more named properties. The named properties are (name, value) pairs that represent behavioural, non-functional and non-computational aspects of a service. For example, a service type for a video-conferencing service might have the cost of the service, the payment mode, the QoS requirements, the name of domain that offers the service, etc, as named properties. Additionally, the named properties can also have specific modes. These modes are *normal*, meaning that the value for this property is not required during the service offer process, *read-only*, meaning the value of a named property can not be changed, or *mandatory* meaning that a value for this named property is required.

Service types can also be derived from other service types. A service type that inherits existing service types inherits also the properties specified for these super types. The concept of the inheritance is exactly the same like the one used in the object oriented systems. In such a way, value added service types can be created. The main reason for the inheritance concept is the re-usability and enhancement of existing service types.

According to previous descriptions, a service type has a name, a set of zero or more property descriptions, and a set of zero or more super types from which it inherits its extra common properties. A property description has a property name, a status, a type, e.g.. string, integer, or sequence of strings or integers. A super type is the name of an existing service type. Service Types can be created and managed either by the administrator of the marketplace or by the Provider Negotiation Agent (PNA). The following definitions can easily translated into the following DTD format of a service type.

```
<!ELEMENT Service Type (type, propdescs, super_types)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT if_name (#PCDATA)>
<!ELEMENT propdescs (propdesc*)>
<!ELEMENT propdesc (prop_name, (normal | readonly | mandatory |
readonly_mandatory), (string | integer | float | boolean | stringseq |
integerseq | floatseq | booleanseq) )>
<!ELEMENT prop_name (#PCDATA)>
<!ELEMENT normal EMPTY>
<!ELEMENT readonly EMPTY>
<!ELEMENT mandatory EMPTY>
<!ELEMENT readonly_mandatory EMPTY>
<!ELEMENT super_types (super_type*)>
<!ELEMENT super_type (#PCDATA)>
```

Service Types are stored and maintained by the Service Type Repository (STR), which is located inside the marketplace domain. For every virtual marketplace only one STR exists. The main reason fro that is that the services types managed and administered by this marketplace should be stored in a central storage system for data consistency purposes. For every service type a service type object is created and stored into the STR. A service type object is actually an abstraction, in object oriented terms, of a service type that provides certain operations to access the information stored inside the object. Every service type and, consequently a service type object, is identified inside the STR by a unique service type name. All these objects are stored into and maintained by the STR. Using the service type name as a unique key to retrieve the service type, retrieval of service type objects is performed easily and effectively. The class model and the basic operations of the STR is depicted in the following Figure 22. The main operations of the STR are to create a new service type object, to delete it, and to modify it. The entity that deploys these operations is the Service Type Agent (STA).

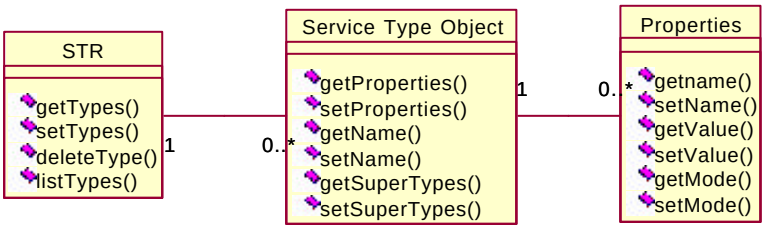

Figure 22: Service Type Repository Class Model

The STA agent is responsible for the management of service types and the main operations that this agent provides are:

- addition of a service type to the STR,
- removal of a service type from the STR,
- listing of existing service types from the STR,
- description of a service type from the STR,

When the STA agent gets an ACL/XML request for an addition of new service type, it first parses the content of the incoming message from the ACL and XML parser and checks the type of the request. Since the request is the creation of a new service type, the Decision Manager requests from the STR to create a new service type object. Then, the Decision Manager inserts the information included in the message and requests from the STR to store it internally. As soon as the request has been fulfilled, the agent composes a response ACL/XML message and sends it back to the requestor. An example of the content of the FIPA ACL/XML message concerning service type addition is the following.

```
<VMPMessage>
<STAMessage>
<STARequest RequestId="abc">
<AddType>
<type> testType1 </type>
<if_name> testTypeIf1 </if_name>
<propdescs>
<propdesc>
<prop_name> testprop1 </prop_name>
<normal/>
<string/>
</propdesc>
<propdesc>
<prop_name> testprop2 </prop_name>
<mandatory/>
<integer/>
</propdesc>
</propdescs>
<super_types>
</super_types>
</AddType>
</STARequest>
</STAMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

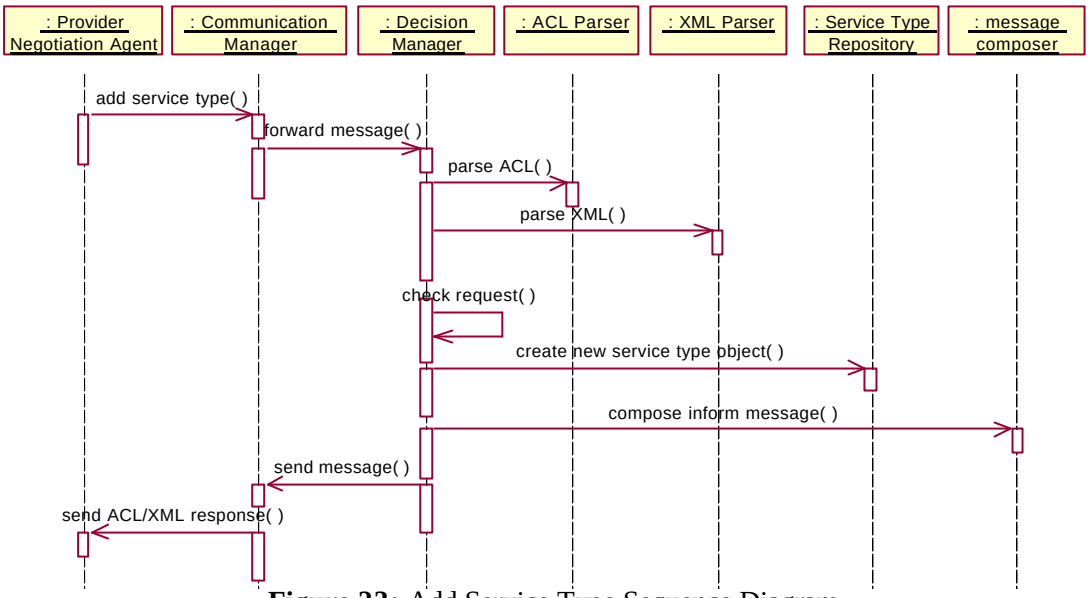

Figure 23: Add Service Type Sequence Diagram

In a similar way like the creation of a new service type, when the STA agent gets an ACL/XML request for the deletion of an existing service type, it first parses the content of the incoming message from the ACL and XML parser and checks the type of the request. Since the request is the deletion of an existing service type, the Decision Manager requests from the STR to locate the corresponding service type object and then to delete it. As soon as the request for the deletion of an existing service type has been fulfilled, the agent composes a response ACL/XML message and sends it back to the requestor. An example of the content of the FIPA ACL/XML message concerning service type removal is the following.

```
<VMPMessage>
<STAMessage>
<STARequest RequestId="removetype1">
<RemoveType>
<type> testType1 </type>
</RemoveType>
</STARequest>
</STAMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

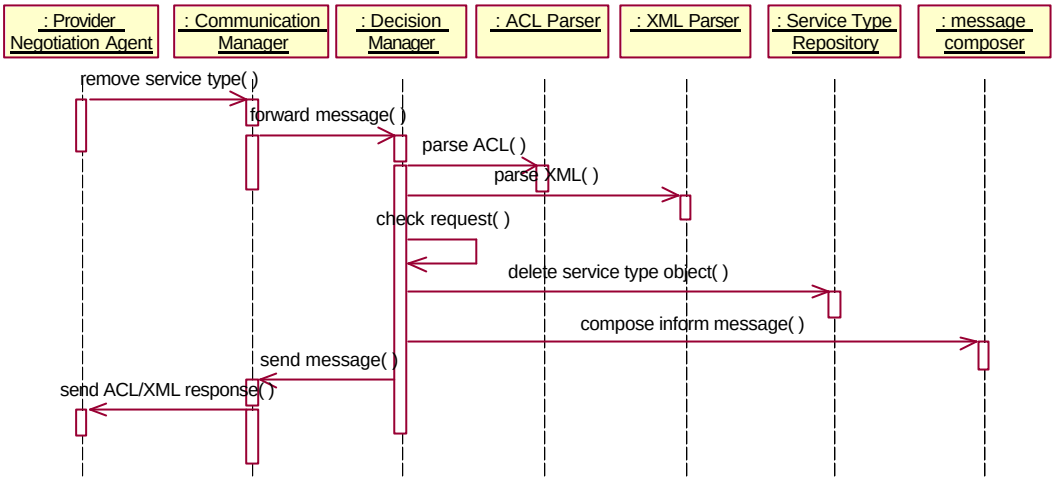

Figure 24: Remove Service Type Sequence Diagram

When a VE candidate partner wants to get the list of existing service types stored into the marketplace, a request for list of existing service types is generated. In that case, the STA agent parses the incoming message and checks the request. Since the request is to list all the existing service types, the Decision Manager conducts the STR and asks from it to return the whole list of the existing service types. In the sequel, the agent composes the appropriate ACL/XML message based on the virtual marketplace ontology and sends it back to the requestor. An example of the content of the FIPA ACL/XML message concerning list service type operation is the following.

```
<VMPMessage>
<STAMessage>
<STARequest RequestId="listtype01">
<ListTypes>
<all/>
</ListTypes>
</STARequest>
</STAMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

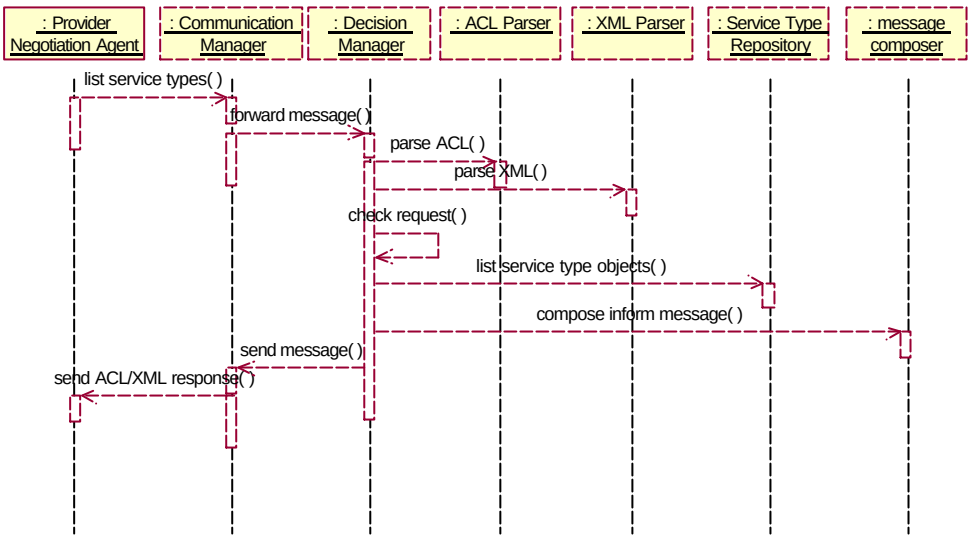

Figure 25: List Service Types Sequence Diagram

Finally, when VE candidate partners want to get all the information, namely service type name and a set of named properties about an existing service stored into the marketplace, they should generate a request for describe service type. In that case, the STA agent parses the incoming message and checks the request. Since the request is to describe an existing service type, the Decision Manager conducts the STR and gets the corresponding service type object. In the sequel, the Decision Manager gets the information stored into the object and composes the appropriate ACL/XML message based on the virtual marketplace ontology and sends it back to the requestor. An example of the content of the FIPA ACL/XML message concerning describe service type operation is the following.

```
<VMPMessage>
<STAMessage>
<STARequest RequestId="describetype01">
<DescribeType>
<type> testType1 </type>
</DescribeType>
</STARequest>
</STAMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

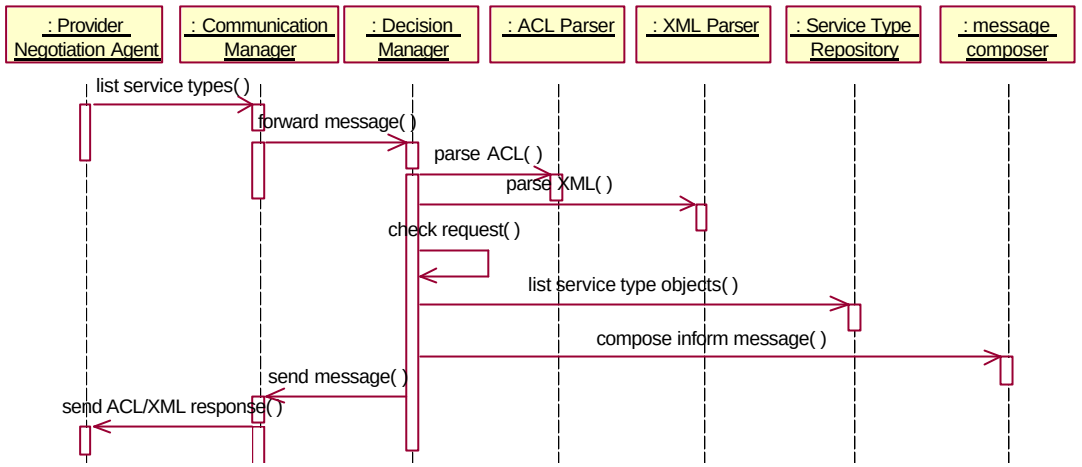

Figure 26. Describe Service Type Sequence Diagram

6.3 Service Offer Management

Another basic concept of the Virtual Marketplace is the service offer. A service offer is actually an instantiation of a specific type, i.e. for all the named properties of a service type certain values are provided. Therefore, for every service type zero or more service offers are associated. A service offer about a service type has a unique id and is related to a specific VE candidate partner that registers the offer. Since properties have different modes, like normal, read-only, or mandatory, service offer registration should provide values to all mandatory and normal values. Properties having the mode of read-only have pre-specified values that could not be changed during service offer registration.

Service offers can be created and managed either by the administrator of the marketplace or by the Provider Negotiation Agents (PNA) located into the VE candidate domains. Service offers are stored in the Service Offer Repository (SOR), which is located inside the marketplace. For every marketplace only one SOR exists. Inside the SOR, every Service Offer is identified by a unique service offer id and is always associated to a corresponding service type. For every service offer a service offer object is created and stored into the SOR. A service offer object is actually an abstraction, in object-oriented terms, of a service offer that provides certain operations to access the information stored inside the object. Using the service offer id as a unique key to retrieve the service offer, retrieval of service offer objects is performed easily and effectively. The structure of the SOR is depicted in the following Figure 27. The main operations of the SOR is to register a new service offer object, to delete it, to modify an existing service offer, and to get the property values of an existing service offer. The entity that deploys these operations is the Service Offer Agent (SOA).

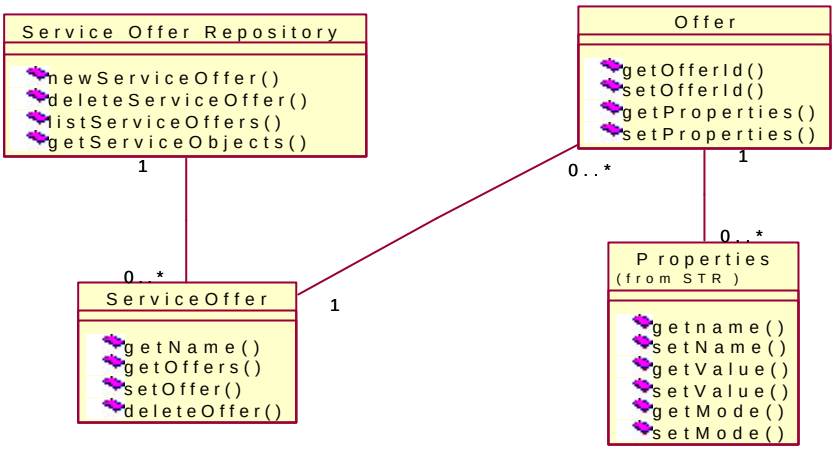

Figure 27: Service Offer Repository Class Model

The SOA agent is responsible for the management of service offers and the main operations that this agent provides are:

- register a service offer,
- withdraw a service offer,
- modify a service offer,
- describe a service offer.

When the SOA agent gets an ACL/XML request for a registration of new service offer, it first parses the content of the incoming message from the ACL and XML parser and checks the type of the request. Since the request is the registration of new service offer, the Decision Manager, based on the service type name, requests from the SOR to create a new service offer object. Then, the Decision Manager inserts the information included in the incoming message and requests from the SOR to store it internally. The SOR returns back to the Decision Manager a unique registration id that identifies uniquely this particular offer in relation to the service type. As soon as the request has been fulfilled, the agent composes a response ACL/XML message with the unique service offer id and sends it back to the requestor. An example of the content of the FIPA ACL/XML message concerning register service offer operation is the following.

```
<VMPMessage>
<SOAMessage>
<SOARequest RequestId="addoffer01">
<ExportOffer>
<type> testType1 </type>
<agent> testAgent1@fokus.gmd.de </agent>
<properties>
<property>
<pname> testprop1 </pname>
<pvalue>
<string/>
<value> teststringvalue1 </value>
</pvalue>
</property>
<property>
```

```
<pname> testprop2 </pname>
<pvalue>
<integer/>
<value> 1234561 </value>
</pvalue>
</property>
</properties>
</ExportOffer>
</SOARequest>
</SOAMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

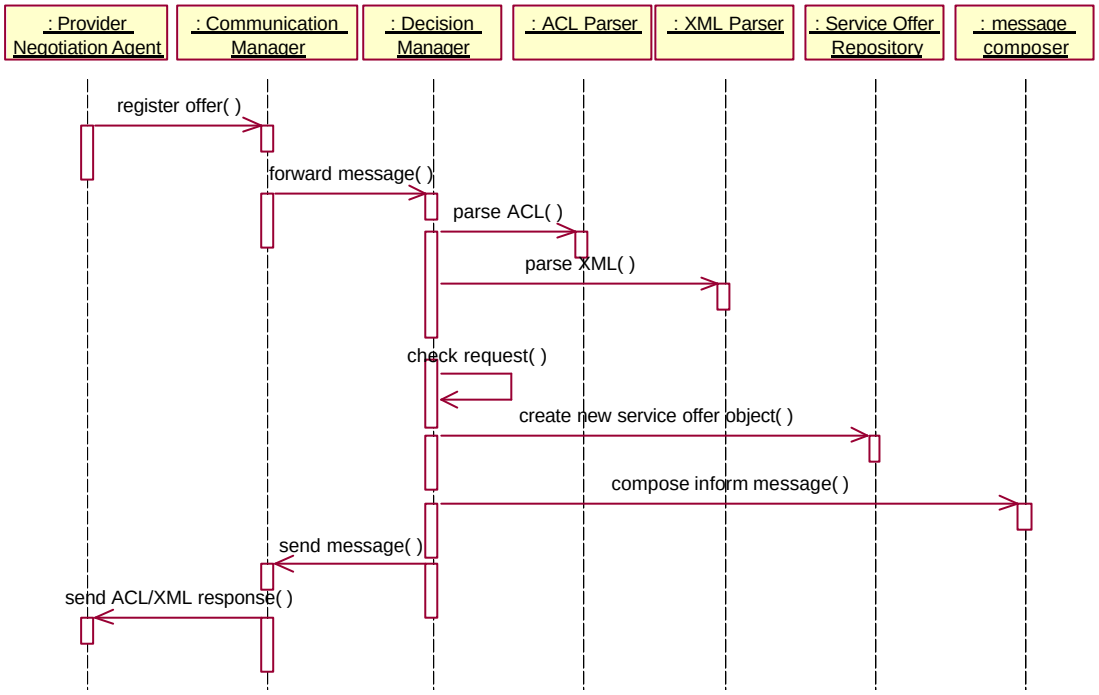

Figure 28: Register Service Offer Sequence Diagram

If a VE candidate partner wants to delete an existing service offer, a request for service offer withdrawal is generated and sent to the SOA agent. This request should refer to the corresponding service type and also include the unique registration id that has been provided to this domain during offer registration. This means that only this domain can withdraw an existing service offer. When the SOA agent gets the request, it initially parses the content of the incoming message from the ACL and XML parser and checks the type of the request. Since the request is the withdrawal of service offer, the Decision Manager, based on the service type name and the unique registration id, requests from the Service Offer Repository to delete the service offer object. As soon as the SOR deleted the object, the Decision Manager composes a response ACL/XML message and sends it back to the requestor. An example of the content of the FIPA ACL/XML message concerning withdraw service offer operation is the following.

```
<VMPMessage>
```

```
<SOAMessage>
<SOARequest RequestId="removeoffer01">
<WithDrawOffer>
<offerid> testType1/R0 </offerid>
</WithDrawOffer>
</SOARequest>
</SOAMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

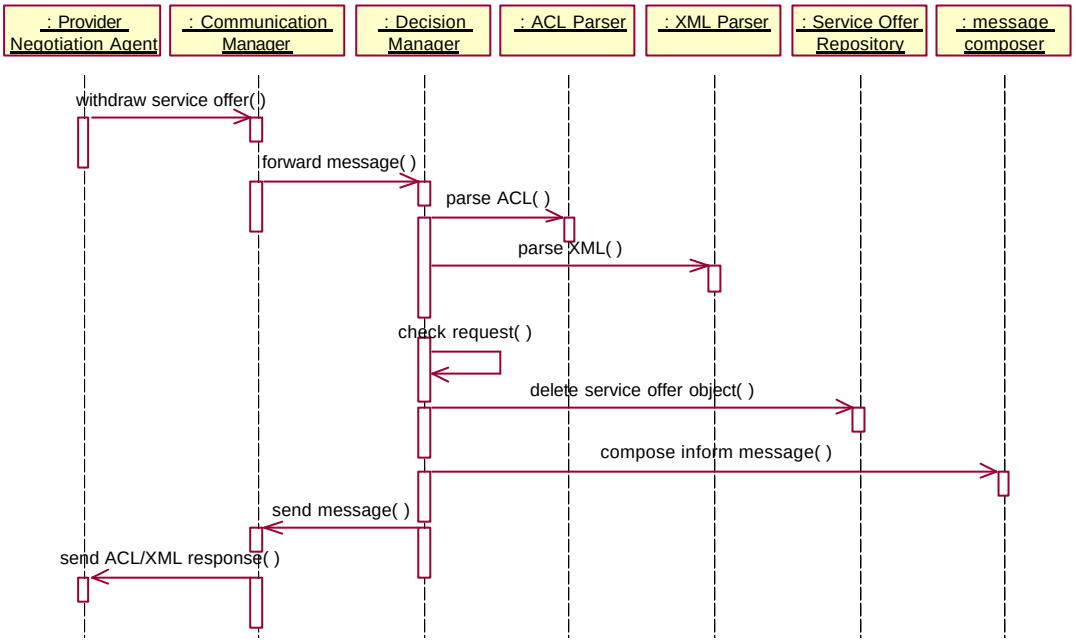

Figure 29: Withdraw Service Offer Sequence Diagram

If a VE candidate partner wants to modify an existing service offer, a request for service offer modification is generated and sent to the SOA agent. This request should refer to the corresponding service type and also include the unique registration id that has been provided to this domain during offer registration and also the new values for specific properties of the service type. When the SOA agent gets the request, it initially parses the content of the incoming message from the ACL and XML parser and checks the type of the request. Since the request is the modification of an existing service offer, the Decision Manager, based on the service type name and the unique registration number, requests from the SOR to locate the service offer object from the SOR. In the sequel, the Decision Manager modifies the values of properties included into the message and stores the service offer object again in the SOR. In that case the SOR do not generate a new service offer registration id. As soon as the SOR modified the object, the Decision Manager composes a response ACL/XML message and sends it back to the requestor. An example of the content of the FIPA ACL/XML message concerning modify service offer operation is the following.

```
<VMPMessage>
<SOAMessage>
<SOARequest RequestId="modifyoffer02">
```

```
<ModifyOffer>
<offerid> testType2/R0 </offerid>
<delete>
<prop_name> testprop1 </prop_name>
<prop_name> testprop3 </prop_name>
</delete>
<modify>
<properties>
<property>
<pname> testprop2 </pname>
<pvalue>
<integer/>
<value> 7654321 </value>
</pvalue>
</property>
<property>
<pname> testprop4 </pname>
<pvalue>
<integerseq/>
<seq_value>
<value> 9999999 </value>
<value> 8888888 </value>
<value> 7777777 </value>
</seq_value>
</pvalue>
</property>
</properties>
</modify>
</ModifyOffer>
</SOARequest>
</SOAMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

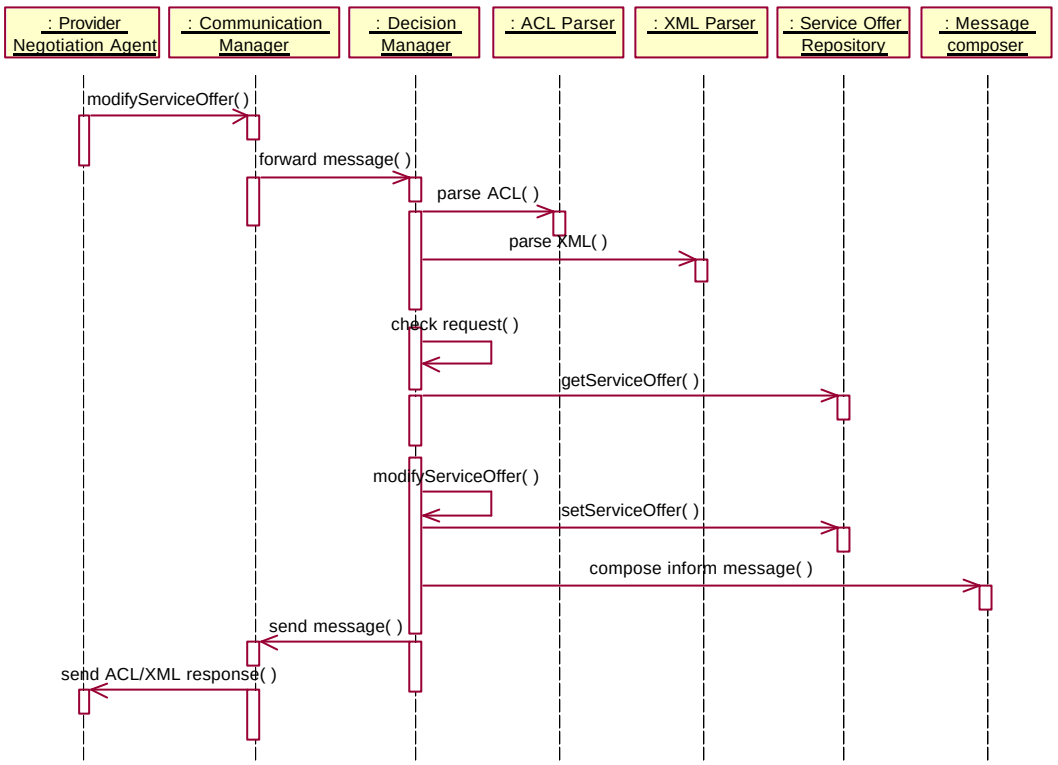

Figure 30: Modify Service Offer Sequence Diagram

Finally, if a VE candidate partner wants to get the properties and values of an existing service offer, a request for service offer description is generated and sent to the SOA agent. This request should refer to the corresponding service type and also include the unique registration id that has been provided to this domain during offer registration. When the agent gets the request, it initially parses the content of the incoming message from the ACL and XML parser and checks the type of the request. Since the request is the description of an existing service offer, the Decision Manager, based on the service type name and the unique registration number, requests from the SOR to locate the corresponding service offer object from the SOR. In the sequel, the Decision Manager gets the values of the properties included into service offer, composes a response ACL/XML message and sends it back to the requestor. An example of the content of the FIPA ACL/XML message concerning describe service offer operation is the following.

```
<VMPMessage>
<SOAMessage>
<SOARequest RequestId="describeoffer01">
<DescribeOffer>
<offerid> testType1/R0 </offerid>
</DescribeOffer>
</SOARequest>
</SOAMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

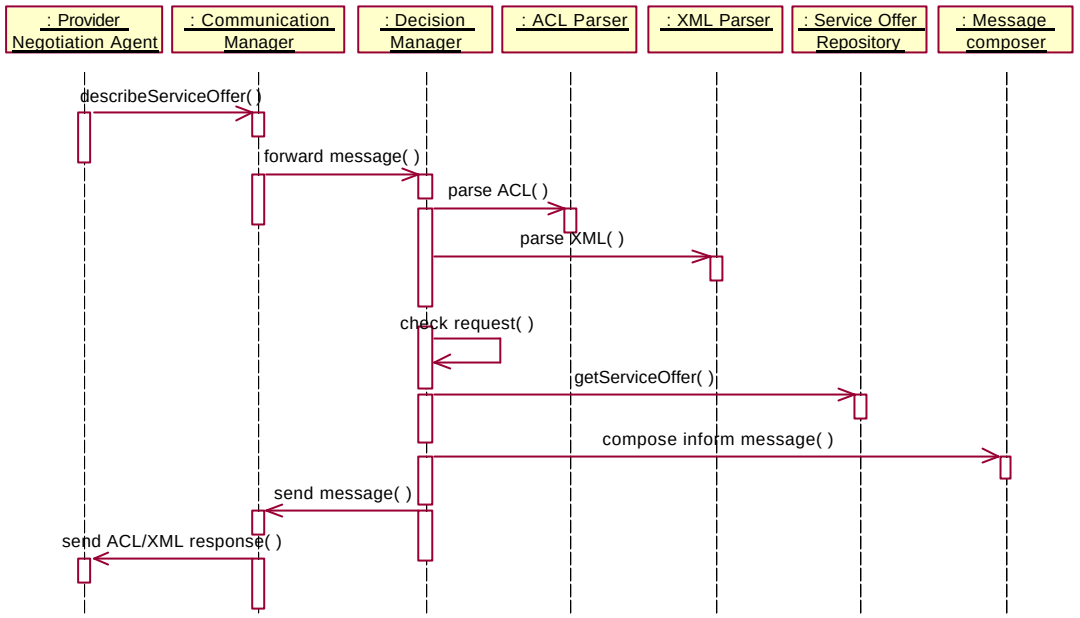

Figure 31: Modify Service Offer Sequence Diagram

6.4 Service Offer Retrieval Management

Whenever a VE representative or partner is looking for a potential VE partner that can provide a specific business process, it always uses the virtual marketplace to find out the potential providers. The selection of the VE candidate partners is done based on some selection criteria. After a set of VE candidate partners have been selected, the negotiation process starts among the VE partner and the VE candidate partners. The result of the negotiation process is the selection of the best VE candidate partner that satisfies certain requirements imposed by the VE representative or partner.

The virtual marketplace agent, responsible for the service offer retrieval management, is the Service Offer Retrieval (SOR) agent. The main operation, that this agent provides, is retrieval of VE candidate partners that can provide a certain business process related to a specific service type and satisfy certain retrieval constraints

The retrieval constraints are being specified in the OMG Constraint Language (CL) [OMG Constraint Language]. More specifically, the retrieval constraints are logical expressions relating the properties of the service type with certain values. The logical operators, supported by the OMG Constraint Language, are all the well-known logical operators like, “>”, “>=”, “AND”, “NOT”, etc. For example, if a service type A has three named properties *X1*,*X2* and *X3*, then a legitimate[11] retrieval constraint would have been [(“X1=”10”) AND (“X2”<=”Vag”) AND (NOT (“X3<”45”))].

[11] Legitimate in the sense that the expression is compatible with the grammar and syntax of the OMG Constraint Language

The OMG Constraint Language is a standard mechanism for specifying constraints in the OMG Trader. Every standard OMG Trader should provide support for the Constraint Language in terms of a Constraint Language Parser (CLP) that interprets and acts upon specific retrieval constraints. The CLP gets as input a retrieval constraint specified in the Constraint Language and, if the expression is syntactically correct, creates an expression tree. The expression tree has as nodes the binary operators of the logical expression and as leafs the named properties and values. Figure 32 depicts the expression tree for the previous presented example. In the sequel, the expression tree is traversed with the pre-order method. The pre-order algorithm is working like this: *"unless a leaf has not been found yet, traverse the left node. If a leaf has been found, return back and traverse the father of the node, and then continue with the right node"*. During the traverse process, reduction of the search space is done, i.e. the identification of all the service offers that can satisfy the specified retrieval constraints. The reduction of the search space is done like this: "*initially select all the service offers related with the service type name from the Service Offer Repository (SOR). In the sequel, when a simple logical expression found from the expression tree, locate all the service offers that satisfy this logical expression. This list will be used in subsequent search space reduction passes*". Based on this simple search space reduction approach, the service offers that satisfy the retrieval constraints are identified. The CLP and search space reduction mechanisms are components included into the standard components of the OMG Trader. In the context of this thesis, only deployment of this components is done.

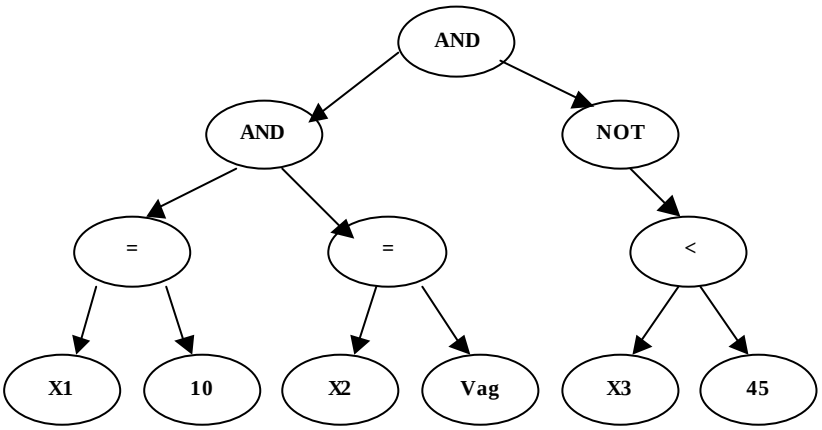

Figure 32: Expression Tree Representation

The selection of potential VE candidate partner form the virtual marketplace is performed in the following manner. Initially, the VE representative or VE partner domain, during a business process execution, creates a Requestor Negotiation Agent (RNA), informs him about the requested business process and the retrieval constraints that should be satisfied, and sends him to the virtual marketplace to search for potential VE candidate partners. The RNA uses the FIPA compliant request-response protocol and the virtual marketplace ontology to communicate with the SOR agent. More specifically, the RNA migrates to the virtual marketplaces, creates a FIPA ACL/XML request message, and sends it to the SOR agent. The message specifies the requested service type name and the corresponding retrieval constraints for the selection of partners.

When the SOR agent gets the request, it initially parses the content of the incoming message from the ACL and XML parser and checks the type of the request. Since the request is the retrieval of existing service offers from the Service Offer Repository, the Decision Manager

conducts the Constraint Language Parser (CPL) to syntactically parse the retrieval constraint. If the retrieval constraint expression is not correct, an error message is returned to the RNA agent. If the constraint is syntactically correct, then the Decision Manager asks from the Retrieval Manager to locate the service offers, which exist in the Service Offer Repository and satisfy the requested constraints. The result of the retrieval is a list of service offer objects. Then, the Decision Manager processes the list and, with the help of message composer, creates a FIPA ACL/XML message with the response. This response message is sent to the RNA agent. An example of the content of the FIPA ACL/XML message concerning retrieval of service offers operation is the following.

```
<VMPMessage>
<SORMessage>
<SORRequest RequestId="query01">
<Query>
<type> testType1 </type>
<constraint>
testprop1=="teststringvalue1" AND testprop2>=1234561”
</constraint>
</Query>
</SORRequest>
</SORMessage>
</VMPMessage>
```

The previously described steps are further explained in the following sequence diagram.

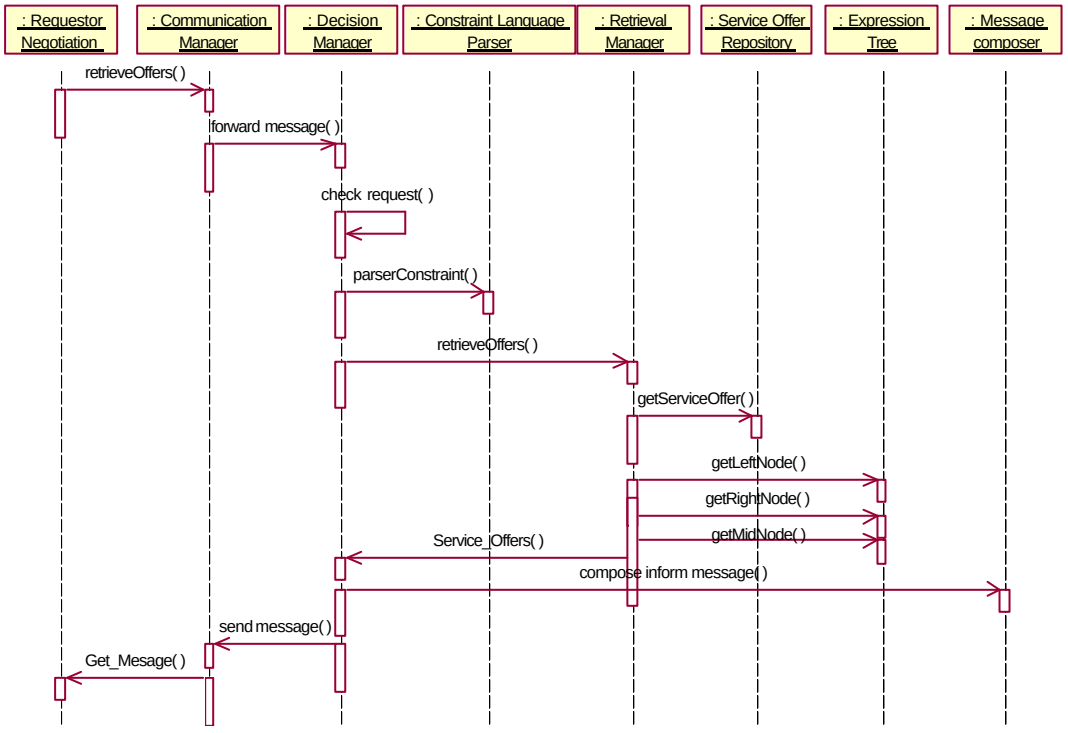

Figure 33: Service Offer Retrieval Sequence Diagram

6.5 Virtual Marketplace Administration

The virtual marketplace is administered from a human operator called the Administrator. The main responsibilities of the virtual marketplace administrator are service type management operations. The administrator uses an administration Graphical User Interface (GUI) that enables him to perform in a easy and user friendly way its main operations.

The main operations that the administrator can perform are similar with the ones that are provided by the STA agent. These operations are:

- addition of a new service type to the virtual marketplace,
- removal of a service type from the virtual marketplace,
- listing of existing service types to the virtual marketplace,
- description of a service type from the virtual marketplace.

The use case diagram of all the operations that the administrator can perform is depicted in the following Figure 34. In this diagram, except the key operations that can be performed, a set of sub-operations is also identified. For example, the add service type operations is supported by a set of sub-operations, namely the provision of a new service type name, the insertion of a new property or properties, the addition of the inheritance relationships with existing services types, and the storage of the new service type in the Service Type Repository (STR).

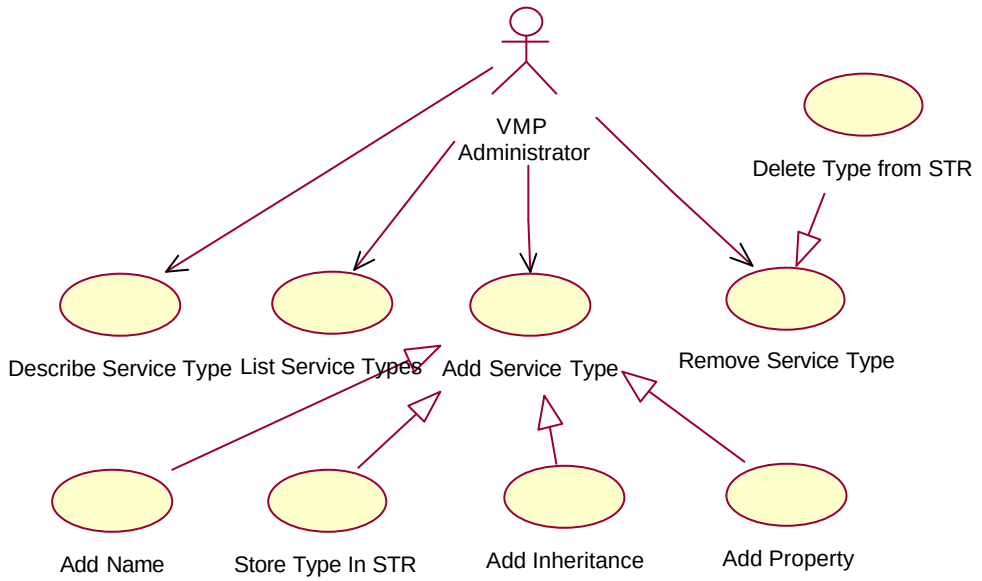

Figure 34: VMP Administration Use Case Diagram

The main entities of the virtual marketplace administration GUI are the:

- **Administrator GUI Manager**: responsible for the provision of the graphical interface to the human operator,
- **Service Type Repository (STR)**: responsible for the provision of the core operations of the service type management and the persistent storage of the service types,

The class diagram of the administration GUI and the relationships between these two entities are depicted in the following Figure 35.

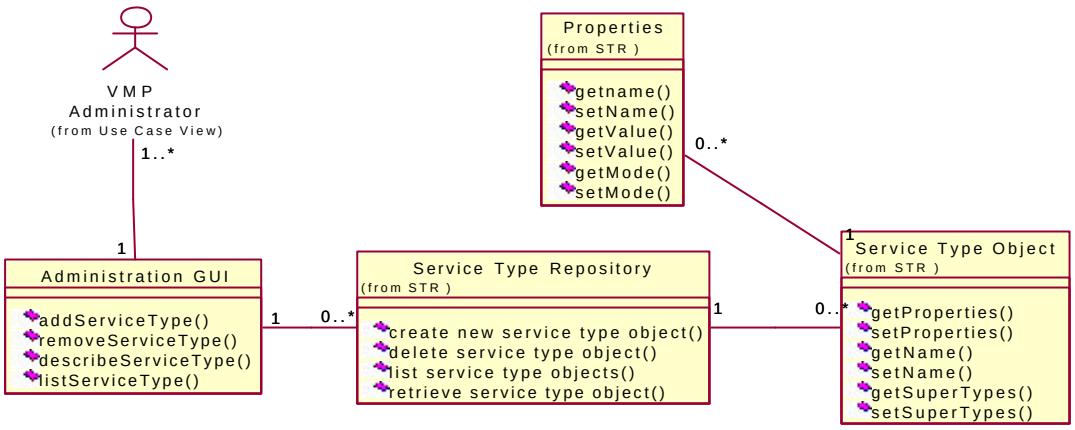

Figure 35: VMP Administration Class Diagram

According to this class diagram, the administrator uses directly the Service Type Repository (STR) to perform his operations. For example, if the administrator would like to remove an existing service type, it first specifies the unique name of the service type and then the STR locates the corresponding Service Type object. Since the operation requested is *delete service type*, the STR deletes the corresponding object. In a similar way, if the administrator would like to modify an existing service type, it first specifies the unique name of the service type and then is allowed to change the properties and inheritance of the service type. The deletion and modification of existing service types might create inconsistencies with the service offers registered into the virtual marketplace. In that case, if at least one service offer has been already registered for a given service type, the administration GUI does not allow the deletion or modification of this type.

6.6 Summary

This chapter presents detailed specification and design of the virtual marketplace agents and the deployment of the standard OMG-Trader. More specifically, three agents are proposed and analysed, namely Service Type Agent (STA), Service Offer Agent (SOA) and Service Retrieval Agent (SRA). For every agent, the internal architecture of it and the key components are specified. Then, for every operation that the agent supports, a UML sequence diagram is given and discussed. The sequence diagrams actually specify the way that the different internal entities of the agents are co-operating to provide the services to other agents with the environment. In addition to the three agents, the Virtual Marketplace ontology is specified. The ontology has been specified in ACL/XML and is based on the FIPA compliant request-response protocol.

In the following two chapters, the two key phases of the VE life-cycle are explained and discussed. These phases are the Business Process Specification and Registration and Business Process Management.

Chapter 7: Business Process Specification and Registration

7.1 Introduction

During the Business Process Specification Phase, VE candidate partners specify their local and remote business processes. The specification of the business processes is done using the Business Process Definition Language (BPDL). The reference model of Workflow Management Coalition defines as business process "a procedure where documents, information or tasks are passed between entities of the workflow according to defined sets of rules to achieve, or contribute to, an overall business goal" (WfMC, 1996). In general, a business process specification is a representation of a real-world activity in a machine readable format. Conceptually, a business process specification is a directed, acyclic graph in which nodes represent steps of execution and edges represent the flow of control and data among the different steps.

For every business process, the input parameters, the output parameters, the sub-processes, the tasks and the conditions among the sub-processes and tasks are being specified. The input and output parameters constitute the flow of data, i.e. the data that are passed among the elements of the process. The conditions constitute the flow of control, i.e. the order according to which the elements of the process will be executed. Additionally, every process or sub-process is specified as local or remote process. Local processes are the processes that can be fully provided by this domain while remote processes are the processes that can be provided only by remote domains. Moreover, for every task the associated business object, that will be deployed, is also specified. Tasks are elementary processing units while processes orchestrating and controlling the whole business process according to the flow of data and control.

During the Business Process Registration Phase, VE candidate partners register their local business processes to the virtual marketplaces. The registration process is performed using the

existing service types provided by the marketplace. If there is no associated service type for a particular process, a new one is being created, by possibly inheriting existing service types. This process can be done either, automatically or through the virtual marketplace administrator. During the registration process, certain values for certain attributes related to the service type, like location, quantity, etc., are provided. These attributes are usually related to the provision of the process to remote administrative domains. In addition to the service provision related attributes, a set of attributes, which will influence the negotiation process are also specified, e.g. price. These attributes might include the low price that can be negotiated upon, the maximum quantity that can be offered, the best and worst delivery dates, etc.

In Business Process Specification and Registration phase, the Business Process Analysts performs the specification of the business processes, the administration of the registration of local processes into the virtual marketplace, and the specification of the terms and conditions of the negotiation process. The following sections present an analytical description concerning the services that the business process analyst uses to fulfil its role.

7.2 Business Process Specification

The Business Process Specification Phase is a manual process performed by the Business Process Analyst of each individual administrative domain. The main responsibilities of this role are the creation of a business process, the modification and the deletion of it.

In order to perform these operations, the Business Process Analyst uses the Business Process Definition Language and the Business Process Repository (BPDL). The BPDL is an XML-based language that enables the specification of complex VE business processes. The language has been specified and designed for the purposes of dynamic VEs and enables the utilisation of remote business processes in a dynamic and flexible way. The business processes of each domain are stored into the Business Process Repository (BPR). The BPR is a persistent system that stores business processes and provides services for the interpretation of processes from XML format into a specialised model that can be easily deployed by the intelligent, autonomous agents.

The following sections present the concept and certain design issues related to the business process specification language and the business process repository.

7.2.1 Business Process Definition Language

The BPDL is an XML-based language that enables the specification of complex VE business processes. The BPDL provides all the necessary mechanisms to describe complex processes and relationships among them. Each step within a process is a special business goal that an autonomous, intelligent agent undertakes the responsibility to execute and manage when specialised conditions are satisfied.

The basic syntactical elements of the Business Process Definition Language are:

- **Business Process**, a description of the sequence of steps to be completed in order to accomplish a business goal. A business process has a name, a set of input and output parameters, and special start pre-conditions. A process consists of one or more sub-processes,

- **Sub-process**, or each step within a process. A sub-process has a name, a set of input and output parameters, and start pre-conditions. With these elements, every sub-process specifies the flow of data and the flow of control. The flow of data, specified through the input/output parameters between activities, is a series of mappings between output data and input data to allow activities to exchange information. Actually, the output parameters of one process can be input parameters of another process. The flow of control, specified by special logical conditions assigned to the sub-processes or atomic processes, is actually the order in which activities are being scheduled and executed. A sub-process consists of one atomic process or one or more sub-processes. A sub-process can be either local, when the current domain can execute the whole sub-process, or remote, when the execution of the sub-process can be performed by another domain. Sub-processes are used for nesting and modular design and reusability reasons.
- **Atomic process** has a name and a business object assigned to it that will be deployed when the atomic process is executed. The atomic process passes to the external business object the input data, waits for the execution of the business object, collects the output data, and finally sends them back to the workflow system. The atomic processes are computational elements, i.e. they provide special elementary operations into the business process. The atomic process has always a well-defined functionality and is always associated with an external business object.
- **Input Parameters**: a sequence of typed variables and structures that are used as input to the invoked activities. An input parameter has a name, a basic type like string, integer, etc. and constraints which associate parameters with certain values. During process specification time, the input parameters have only default values and optional constraints associated to them.
- **Output Parameters**: a sequence of typed variables and structures denoting the output of an invoked sub-process or atomic process. An output parameter, in a similar way like the input parameters, has a name, a basic type like string, integer, etc. and constraints which associate parameters with certain values. During process specification time, the output parameters have only default values and optional constraints associated to them.
- **Conditions**, which specify the circumstances under which certain events will happen. When a condition becomes true then the corresponding sub-process should start its execution. Conditions can be either atomic or composite.
- **Atomic Conditions** are simple logical expressions formed among an input or output parameter, a binary comparison operator and a value. The comparison operators might be >, <, >=, <=, =,etc.
- **Composite Conditions** are complex logical expressions formed in terms of atomic conditions and logical operators such as AND, OR, or NOT. The BPDL also enables the specification of conditions on an existing condition specification language, like the JESS language or OMG Constraint Language. These conditions are computed based on several standard third party conditions checkers included into the agents.
- **Business Object**: is any type of external object that can be accessed either locally or remote through Java RMI or CORBA-IIOP. It is the responsibility of the atomic process to deploy and integrate the business into the business process specification.

The following picture depicts the relationships among the different entities of the BPDL.

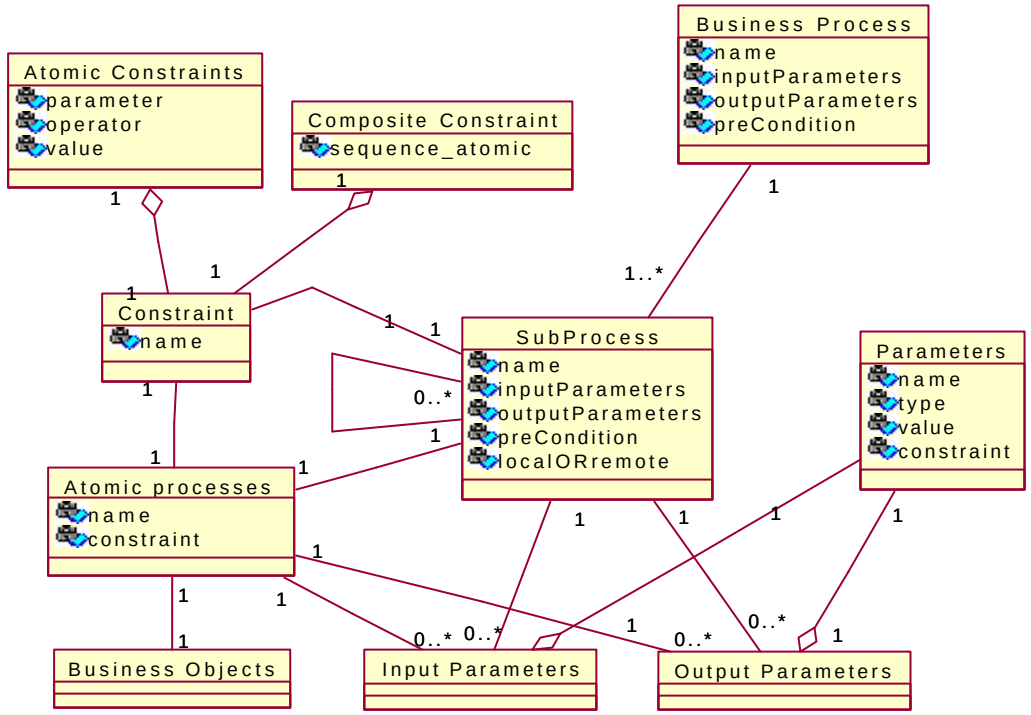

Figure 36: Business Process Definition Language Class Model

The specification of the business process definition language in XML-DTD format is the following. This specification is actually a direct translation of the model presented in Figure 36.

```
<!-- Entry point for business process definition in XML -->

<!ELEMENT business-process-definition (process-definition | condition-definition | parameter-definition)*>

<!-- Generic class for representing a business process definition -->

<!ELEMENT process-definition (process-name, comment?, in-data*, out-data*, (atomic-process | composite-process))>
<!ELEMENT process-name (#PCDATA)>

<!ELEMENT atomic-process (external-task-name)>
<!ELEMENT external-task-name (#PCDATA)>

<!ELEMENT composite-process (composite-process-element+, exception-handling-process-element*)>
<!ELEMENT composite-process-element (precondition-name?, sub-process-name, time-allowed-to-complete?, (is-remote | to-be-negotiated)?, in-data*, out-data*)>
<!ELEMENT exception-handling-process-element (exception-source+, sub-process-name, time-allowed-to-complete?, (is-remote | to-be-negotiated)?, in-data*, out-data*)>
<!ELEMENT precondition-name (#PCDATA)>
<!ELEMENT exception-source (#PCDATA)>
<!ELEMENT sub-process-name (#PCDATA)>
<!ELEMENT is-remote EMPTY>
```

```
<!ELEMENT to-be-negotiated EMPTY>
<!ELEMENT in-data (#PCDATA)>
<!ELEMENT out-data (#PCDATA)>
<!ELEMENT time-allowed-to-complete (#PCDATA)>
<!ATTLIST time-allowed-to-complete
              unit (seconds | s | minutes | m | hours | h | days | d) #REQUIRED
              trials CDATA "1">

<!-- Generic class for representing a condition definition -->

<!ELEMENT condition-definition (condition-name, comment?, (atomic-condition |
composite-condition))>

<!ELEMENT condition-name (#PCDATA)>

<!ELEMENT rule (rule-language, rule-body)>
<!ELEMENT rule-language (#PCDATA)>
<!ELEMENT rule-body (#PCDATA)>

<!ELEMENT composite-condition (is-not?, (is-or | is-and), branch-condition-
name+)>
<!ELEMENT is-not EMPTY>
<!ELEMENT is-or EMPTY>
<!ELEMENT is-and EMPTY>
<!ELEMENT branch-condition-name (#PCDATA)>

<!ELEMENT atomic-condition (((is-not?, process-status?) | rule), external-
condition-name)>
<!ELEMENT process-status (process-name)>
<!ATTLIST process-status process-state (running | notStarted | suspended |
aborted | terminated | completed) #REQUIRED>

<!-- Generic class for representing parameter definition -->

<!ELEMENT external-condition-name (#PCDATA)>

<!ELEMENT parameter-definition (parameter-name, comment?, parameter-type,
parameter-value)>
<!ELEMENT parameter-name (#PCDATA)>
<!ELEMENT parameter-type (#PCDATA)>
<!ELEMENT parameter-value (#PCDATA)>

<!ELEMENT comment (#PCDATA)>
```

The above provided business process definition DTD and an example business process definition are provided at the end of the thesis on the ANNEX.

The BPDL has been defined for the purposes of dynamic VEs. During specification time, each process can be specified either as local or remote. A process is considered local when the specification of this process exists fully in this domain. On the contrary, a process is considered remote, when the specification and execution of this process can be provided only by another domain. This is achieved by the setting of a special flag called *isRemote*. When a sub-process has been specified as remote, then the corresponding flag *isRemote* has been set to true. Otherwise, if the process is not considered remote, then it is local, and in that case the execution and management of the sub-process will be done within the same domain.

The BPDL has been specified in XML-DTD while the specification of each process is done in XML. This design choice has four very serious consequences. The main reason is that the

extension of the language can be easily performed due to the dynamic capabilities of the XML. Additionally, the specification of processes is done in XML, which is simple ASCII text, and thus, the business process descriptions are ASCII text files that can be easily stored in any conventional file system. Furthermore, XML is an open standard that most of the emerging systems and especially agent-based ones will in the near future support. And finally, a business process definition parser in XML can be easily developed due to the existing commercial support of XML parsers.

The Business Process Definition Language supports increased re-usability and modular design. This is achieved due to the *call by name* concept. This means that existing business process specifications can be easily used in other process specifications by only referring to the name of the process, sub-process, or atomic process. For example, if there is a sub-process called "find_book", that has been specified within one process A, then this sub-process is specified only one time and its name can be used to represent the specification of this sub-process in different other processes, though the specification of the sub-process has been done in another process. This means that process specifications can easily include references to other processes. In this way, high degree of reusability and modularity is achieved. This concept leads directly to reusability of process specification building blocks. In a similar way like the middleware services, processes can be built by combining existing business process specifications. This reduces significantly the business process specification costs and increases the degree of flexibility of the specified processes.

7.2.2 Business Process Repository

Business processes are stored into the business process repository (BPR). In every administrative domain only one BPR exists. In general, the BPR is a persistent storage system that maintains the current XML specifications of the business processes. Business processes are specified in XML ASCII files and thus, they can be stored into conventional file systems.

The main operations offered by the BPR are:

- insert a new business process specification,
- delete an existing business process specification,
- modify a business process specification,
- parse and interpret a business process specification in XML.

The BPR actually maintains all the associations between business process specifications and XML files. For every business process specification, an XML ASCII file is specified. The XML file contains the definition of the process following the syntax of BPDL. The association between processes and XML file specification is achieved by having a special configuration file called *business.dtd*. The syntax of this DTD is the following:

```
<!ELEMENT BUSINESS-CONTROL (BUSINESS*)>
<!ELEMENT BUSINESS  (NAME)>
<!ELEMENT NAME (#PCDATA)>
```

For all local business processes specified in this domain the corresponding XML file name is specified. An example of the *business.xml* file, that contains two processes, namely Process1 and Process 2, is given:

```
<?xml version="1.0"?>
<!DOCTYPE BUSINESS-CONTROL SYSTEM "Business.dtd">
<BUSINESS CONTROL>
 <BUSINESS>
   <NAME>Process1.xml</NAME>
 </BUSINESS>
 <BUSINESS>
  <NAME> Process2.xml </NAME>
 </BUSINESS>
</BUSINESS-CONTROL>
```

Initially, the BPR reads and interprets the *business.xml* file. The BPR locates all the files of the currently specified business process specifications, opens the specification files, reads the specifications in XML, produces the Definition Model for each process, and creates the List of Business Process Specification (LBPS). The LBPS is actually the list of all DMs, i.e. business processes that can be provided by this domain. Access to this list is done through the names of the business processes that uniquely identify a business process specification.

The Definition Model (DM) is an object-oriented representation of the business process specification. The DM contains actually the relationships of different processes, sub-processes, atomic processes and conditions and consists of the following modules:

- **process definition**: provides the operations for the retrieval and setting of the appropriate elements of a process, like set and get input/output parameter, set and get external task, or set and get sub-processes. One of the key methods is the *createInstance* that enables the creation of an instance of this process. The Workflow Provider Agent (WPA) uses this method to create a new instance of the process (see Workflow Provider Agent (WPA) section in next chapter).
- **Sub-process definition**: provides operations for the retrieval and setting of the appropriate elements of a sub-process, like set and get input/output parameter, set and get atomic process, set and get sub-processes. This is actually what the Workflow Provider Agent uses to locate the sub-processes of a process and execute them recursively.
- **process condition definition**: provides operations for the retrieval and setting of the appropriate elements related to the conditions of a process or sub-process like the logical expressions of the conditions.

The following Figure 37 depicts the class model of the DM, the key operations that the different modules provide and the relationships that they have.

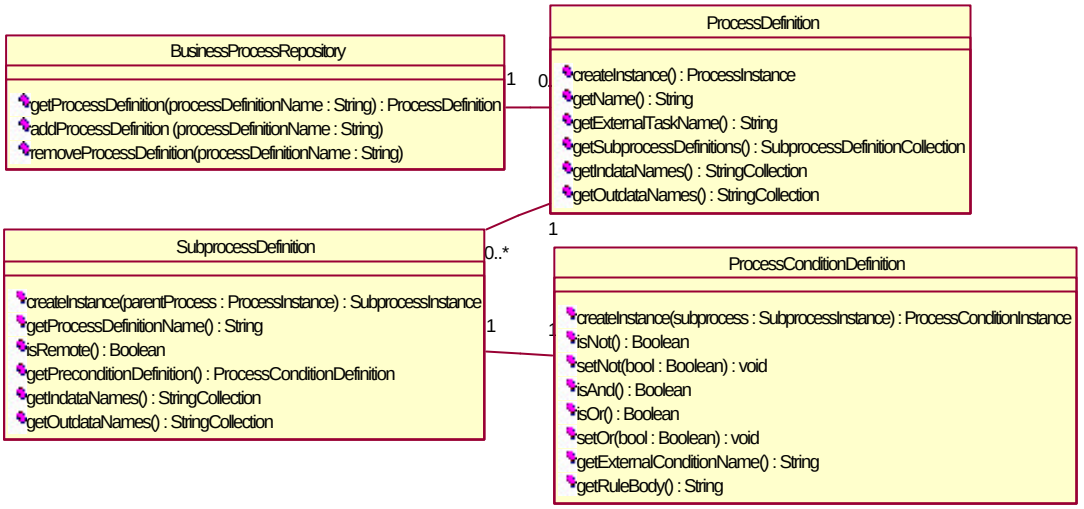

Figure 37: Process Definition Class Model

When a particular business process with a particular name is requested for execution, the BPR searches the LBPS, locates the Definition Model for this process and returns it back to the agent that requested the process. The agent, based on the DM, can create an instance of the process, can find all the sub-processes and atomic processes, and can get all the conditions associated with a particular sub-process. More details regarding how the DM is deployed during the instantiation, execution and management of a process are provided in the Workflow Provider Agent section.

The main operations of the BPR are provided to the Business Process Analyst (BPA) in a manual way. When the BPA wants to specify a new business process, it uses the BPDL and writes manually in a XML file the specification of the process including all the necessary elements like sub-processes, input and output parameters, and conditions. During the specification phase, the BPA can reuse existing specifications of processes and sub-processes that have been previously specified in other specifications. It is assumed that the BPA has a thorough understanding of the syntax and the grammar of the BPDL. Then, the BPA updates the configuration file of the existing business processes, i.e. the *business.xml*, by declaring the newly specified process. In a similar way, when the BPA wants to modify an existing business process, it opens the corresponding XML file of the business process specification and makes the appropriate modifications. Changes to the configuration file, namely the *business.xml*, are not required since the process has been already registered. The BPR will automatically get the modified specification of the process and put it into the LBPS. Finally, when the BPA wants to delete a business process specification, he only needs to delete the name of the process from the corresponding configuration file. Optionally, the BPA can also delete the XML specification file from the file system. If the deleted process specification is referenced by name by other processes, then inconsistencies will occur. In that case, the BPA is responsible for modifying the process specifications accordingly to avoid the problem.

In the following sections, the relationship of BPR with the execution of the business processes and the workflow engine are provided.

7.3 Business Process Registration

The Business Process Registration Phase is a process performed by the Business Process Analyst of each individual domain with the help of an autonomous intelligent agent called Provider Negotiation Agent (PNA). The main responsibilities of the Business Process Analyst (BPA) is this phase are specification, modification, and deletion of certain terms and conditions related to the registration and negotiation of local and remote business process to potential partners. These terms and conditions will be used from this domain during the negotiation process with potential VE partners.

The business process registration phase includes the following steps for every local process:

- The Business Process Analyst specifies in the Offer Repository the local and remote processes, including the input and output parameters that can be provided to potential VE partners. For that reason, reuse of the existing business process specifications stored in the Business Process Repository can be done. Additionally, the Business Process Analyst specifies the constraints related to the negotiation parameters. These constraints specify the lower and upper bounds of the accepted values and will, in general, drive and determine the negotiation process,
- The Provider Negotiation Agent (PNA) agent retrieves from the offer repository all the local processes and the corresponding input, output, and negotiation parameters and migrates to the virtual marketplace where it checks for every local process the existence of a corresponding service type. For every local process a generic service type should exist. This service type should have as properties the input and output parameters of the local process and some extra properties related to the negotiation process, e.g. price, delivery day, payment method, payment due, etc (see next section). If there is no existing service type available on the marketplace for a given local business, the PNA agent creates a new one. In the sequel, the PNA registers every local process in the marketplace in relation to an existing compatible service type. For that reason, the PNA agent interacts with the Service Offer Agent (SOA) in the virtual marketplace. After the successful registration of local business processes, potential VE partners can search the virtual marketplace and locate this domain as a potential VE candidate partner. This is the initial step for the beginning of the negotiation process among the domains that will be described in the next sections (see NRA section).

The Offer Repository (OR) stores information regarding all the local and remote processes that can be provided to potential VE partners. For every local and remote process, the name of the process, the input, output, and negotiation parameters are stored. All the different types of parameters have a name associated to it, a type, e.g. a string, a value, and a constraint. In addition to that, the negotiation parameters have two classification modes, namely public and private. The values of the negotiation parameters that have public mode can be revealed into the different agents of other domain and the virtual marketplace. On the contrary, values of negotiation parameters that have private mode should not be revealed into the public agents and virtual marketplace, and are the ones that are used for determining the negotiation process. For example, *price* can be a negotiation parameter with private mode. This means that the price of the local process can be revealed when a negotiation process will start. On the contrary, *discount* might be another negotiation parameter with private mode. This means that the percentage of discount offered by this domain concerning this process will be revealed only when the other domain will request it. The constraints are simple binary logical expressions related to

parameters and values. For example, if one potential negotiation parameter is *price*, then the following constraint can be assigned (*"price"<="32"*). This means that the price that can be in worst case agreed should not be maximum than 32. In such a way, the Business Process Analyst can specify which negotiation parameters and values will be used as private and public and which lower and higher values can have during the negotiation process.

The main operations that the Business Process Analyst can do with the Offer Repository are:

- registration of a new local or remote process, input, output, and negotiation parameters and values
- modification of a local or remote process by providing new values to input, output, or negotiation parameters.
- deletion of a local or remote process and the associated values for the input, output, and negotiation parameters
- retrieval of a local or remote process and the associated values for input, output, and negotiation parameters

The use case diagram of all the operations that the Business Process Analyst can perform is depicted in the following Figure 38. In this diagram, except the key operations that can be performed, a set of sub-operation is also identified. For example, the *register new local process* operation is supported by the *set process name*, *set input* and *output parameters* and *set negotiation parameters*.

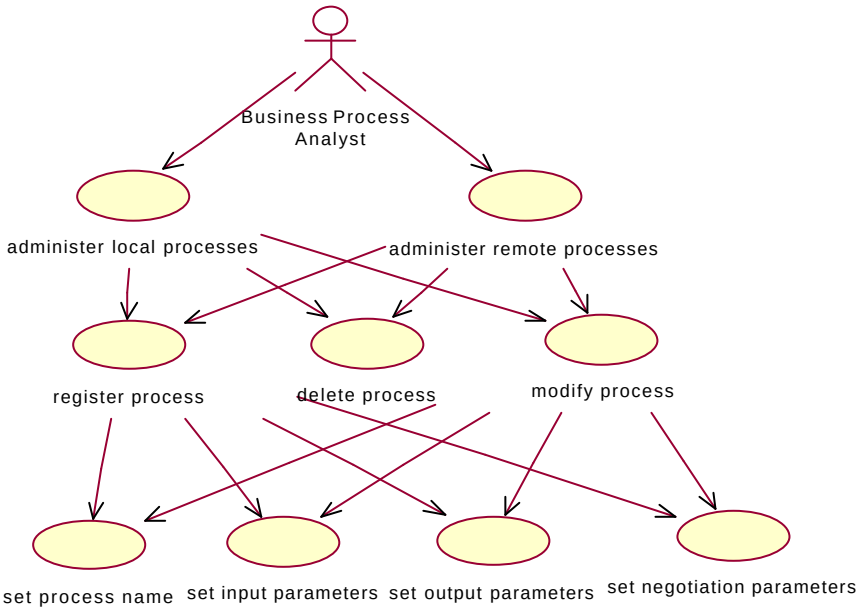

Figure 38: Use Case Diagram of Local Business Process Specification in Offer Repository

The above mentioned operations are provided though a Graphical User Interface that helps the user to fulfill his role in a friendly way. The main entities of the Offer Repository administration GUI are the following:

- **Offer Repository GUI Manager**: responsible for the provision of the graphical interface to

the human operator,

- **Offer Repository (OR)**: responsible for the provision of the core operations of the local and remote process management and the persistent storage of them,
- **Offer**: represents an offer stored into the repository and provides a set of operations for the access and modification of them. Operations provided by the entity are to get and to set the input, output parameters, and negotiation parameters.
- **Parameters**: represents an abstract class that describes the main operations for accessing and modifying parameters like get and set parameter name, value, and constraint.

The class diagram of the Offer Repository administration GUI and the relationships among them entities are depicted in the following Figure 39.

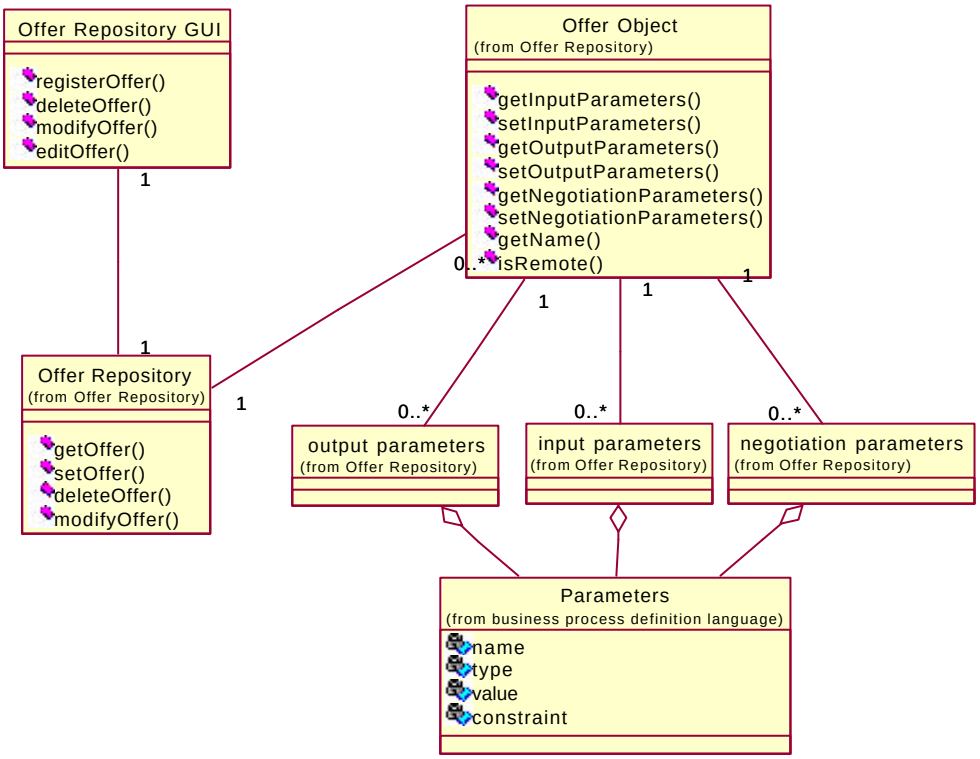

Figure 39: Offer Repository Administration GUI Class Model

All the offers that have been specified in the Offer Repository are stored in an XML file in ASCII format following a certain structure determined by the following XML-DTD specification. This is actually the specification of the above stated structure of offers in XML style. However, this choice makes the retrieval of offers from the Offer Repository easy and flexible. The Offer Repository supports special XML interpretation mechanism that enables the transformation from the XML into the previously described class model and the opposite. In such a way, the Provider Negotiation Agent can easily access the information stored into the repository and perform the registration of offers into the virtual marketplace. The Offer Repository information is stored on a normal file system due to the fact that the XML file that contains the values is actually an ASCII file.

The XML-DTD specification of the Offer Repository is the following.

```
<!ELEMENT Offers (local_processes*)>
<!ELEMENT local_processes (process)>
<!ELEMENT remote_processes (process)>
<!ELEMENT process (processName, inputParameter, outputParameter,
negotiationParameter)>
<!ELEMENT inputParameter (parameter*)>
<!ELEMENT outputParameter (parameter*)>
<!ELEMENT negotiationParameter (parameter*)>
<!ELEMENT processName (#PCDATA)>
<!ELEMENT parameter (parameter-name, parameter-type, parameter-value)>
<!ELEMENT parameter-name (#PCDATA)>
<!ELEMENT parameter-type (#PCDATA)>
<!ELEMENT parameter-value (#PCDATA)>
<!ATTLIST parameter-value constraint (lessequal | greaterequal | equal)
#IMPLIED>
<!ELEMENT listOfDeleted (processName*)>
<!ELEMENT listOfModified (processName*)>
```

In the following section certain analysis and design issues regarding the internal structure of the PNA agent and how it functions in order to perform the registration of business processes into the virtual marketplaces are discussed.

7.3.1 Provider Negotiation Agent

The Provider Negotiation Agent (PNA) is responsible mainly for two key activities, namely for registering the local processes of an administrative domain into the virtual marketplaces, based on the values stored into the Offer Repository, and for negotiating with VE partners about certain local processes.

In the first case, the PNA retrieves all the local processes that have been specified in the Offer Repository, migrates to the virtual marketplace and starts communicating with the STA and SOA agents by exchanging FIPA compliant ACL/XML messages based on the Virtual marketplace ontology. The communication protocol for the exchange of messages is the standard FIPA request-response protocol. If a service type for a given local business process already exist, then the PNA registers an offer to the marketplace by sending a register service offer request message to the SOA agent. If a service type does not exist, then the PNA agent initially creates a new service type, by communicating with the STA agent, and then registers the service offer.

In the second case, if a VE partner needs to execute a remote process, it first conducts the virtual marketplace to find a set of VE candidate partners for this process and then starts to negotiate with them based on a standard negotiation protocol. The agent representing the requestor domain is the Requestor Negotiation Agent (RNA) (see RNA section) while the agent representing the VE candidate domain is the PNA. These two agents communicate by exchanging messages specified in FIPA compliant ACL/XML format using the negotiation ontology. The protocol used during the negotiation process is the standard FIPA Contract-Net. More details about the negotiation process and how the PNA communicates with the RNA are provided in the section related to the Requestor Negotiation Agent (see the related section to the RNA).

In both cases, the Provider Negotiation Agent should send, receive and parse incoming and outcoming ACL/XML messages by using both an ACL and an XML parser. The PNA should also support two different FIPA compliant communication protocols, namely the Request-

Response and the Contract-Net protocol. Finally, the PNA should also formulate outgoing ACL/XML messages accordingly and take decisions about the proposals that he should do during negotiation based on a well-defined but simple strategy. All the above stated operations are provided by specialized entities included into the PNA agent. In addition to the generic entities of a FIPA compliant agent, the internal architecture of the PNA agent contains the following modules:

- **VMP and INDO XML Parser**: responsible for parsing the content of the FIPA ACL messages based on the marketplace and the Inter-domain ontology,
- **VMP and INDO Message composer**: responsible for composing the appropriate response FIPA ACL-XML messages related either with the marketplace agents or the RNA The structure of the messages is based on both the virtual marketplace and the Inter-domain ontology,
- **Decision manager**: responsible for controlling the basic operations of the agent, communicating with the Offer Repository, the Strategy manager, and the other entities
- **Strategy manager**: responsible for providing proposals during the negotiation process and according to a well-defined strategy. In the context of this thesis[12], only a simple strategy algorithm has been implemented.

In the following picture the internal architecture of the PNA agent and the relationships among the basic modules is depicted.

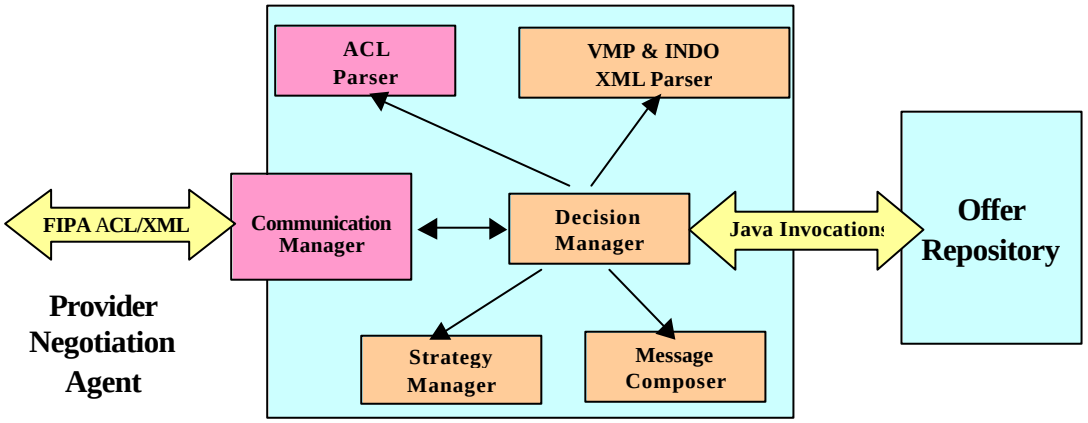

Figure 40: Provider Negotiation Agent Internal Architecture

The PNA initially retrieves from the Offer Repository all the local process that have been specified. For that reason, the agent accesses the XML file of the Offer Repository and instantiates the Offer Repository object. In the sequel, for every local process specified in the Offer Repository, the agent retrieves the name of the process, the input, output and negotiation parameters and the values associated with them. As soon the retrieval of the data from the Offer Repository finished, the agent migrates to the virtual marketplace using the native migration services of the agent platform. Then, for the local processes specified in the Offer Repository

[12] The strategy that can be followed during negotiation process is out of the scope of this thesis. However, the interesting reader can have a look on the following reference (Bichler 98)

performs the following operations. The PNA generates a service type registration message and sends it to the STA agent. The service type has as a name the name of the local process, and properties the names and values of the input, output and negotiation parameters. If there is no service type with such characteristics, then a new service type is created. As soon as a service type has been found or created, the PNA agent generates a service offer registration request message and sends it to the SOA agent. The message is a FIPA compliant ACL/XML message that follows the virtual marketplace ontology. Property values for the negotiation parameters are only the allowable values specified in the negotiation parameters, i.e. the negotiation parameters that have public mode. On the contrary, private property values that will influence the negotiation process are kept secret and are not included into the offer registration. When all the local processes have been successful registered into the marketplace, the PNA migrates back to the original domain using the migration services of the mobile agent platform. In the following Figure 41 the steps involved in the registration of the local business processes into the virtual marketplaces are depicted.

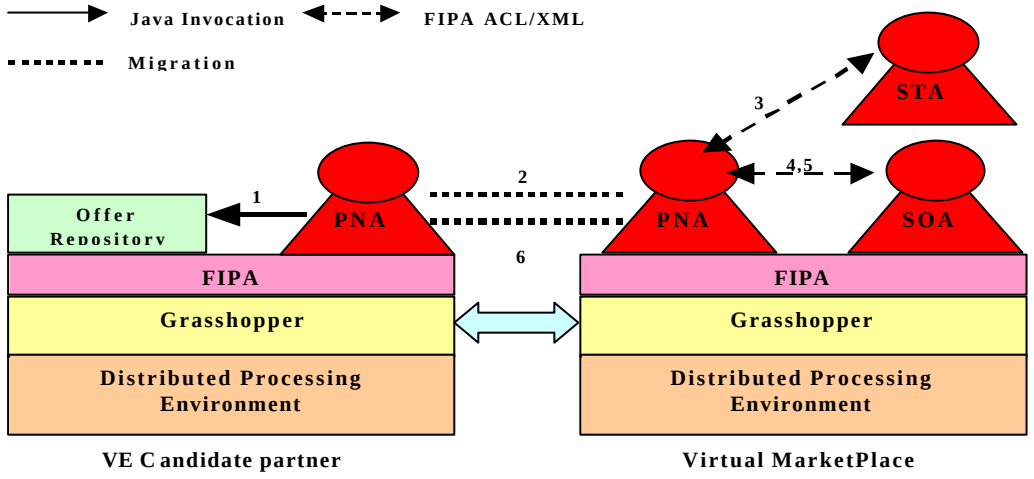

Figure 41: Business Process Registration into the Virtual Marketplace

In the following sequence diagram, the internal steps that the PNA agent follows to create a service type based on the data stored into the Offer Repository in relation to the internal modules is depicted. When the PNA agent launched, it first parses the Offer Repository in XML and instantiates an Offer Repository object. In the sequel, the Decision Manager gets all the stored offers. For every Offer object, the Decision Manager retrieves the name of the local process, the input parameters, the output and the negotiation parameters. For each one of these objects, the agent retrieves the name of the parameter, value and the constraints specified for it. Based on these information, the agent with the help of the Message Composer creates a *create service type* request message, that complies with the virtual marketplace ontology, and sends it to the STA agent located in the virtual marketplace.

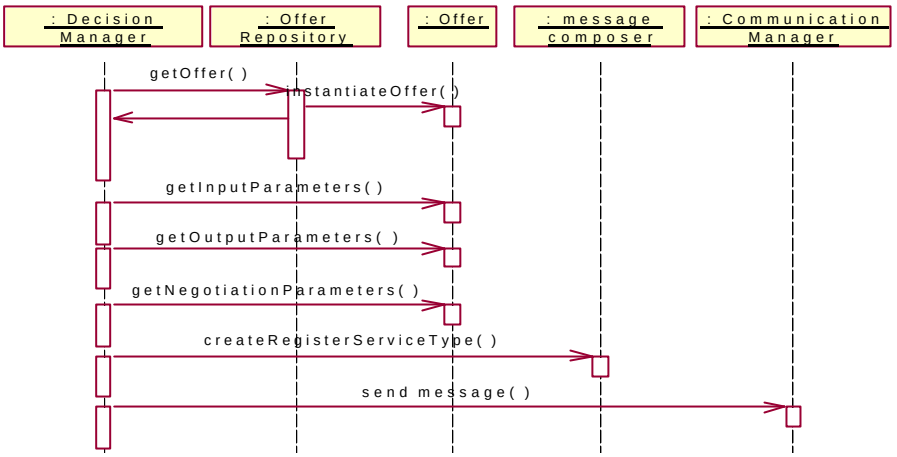

Figure 42: PNA Service Type Generation Sequence Diagram

Furthermore, in the following sequence diagram, the internal steps that the PNA agent follows to create and register a service offer based on an existing service type in relation to internal modules is depicted. In similar way like in previous case, the Decision Manager gets all the stored offers and for every Offer object, the name of the local process, the input parameters, the output and the negotiation parameters are retrieved. Based on this information, the agent creates a *register service offer* request message, which complies with the virtual marketplace ontology, and sends it to the SOA agent located in the virtual marketplace.

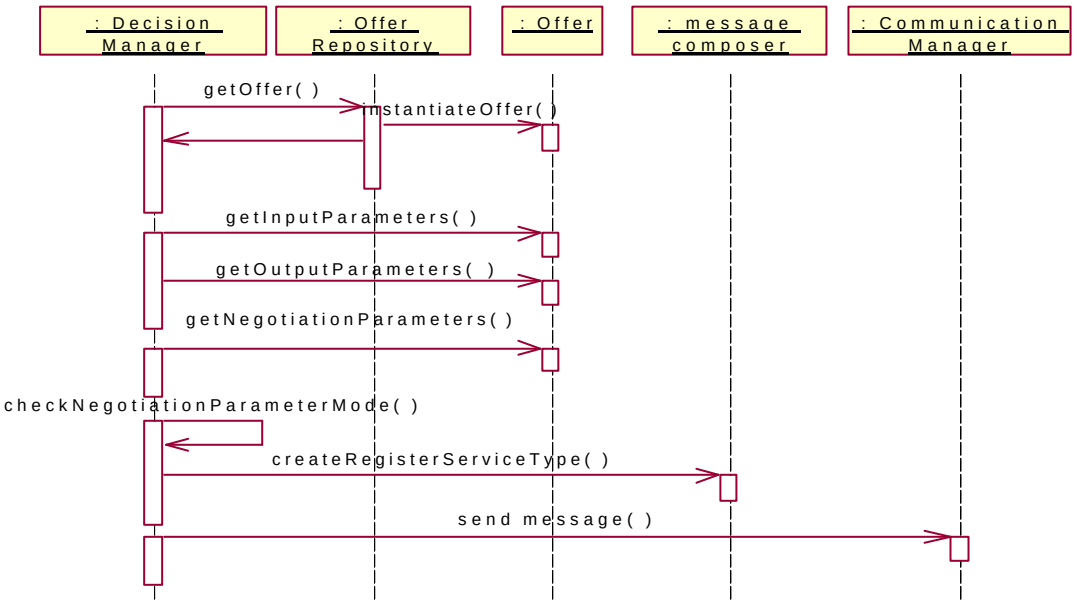

Figure 43: PNA Service Offer Generation Sequence Diagram

Moreover, when the Business Process Analysts deletes or modifies an offer from the Offer Repository, the corresponding offer in the virtual marketplace should be deleted or modified. In both cases, the PNA agent is re-launched again. In principle the agent follows the same process as previously described. In more details, the agent retrieves from the Offer Repository the set of

offers that have been deleted and modified. The names of these offers are stored in individual lists maintained by the Offer Repository. In the case of the deleted ones, the PNA migrates to the virtual marketplace, creates a *withdraw service offer* request messages and sends them to the SOA agent. In the case of modified, the PNA retrieves from the Offer Repository the modified offers and creates corresponding *modify service offer* request messages and sends them to the SOA agent in the virtual marketplace. As soon this service offer update process has finished, the PNA migrates back to its original domain and deletes the list of deleted and modified offers maintained in its local Offer Repository. When new modifications to existing offers will be done, the PNA agent will perform in a similar way, i.e. by updating the services offers in virtual marketplace and thus, maintaining consistency with the offers stored into the local, domain specific Offer Repository.

Finally, the PNA agent is also responsible for negotiation process during partner selection. For that reason, the PNA negotiates with the Resource Negotiation Agents (RNAs) located in the VE partner domains by following a standard negotiation ontology and protocol. Details regarding how the PNA agent is functioning, the type of messages exchanged, and the protocol and ontology used will be provided in the section related to Resource Negotiation Agent.

7.4 Summary

This chapter presents the detailed specification and design of the business process specification phase. More specifically, the XML-based business process definition language for shared business processes and the key constructs of it are fully analysed. In addition to that, the business process repository that stores business processes is presented and specified. Finally, the business process registration process is presented and analysed. This is actually the process that the different providers are using to register offers in the virtual marketplace. In that case, the Negotiation Provider Agent (NPA) is specified and analysed. The internal architecture of the agent, the services that it provides, as well as, a set of sequence diagrams, that explain the involvement of the internal modules, are provided.

Chapter 8: Business Process Management

8.1 Introduction

Business process management is related with the execution and management of shared business processes across different administrative domains. The management of business processes is performed in a autonomous and distributed way and is fully performed by autonomous intelligent mobile agents without human intervention.

The execution of a shared business process starts by the end-user of the VE. The main operations that this role performs are:

- log into the web site of the VE representative that provides the shared business processes by using a standard web browser,
- execution of a shared business process and monitor of the status of the running process,
- management of a shared business process, i.e. suspension, resumption, or termination of a running business process.

In addition to the above user initiated operations, the following situations occur. When a running business process is completed, the results of the process, i.e. the output parameters of it, are returned to the user automatically. Furthermore, If during the execution of a process a problem occurs, then the process is aborted and the user is notified about this fact.

The execution and management of shared business processes, in the context of dynamic VEs, are performed from the following autonomous, intelligent, FIPA compliant agents.

- **Personal User Agent (PUA)** responsible for managing the requests of the end-users coming from the standard web browsers. This agent is located on the VE representative domain. Every user request is checked for authorization and then is forwarded to the Domain Representative agent (DR).

- **Domain Representative (DR)** responsible for managing the requests of the PUA, if the domain plays the role of the VE representative and the requests of the remote domains, if the domain plays the role of the VE partner. In both cases, the DR authenticates the requests by conducting the Contract Repository. If the request is an authorized one and is related to the instantiation of a new process, the DR creates a Workflow Provider Agent (WPA) that will serve the request, otherwise the corresponding existing WPA is located and the request is forwarded to him,
- **Workflow Provider Agent (WPA)** responsible for executing and managing an instance of a process or sub-process. The WPA replies to requests coming from the DR or informs the DR about the status of the process that it executes. Additionally, the WPA co-operates in an autonomous way with other WPAs during the execution of the business processes. Finally, the WPA controls the execution of atomic processes involved into the business process by invoking, requesting, or informing different Resource Provider Agents (RPA),
- **Resource Provider Agent (RPA)** responsible for carrying out one specific atomic process of the business process. One atomic process is a simple elementary processing unit that can be included into one or more business processes. An RPA agent always deploys existing resources or business objects provided by the domain in a distributed and interoperable way.
- **Requestor Negotiation Agent (RNA)** responsible for managing the partner search, negotiation, and selection process. When a WPA realizes that a remote process is required for the continuation of the currently executed business process, it creates automatically a RNA agent. This agent migrates to the virtual marketplace, selects the potential VE candidate partners, based on some constraints, and starts a negotiation process with them. The result of the negotiation is an electronic contract that regulates this agreement.
- **Provider Negotiation Agent (PNA)** represents a VE candidate domain during the negotiation process and is responsible for the automated negotiations with other RNAs. Additionally, PNAs manage the business process registration to the virtual marketplace and update the contract repository when a negotiation process has been successfully ended, i.e. a contract has been agreed upon.

Based on the above definitions, the reference architecture of the VE representative or the VE partner domain is depicted in the Figure 44.

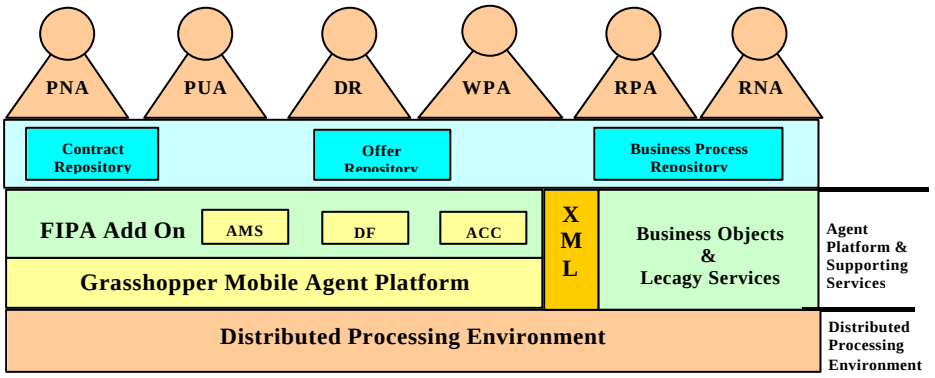

Figure 44: Business Process Specification, Registration, and Management Reference Architecture

In general, a business process consists of sub-processes. Each sub-process consists of one or more sub-processes or one atomic process. For a given process, the father of this sub-process is called the *super-process* while a child of the process is called the sub-process. Every sub-process has one and only one super process and one ore more sub-processes. On the contrary, an atomic process has one super process and no one sub-processes. This hierarchical organization of processes and sub-processes leads to the formation of a directed graph.

Based on the operations provided to the user, a business process can be in different states. The different states of a business process are:

- ready, after the instantiation of the process and before the execution of it,
- running, during the execution of the process,
- suspended, when the process has been suspended by the user or other super process,
- resumed or running, when the process were previously suspended and has now been resumed,
- completed when the execution has been completed,
- terminated when the initiator of the process has requested to terminate the process,
- aborted when the process, sub-process, or an atomic process has declared that it can not perform its operation and thus it aborts itself.

The following state transition diagram reveals the different states and how the process can change states. The nodes in the diagram denote the states while the arrows represent transitions from one state to the other. The messages on the arrows represent the events that might trigger the transition of the process from one state to the other. The dotted arrows reveal that the process it self, without any external intervention, can transit from one state to other. This means that when a process has been completed then it moves automatically, without any external intervention, to the terminated state. In a similar way, when a process has been aborted, it automatically moves into the aborted state. In all other cases, the process changes status after an external event generated by the end-user or the super process.

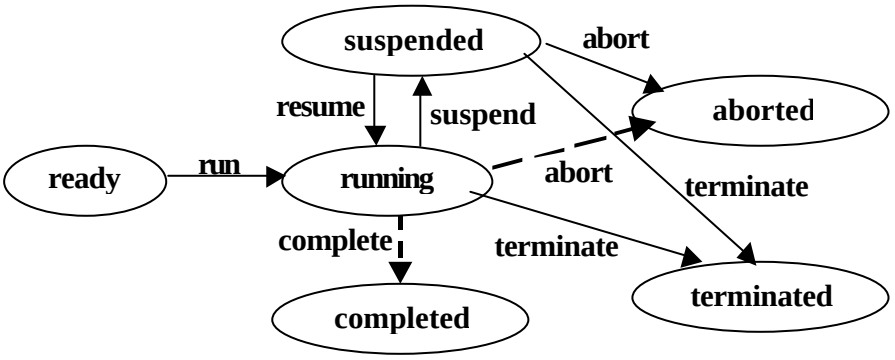

Figure 45: Process State Transition Diagram

When a process or sub-process changes state, then all the sub-processes should be informed accordingly. Therefore, when a business process is suspended, then all the running sub-processes and atomic processes should be suspended as well. In a similar way, when a process is resumed,

all the sub-process that have been suspended previously should also be resumed. Finally, if a process is aborted, then the execution of this process cannot be continued. In this case, the whole execution of the process should be aborted, i.e. all the sub-processes and atomic processes should also be aborted.

For example, when a request for suspension is arriving to a process, the process requests from its sub-processes to suspend. The sub-processes request from their sub-processes to suspend and so on. When the final level of sub-processes within the same process is reached, then these sub-processes are suspended and they inform their super-processes about their suspended state. When all the sub-processes of a process reported that they suspended, then the process can also suspend and inform its super-process. The similar phenomenon occurs and in the case of resumption, completion and termination. The lowest level sub-processes are completed or terminated first and then the highest levels complete or terminate one by one until the initial starting process is reached. Finally, the process informs the end-user about its current state or results.

However, in the case of abortion, the algorithm is working in a different way. If one sub-process cannot continue its operation, it informs its super-process and then aborts. When the super process receives an abort message from one of its sub-processes, it requests from the other remaining sub-processes to abort. When all the sub-processes have been aborted, then the process informs its super process about this fact and finally aborts. If a super process receives an abort message, it functions in a similar way. The whole procedure is continuing until the initial process aborts and reports its state to the end-user.

In general, the following conditions are hold for the status of a process:

- a process is running when all of its sub-processes and atomic processes are running,
- a process is suspended when all of its sub-processes and atomic processes are suspended,
- a process is resumed or running again when all of its sub-processes and atomic processes are suspended,
- a process is terminated when all of its sub-processes and atomic processes are terminated,
- a process is aborted when at least one of the sub-processes or atomic processes is aborted,
- a process is completed when all of its sub-processes and atomic processes are completed.

This back tracking alterations on process and sub-processes status are depicted in the following Figure 46. The numbers on the arrows depict the order that the events occur while the direction the requestor and the receiver.

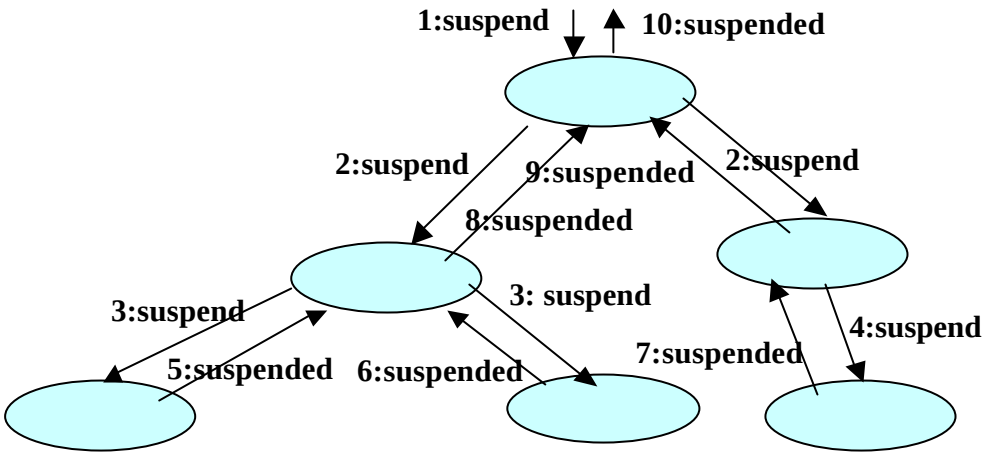

Figure 46: Process Status Mechanism

For every running business process a process instance is instantiated. This means that different instances of the same process can be executed and managed. Every instance of a process has a unique process id that differentiates it from the others. Although all of these instances are instantiated based on the definition of the business process, they are different due to the different input values that the users have provided for them. Therefore, a process instance is, in a similar way to object oriented systems, an object while a business process definition is the class specification. Process instances are executed and managed by autonomous, intelligent, FIPA compliant agents. In particular, a process or sub-process is managed and executed by a Workflow Provider Agent (WPA), while an atomic process is managed and executed by a Resource Provider Agent (RPA). This means that the execution and management of shared process instances is provided by a set of WPAs and RPAs that co-operate autonomously to accomplish the completion of the process.

The communication between WPAs and RPAs is performed by the exchange of FIPA compliant ACL/XML messages while the communication protocol used is the FIPA Request-Response protocol. The content of the messages is specified based on the inter and intra domain ontology. The intra and inter domain ontology specifies all the messages that the autonomous agents can exchange during the execution and management of a business process. If the agents belong in the same administrative domain, then the intra-domain ontology is used. If the agents belong to different domains, then the inter-domain ontology is used.

In the following sections, the different agents participate in the execution and management of VE processes are presented and certain details regarding the internal architecture, the modules and the operations that they provide are explained.

8.2 Personal User Agent

The Personal User Agent (PUA) is responsible for managing the requests of the end-users. This agent is located on the VE representative domain. In principle, the Personal User Agent "represents" the end user in the multi-agent system. Actually, the PUA is technically a gateway among the conventional web server of the VE representative and the multi-agent system. This means that all the requests of the user are managed from the web server and the PUA. Every

request from the user is translated into a legitimate ACL/XML message following the intra-domain ontology and sent to the Domain Representative (DR) agent for further fulfilment.

The main operations of the PUA agent are the following: the user, using a normal Web browser, logs into the VE representative web site and selects one VE business process for execution. The VE representative web site initially checks if the user is authorized[13] to use this process and then requests from him to provide values for the input parameters of the process. After the user has provided values for the input parameters, the execution of the process can start. This request for process execution is forwarded to the Personal User Agent (PUA), which creates a legitimate ACL/XML message with the name of the process, the instance id, and the input parameters and values, and sends it to the Domain Representative (DR) agent for further fulfilment. The DR agent gets the request and starts the execution of the corresponding process. When the execution of the process has been started, the user can always suspend the process, resume the process if it was previously suspended, terminate the process or ask for the status of the process. All the above mentioned requests are managed initially by the VE representative web server and are then forwarded to the PUA.

The following figure depicts these interactions. The requests from the user are forwarded from the web site to the PUA through a normal TCP/IP connection, i.e. the web server and Java servlets open a TCP/IP socket connection with the PUA and forwards to him the request.

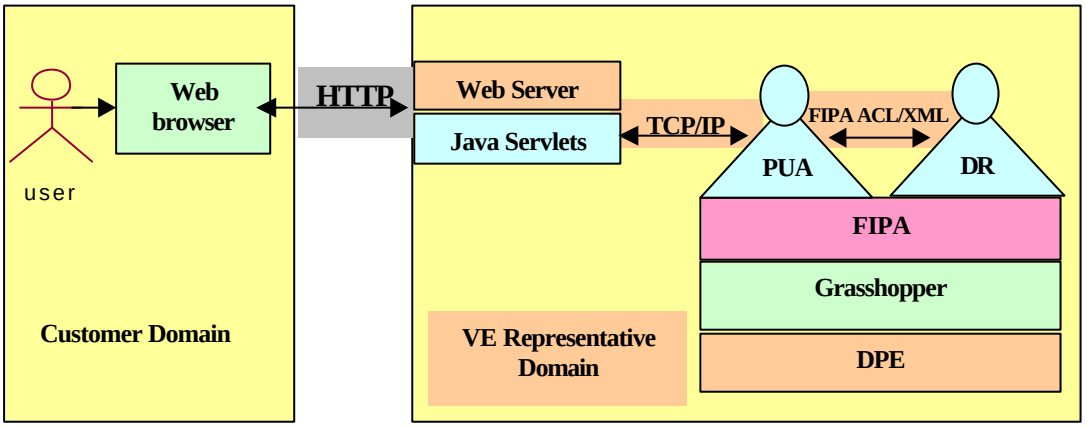

Figure 47: Personal User Agent Interaction Model

The PUA maintains for each user of the VE representative the list of processes that have been initiated by him. This information is stored into the Active User Repository (AUR). The AUR stores for every user the process instances, the status of each process instance, and input and output values. Whenever a new process instance is instantiated, a new entry is created into the AUR in relation to a user. When a process instance is completed, the corresponding process instance entry is deleted from the AUR. If the user requests to suspend, resume, or terminate a process instance, the current status of the process instance is also stored into the AUR.

The Active User Repository consists of the following three key entities, namely:

[13] Subscription of the user to the VE services is performed off-line in a manually way. The off-line subscription is considered out of the scope of the thesis. However, the interesting reader can get more information about on-line and off-line subscription on http://www.fokus.gmd.de/research/cc/platin/projects/

- **Active User Repository** (AUR) which provides the main operations to the PUA, like management of active users and process instances
- **Active User** (AU), which provides operations for the management of process instances of a particular user like create process instance, retrieve all instances, etc.
- **Process Instance Status** (PIS), which provides operations for the management of a particular process instance, like update process status instance, get and set instance name, etc.

The class model of the Active User Repository and the relationships among the main entities is depicted in the following Figure 48.

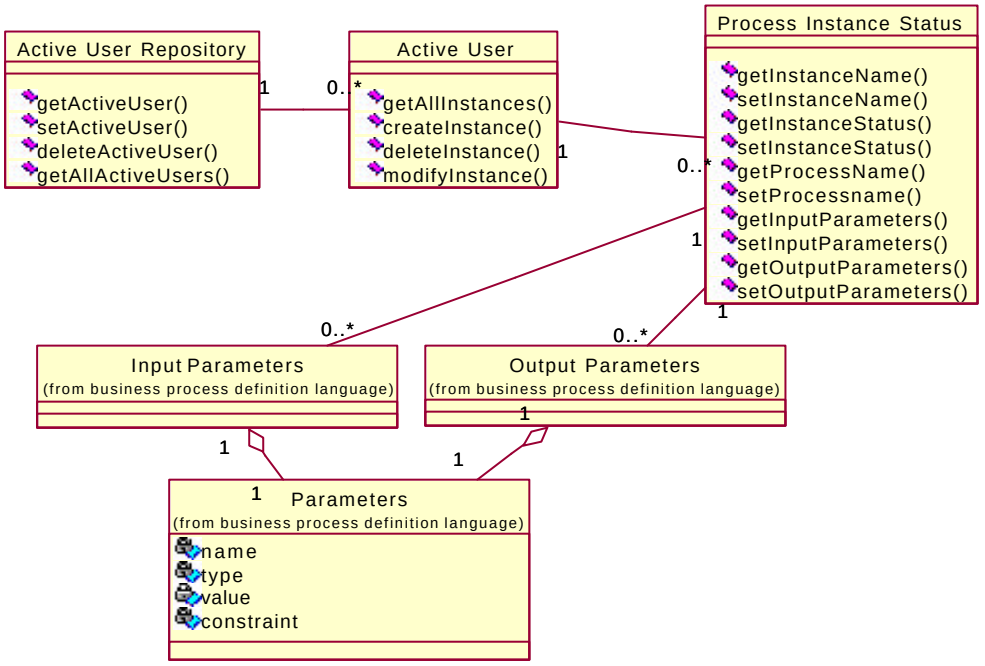

Figure 48: Active User Repository Class Model

In addition to the key entities of a FIPA compliant agent, the PUA agent contains the following key modules:

- **INDO XML Parser**: responsible for parsing the content of the FIPA ACL messages based on the intra-inter domain ontology,
- **INDO Message composer**: responsible for composing the appropriate response FIPA ACL-XML messages related to the DR agent. The structure of the messages is based on the inter-domain ontology,
- **Decision manager**: responsible for controlling the basic operations of the agent, communicating with the Active User Repository and the other entities
- **TCP/IP server**: responsible for getting the requests of the users from the web site and for initiating the necessary actions. It also forwards back the web site the results of the requests.

In the following picture the internal architecture of the PUA agent and the relationships among the basic modules is depicted.

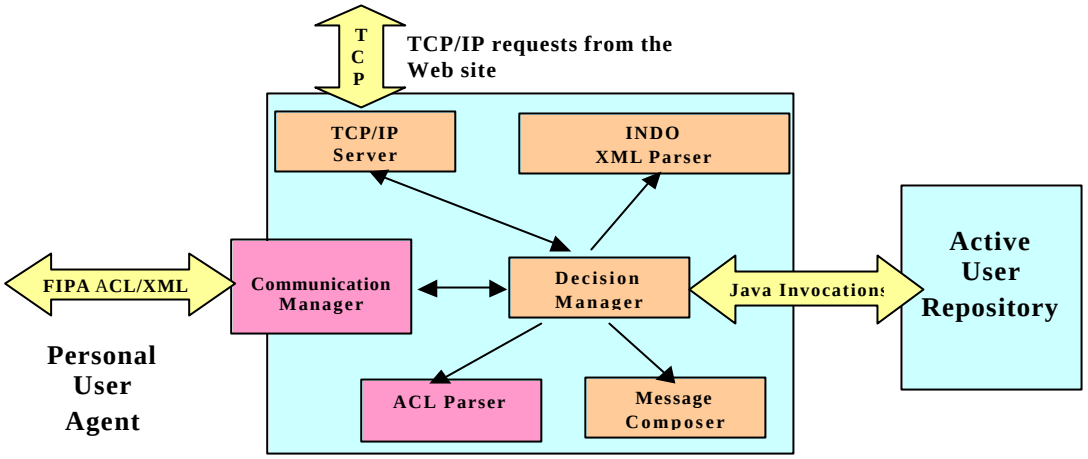

Figure 49: Personal User Agent Internal Architecture

In the following sequence diagram, the internal steps that the PUA is undertaking to start a business process for a given user are provided. The other type of operations, like suspend, resume and terminate a process instance or ask the status of a process instance are performed in a similar way. Initially, a start process instance request is generated by the end-user and sent to the PUA through a TCP/IP connection. The TCP/IP server of the PUA agent gets the request of the user and forwards it to the Decision Manager. The Decision Manager identifies the nature of the request, i.e. to a start process, creates a new entry in AUR, and composes a start process ACL/XML message that the Communication Manager sends to the DR agent. When the process instance has started (see section about DR), the DR agent informs the PUA agent, by sending a FIPA ACL/XML *inform* message, about the successful instantiation of the process instance, i.e. the process instance is running. The Communication Manager forwards it to the Decision Manager, which parses it with the help of ACL and XML parser. Afterwards, the PUA checks semantically the message, updates the AUR and forwards the response to the TCP/IP server. The TCP/IP server forwards the request to the web server and finally to the user.

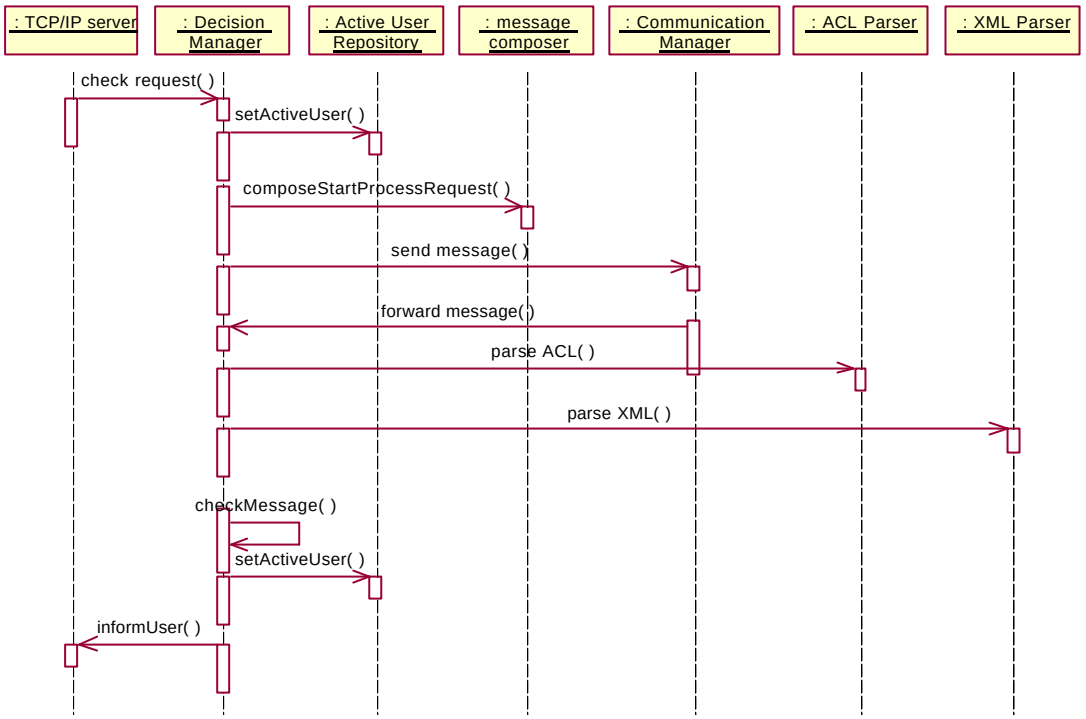

Figure 50: Start Process Sequence Diagram

In a similar manner, when a particular process instance has been completed, the DR agent sends to the PUA agent a FIPA ACL/XML message informing him that the process instance has completed. The message contains the name of the process, the instance id, and the set of output parameters and values of the process. The PUA agent parses the message using the ACL and XML parser and checks the type of inform message. Since the message is *completed*, the Decision Manager informs the Active User Repository by setting the status of the process and the output parameters and values. In the sequel, the Decision Manager informs the user about the new status of the process and the output results. The internal steps that the PUA performs to provide this operation are depicted in the following figure.

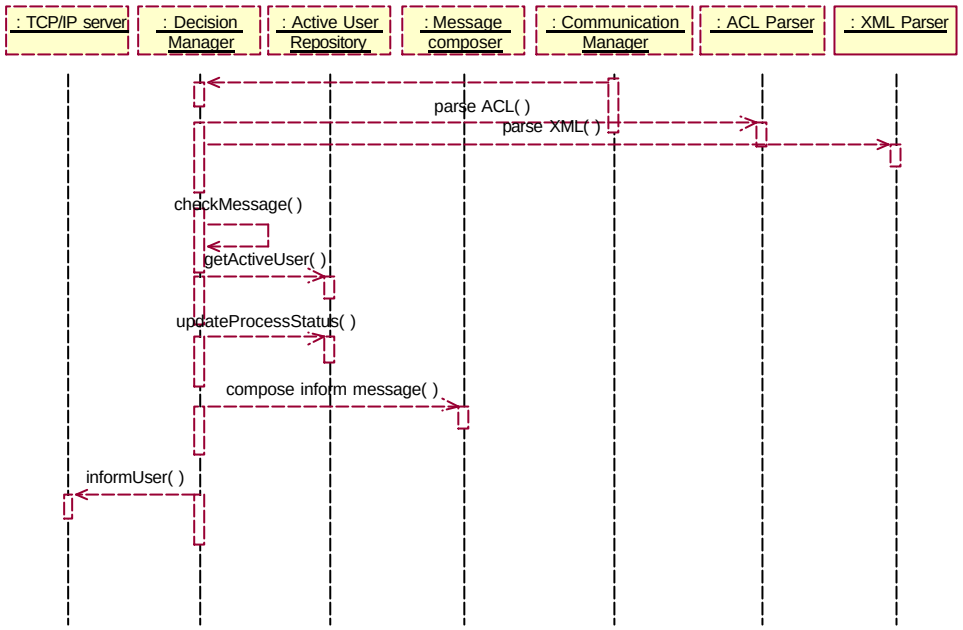

Figure 51: Process Instance Completes Sequence Diagram

In the following section, details about the functionality and internal modules of the DR agent and how it operates to perform its key operations are provided.

8.3 Domain Representative Agent

The Domain Representative agent is the central communication point within one administrative domain. The DR is responsible for managing the requests and responses among the agents executing the business processes within the domain and the users or the remote domains. The DR agent should be in every administrative domain independent if the domain plays the role of the VE representative, the VE partner or the VE candidate partner.

The DR agent gets requests from the following agents:

- PUA when a user starts a new process or manages an existing one. In that case the DR is located in the VE representative domain and the communication among the DR and PUA agents is intra-domain communication.
- Remote Workflow Provider Agents when a remote process should be started or managed. In that case the DR is located on a VE partner domain and the communication among the DR and remote WPA agents is inter-domain.

The different type of requests that the DR can get from these two agents are to start a process, to suspend a process, to resume a process, to terminate a process or to ask for the status of a process. The communication among the agents is performed based on message exchanges, i.e. FIPA compliant ACL/XML messages. The content of the messages follows the inter- or intra-domain ontology accordingly.

The DR agent always checks the identity of the requesting agents and authorizes or not the execution of the process. In the first case, i.e. when the request is coming from the PUA, the authorization of the user is performed by the PUA and thus, no extra authorization is performed. The PUA agent considered by the DR a trusted agent. In the second case, i.e. when a remote WPA requests the execution of a business process, the DR always checks whether this agent has the right to access this process. This performed by conducting the Contract Repository.

The Contract Repository maintains a list of contracts that have been established with remote domains. Every contract is a result of a successful negotiation process between a remote requestor domain and this domain. The meaning of the contract is strictly technically and not legal[14]. The existence of a contract concerning a particular local process means that a specific WPA agent from a specific remote domain can access this process. The contract has a limited duration and lasts only during the execution of the local process. When the requested local process has been completed, then the contract and its corresponding id is deleted. In that way, processes that are heavily requested by remote domains can be provided with different pricing schemes while processes that have not been requested can be provided with better prices.

In general, a contract has always a unique id, the date of issue, is characterized as remote or local and contains data related to both domains. A contract is characterized local when the domain should provide the process to another domain. On the contrary, a contract is characterized remote when the domain deploys the business process from another domain. Additionally, the information stored for every domain can be categorized in three sections, namely the technical, the administrative and the pricing[15] section. In the technical section, the name of the agents involved in the provision and management of the process, i.e. the DR and WPA agents, the FIPA address of these agents (see GUID in chapter 4), the ontology used, i.e. the inter-domain ontology, and the protocol used, i.e. the FIPA compliant Request-Response protocol, are stored. In the administrative section, the name of the domains, the physical addresses, the name, telephone, fax and e-mail address of the administrative person, etc. In the pricing section, the price agreed, the payment method, the payment deadline, and the bank ids, are stored.

The following XML-DTD specification determines the format of a legitimate contract that can be established among two VE partners.

```
<!-- Contract Repository -->
<!ELEMENT contract (Contract_ID, DateOfIssue, IsRemote, RequestorSection, SupplierSection)>

<!-- Contract Characteristics  -->
<!ELEMENT Contract_ID (#PCDATA)>
<!ELEMENT DateOfIssue (day_entity)
<!ELEMENT day_entity (day, month, year)>
<!ELEMENT day (#PCDATA)>
```

[14] The interesting reader might get more information about the legal aspects of contracts in the following reference: *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data*, OJEC L 281 of 23 November 1995 and *European Directive EC/97/7 of the European Parliament and the Council of 20 May 1997 on the protection of consumers in respect of distance contracts*, OJEC L 144 of 4 June 1997, and on *COM (97) 503 "Towards A European Framework for Digital Signatures And Encryption*, available on http://www.ispo.cec.be/eif/policy/97503.html

[15] Accounting mechanisms for inter-domain usage of services is out of the scope of this thesis. However, the interesting reader can have more information about that on "Introduction to Accounting Management" by B. Aboba, J. Arkko, D. Harrington in the following IETF draft-ietf-aaa-acct-03.txt, 02-May-00

```
<!ELEMENT month (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT IsRemote(#PCDATA)>

<!-- Contract Main Entities  -->
<!ELEMENT    RequestorSection    (Technical_Section,    Administrative_Section,
Pricing_Section)>
<!ELEMENT   Supplier   Section   (Technical_Section,   Administrative_Section,
Pricing_Section)>

<!-- Contract Technical section specification  -->
<!ELEMENT Technical_Section (agentName, FIPA_Address, Ontology, Protocol)>
<!ELEMENT agentName (#PCDATA)>
<!ELEMENT FIPA_Address (#PCDATA)>
<!ATTLIST  Technical_Section  Ontology  (Inter-Domain  |  Intra-Domain  )  'Inter-
Domain' #REQUIRED>>
<!ATTLIST  Technical_Section  Protocol  (Request-Response  |  Query-Response  |
Contrat-Net) 'Request-Response' #REQUIRED>>

<!-- Contract Administrative section specification  -->
<!ELEMENT Administrative_Section (DomainName, Address, telephone, fax, e_mail)>
<!ELEMENT DomainName (#PCDATA)>
<!ELEMENT Address (address_entity)>
<!ELEMENT address_entity (street, city, zipcode, country)
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT zipcode (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT telephone (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>

<!-- Contract Pricing section specification  -->
<!ELEMENT Pricing_Section (price, paymentMethod, paymentDateline, BankZIp)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST pricing_Section paymentMethod (visa | eu-card | bank-transfer) 'visa'
#REQUIRED>>
<!ELEMENT paymentDateline (day_entity)>
<!ELEMENT BankZip (#PCDATA)>
```

All the contracts are stored in a conventional file system as ASCII XML files that follow the previously mentioned format. In addition to that, a configuration file called *contract_repository.xml* maintains all the existing contracts, i.e. the names of the contract files.

The main operations that the contract repository provides are to create a new contract and to delete an existing one. Modification of the contract is not allowed as soon as a contract has been created. The main reason is that existing contracts have been agreed on during negotiation and of course can not be changed individually by one domain without prior knowledge of the other. In the following Figure 52 the class model, the key entities and the relationships of the Contract Repository are depicted.

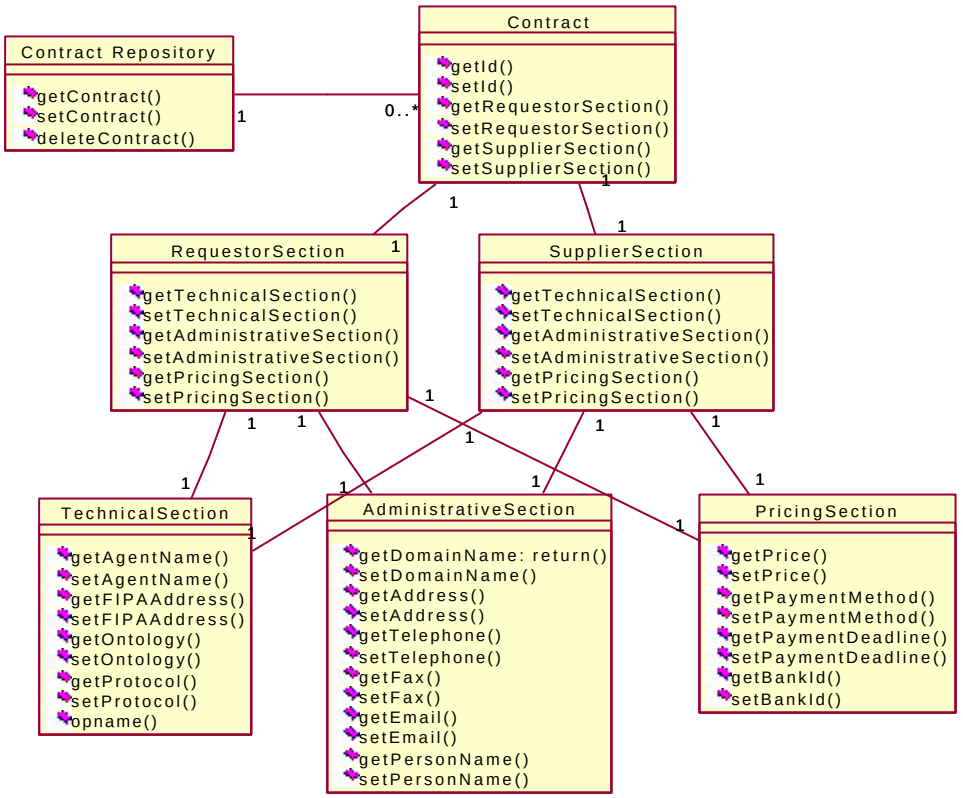

Figure 52: Contract Repository Class Diagram

The Contract Repository used by the Provider Negotiation Agent (PNA) when a successful negotiation process has been performed. In that case, when the negotiation completes successfully, the PNA creates a new contract XML file, stores it into the file system and updates the *contract_repository.xml* configuration file about the new contract. In the sequel, when a request about a process execution arrives from a remote domain, the DR retrieves the corresponding contract, checks the contract id and the name of the requesting agent, and enables the execution or not of the requested process. In such a way, access control and authentication is performed.

When the DR gets a request from the PUA or from a remote WPA it functions in the following way: if the request is to start a process, the DR instantiates a new WPA agent and forwards to him the request. The request contains the name of the process, the process instance, the contract id, and the input parameters and values. From now on, the newly created WPA is responsible for the execution of the requested process. If the request is to suspend, to resume, or terminate a process, the DR locates the corresponding WPA agent for this process and forwards the request to him.

In order the DR to perform the mapping between the running process instances and the corresponding WPA agents executing these instances, the DR maintains a List of Active Processes (LAP). Whenever a new instance is instantiated, a new entry is created on the LAP. Whenever an instance terminates, completes, or aborts the DR deletes the corresponding entry from the LAP. The LAP entries are actually records containing the process name, the instance

id, the contract id, the name of the WPA agent, and the name of the agent that initially requested to start the instance of the process.

Whenever a process instance changes status, it informs the initial requestor agent of this instance. This means that the responsible for this instance WPA, informs the DR about the status change by sending a FIPA ACL/XML message following the intra-domain ontology. The DR checks on the LAP to find the initial requestor agent, constructs a legitimate ACL/XML message, and forwards it to the related agent. It should be noted that responses about status changes are performed after corresponding requests have been issued. This means that a process instance will send a *suspend status* message only after getting a request to suspend. The only case that responses are generated without prior requests is when a process completes or when a process aborts. In that case, the process instance changes automatically status and informs the other agents about the change.

In addition to the key entities of a FIPA compliant agent, the DR agent contains the following modules:

- **INDO XML Parser**: responsible for parsing the content of the FIPA ACL messages based on the intra-inter domain ontology,
- **INDO Message composer**: responsible for composing the appropriate response FIPA ACL-XML messages related to the local WPA, PUA and remote WPA agent. The structure of the messages is based on the inter-intra domain ontology,
- **Decision manager**: responsible for controlling the basic operations of the agent, communicating with the Contract Repository, the LAP and the other entities
- **List of Active Processes (LAP)**: responsible for maintaining all the active local processes running on this domain.

In the following picture the internal architecture of the DR agent and the relationships among the basic modules is depicted.

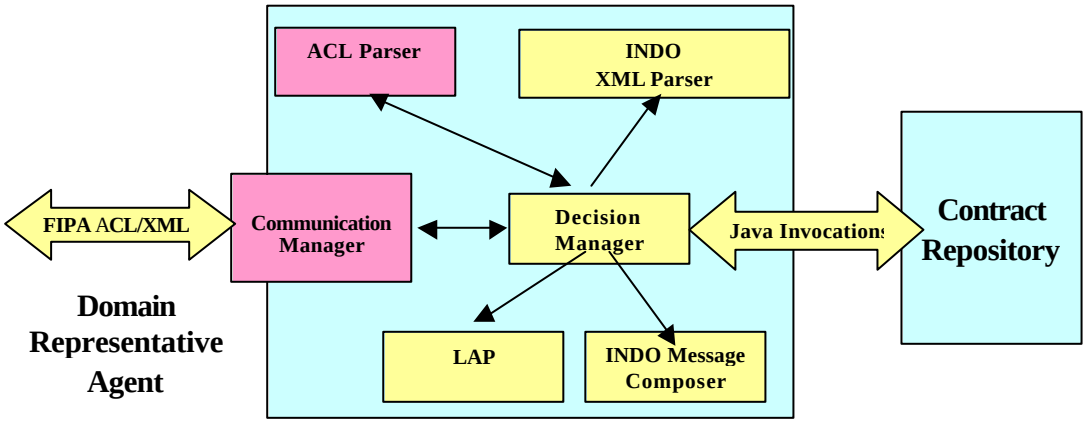

Figure 53: Domain Representative Agent Internal Architecture

In the following sequence diagram, the internal steps that the DR agent follows to serve a request for a new process instance are provided. Initially, the DR gets an ACL/XML message from a remote WPA. It first parses the message using the ACL and the inter-domain XML

parsers. In the sequel, the DR checks the type of the request, i.e. start new process instance, and conducts the Contract Repository to check if there is an existing contract with this remote WPA agent. If a contract already exists, the DR creates a WPA agent using the native commands of the agent platform, i.e. the Grasshopper, updates the List of Active Processes and composes an ACL/XML message. The request message for the creation of new process instance is send to the newly created WPA. The message contains the name of the process, the instance id assigned to this process instance, the input parameters and the values for each one of them. Requests for the creation of new process instance coming from the PUA are served in a similar way. The only difference is that the contract repository is not involved since the PUA agent is considered a trusted agent.

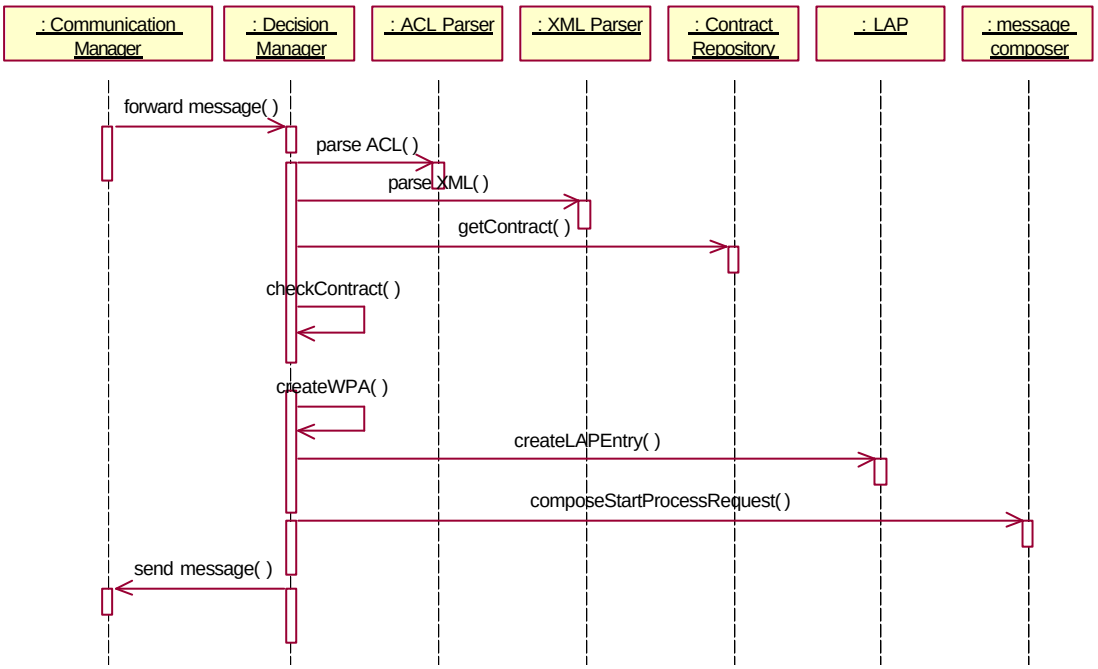

Figure 54: Start Process Sequence Diagram

In the following sequence diagram, the internal steps that the DR agent follows to serve a response coming for an existing process instance and should be forwarded to the initial requestor agent, i.e. the PUA or the WPA discussed. Initially the DR gets an ACL/XML response message from one of its local WPA, parses it with the ACL and XML parser, and then checks the LAP. If the response message is terminated, aborted, or completed, the DR deletes the corresponding entry from the LAP. In addition to that, in case that the process has been completed, the corresponding contract has been fulfilled and thus, should be deleted from the contract repository, i.e. the *contract_repository.xml*. Access to this local process by this remote WPA is not valid any more unless a new contract will be established. In the sequel, the DR agent creates a legitimate response ACL/XML message and forwards it to the agent that started the process.

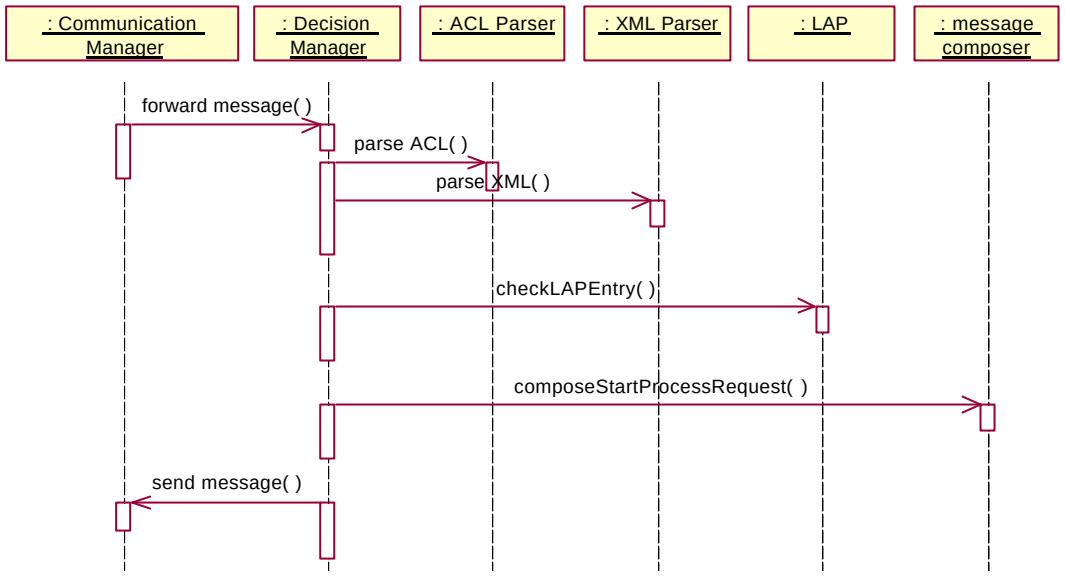

Figure 55: Response Sequence Diagram

In the following section the key operations of the WPA agent are provided and certain details about the internal architecture, as well as, the modules are provided.

8.4 Workflow Provider Agent

The Workflow Provider Agent (WPA) is responsible for the execution and management of a business process instance. The execution and management of a process instance is done in a distributed, autonomous and co-operatively way by a set of WPA agents.

In general, the life-cycle and status of a WPA are the same like the life-cycle of a business process instance, i.e. a WPA exists, as soon as, the corresponding process instance exists. If the process instance terminates, aborts or completes, then the WPA dies out. Additionally, the WPA, in a similar way like the process instance, can be in different states based on the status of the instance that it executes. Therefore, when the process instance status is running, then the WPA is also running, while when the process instance status is suspended, then the WPA suspends. If the process instance status is aborted then the WPA also aborts, while when the process instance status is completed, then the WPA also completes.

In general, the following conditions are hold for the status of a WPA and its co-operating agents:

- a WPA is running when all of its sub-process WPAs and RPAs are running,
- a WPA is suspended when all of its sub-process WPAs and RPAs are suspended,
- a WPA is resumed or running again when all of its sub-process WPAs and RPAs are suspended,
- a WPA is terminated when all of its sub-process WPAs and RPAs are terminated,
- a WPA is aborted when at least one of the sub-process WPAs or RPAs is aborted,

- a WPA is completed when all of its sub-process WPAs and RPAs are completed.

A WPA can be instantiated by either the DR, when a new process has been requested, or by another WPA, if the originator WPA needs to start a new sub-process for this process. In both cases, the creation of the WPA by either, the DR, or the WPA is provided by the underlying agent platform. Initially, the parent agent instantiates the new WPA and sends him a message to start executing a specific process, with a given name, instance id, and given set of input parameters and values. The newly created WPA retrieves from the Business Process Repository the specification for this process and starts the execution of it.

Due to the fact that different instances of the same process might simultaneously exist, there should be a way to differentiate the instances and consequently the different WPAs that provide them. For that reason, the concept of the instance id has been used. For every given process, there are different instances with different ids. The combination of process name and instance id is unique and corresponds to only one active process instance. The instance ids are created dynamically by the DR, when a new process instance is started, or by the parent WPA when a new sub-process starts.

Additionally, there should be a mechanism for the agent platform to differentiate the WPAs that take part in the execution of a particular instance. For that reason, the name of the WPA has the following format "*process name&instance id*". The same is hold for the sub-processes that also have a unique process name and a unique instance id. In that case, the parent WPA generates a new instance id, it stores it internally, and then creates a WPA with the following name "*subprocess name&instance id*". The instance id and the names of WPAs are vital information for message exchanging and are stored inside the WPA in the List of Active Sub-Processes (LAP). The LAP, in a similar way like in DR, stores the names and instances of all the sub-processes and atomic process and the corresponding names of the WPA and RPA agents that currently provide them. Due to the fact that the communication among the agent is based on the FIPA ACL, the names of agents are playing a very important role.

The type of request messages that the WPA can get from the DR or the parent WPA are the same as in the case of the DR, i.e. to start a process, to suspend it, to resume it, to terminate it, or to ask about the status of the process. For every request, a response is generated. The response actually contains the name of the process, the instance id and the current status of the process instance. The status of a process instance can be running, suspended, aborted, terminated, or completed. Particularly, in case that the process instance aborts or completes, no request message is required. The process instance, i.e. the WPA, creates the corresponding status messages, when the status has been changed, and sends them back to the DR or the parent WPA. Additionally, in case that the process has completed, the values of the output parameters are also included into the message. In that way, sub-process instances can always exchange data with the parent process, i.e. the output values of a process might be input values for another sub-process. The communication among the DR and WPA, WPA and WPA, or WPA and RPA is based on the FIPA compliant request-response protocol, while the ontology used for the description of messages is the intra-domain ontology.

In general, a business process consists of sub-processes and external tasks. For every sub-process, a WPA is created for executing the sub-process, while for every external task, a RPA is created for executing the atomic process. Initially, when a new process is requested, the DR creates a WPA that is responsible to execute the requested process. If the process has a sub-process, then the WPA creates a new WPA and assigns to him the responsibility to execute the sub-process. If the process has an atomic process, then the WPA creates a RPA and delegates to

him the responsibility to execute the atomic process. This means that the execution of a process instance is provided autonomously and distributed by a set of WPAs and RPAs that co-operate to execute the process and thus to accomplish a business goal. The co-ordination is achieved by message exchanges. The format of the messages is FIPA compliant ACL/XML while the ontology used is the intra-inter domain ontology. The basic algorithm and mechanism for the co-ordination of agents during process execution has been explained in the process management section. Details about how the co-operation mechanism is achieved are provided subsequently.

In addition to the key entities of a FIPA compliant agent, the WPA agent contains the following modules:

- **INDO XML Parser**: responsible for parsing the content of the FIPA ACL messages based on the intra-inter domain ontology,
- **INDO Message composer**: responsible for composing the appropriate response FIPA ACL-XML messages related to the sub-process WPAs, and the parent WPA or the DR agent. The structure of the messages is based on the inter-intra domain ontology,
- **Decision manager**: responsible for controlling the basic operations of the agent, communicating with the Workflow Engine, the LAP, and the other entities
- **List of Active Processes**: responsible for maintaining all the active sub-processes and the corresponding responsible WPAs,
- **Workflow Engine**: responsible for controlling the execution and management of the process instance in relation to sub-processes and external tasks.
- **External Condition Checker**: responsible for evaluating the conditions associated with every sub-process and for informing the Workflow Engine about when one of them becomes true.

In the following picture the internal architecture of the WPA agent and the relationships among the basic modules is depicted.

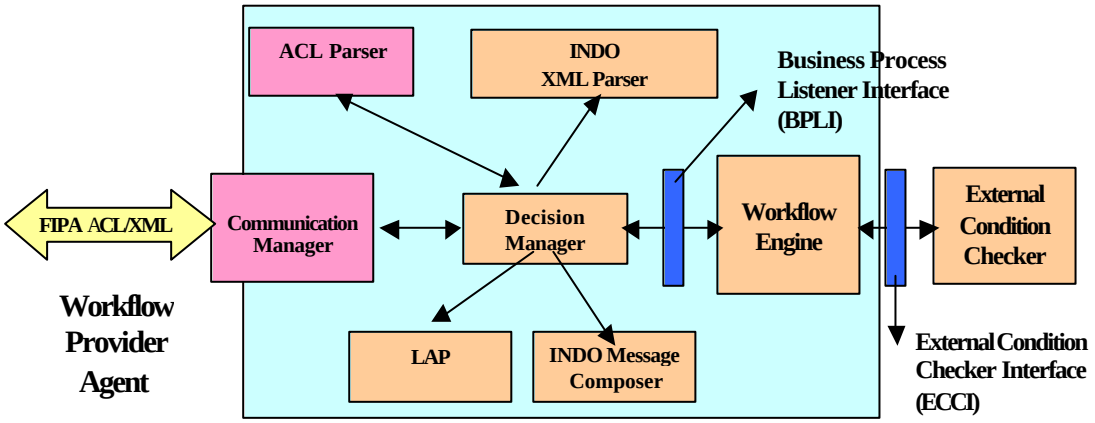

Figure 56: Workflow Provider Agent Internal Architecture

The entity within the WPA responsible for controlling the execution of the process instance in relation to sub-processes and atomic processes is the Workflow Engine (WE). The WE maintains the status of the process instance and all of the sub-processes and atomic processes,

evaluates the conditions associated with every sub-process, generates asynchronously the requests that will be forwarded to the sub-processes and atomic processes, and generates asynchronously the inform messages that will be sent to the parent WPA or DR containing the process status changes. The asynchronous generation of request messages is performed automatically in the form of events sent to the Decision Manager of the agent by the WE. For that reason, a special interface has been specified called the **Business Process Listener Interface** (BPLI).

The Business Process Listener Interface is actually a set of operations that the Decision Manager should provide in order to receive the notifications generated by the WE. The WE invokes these operations asynchronously when certain conditions are met. Actually, all the messages that the WPA can send to its sub-processes, atomic processes, parent WPA or DR are generated by the operations of the Process Listener Interface. The class model of the WE is depicted in the following figure and is strongly related to the Definition Model described into the Business Process Repository section. In general the WE is instantiated automatically when a new process instance is requested by the BPR. The WE provides the basic mechanisms to update the status of sub-processes, to insert input and output values, to suspend, resume, terminate, or abort the instance and so on. Subsequently the basic mechanism of the WE will be explained in relation to the WPA operations.

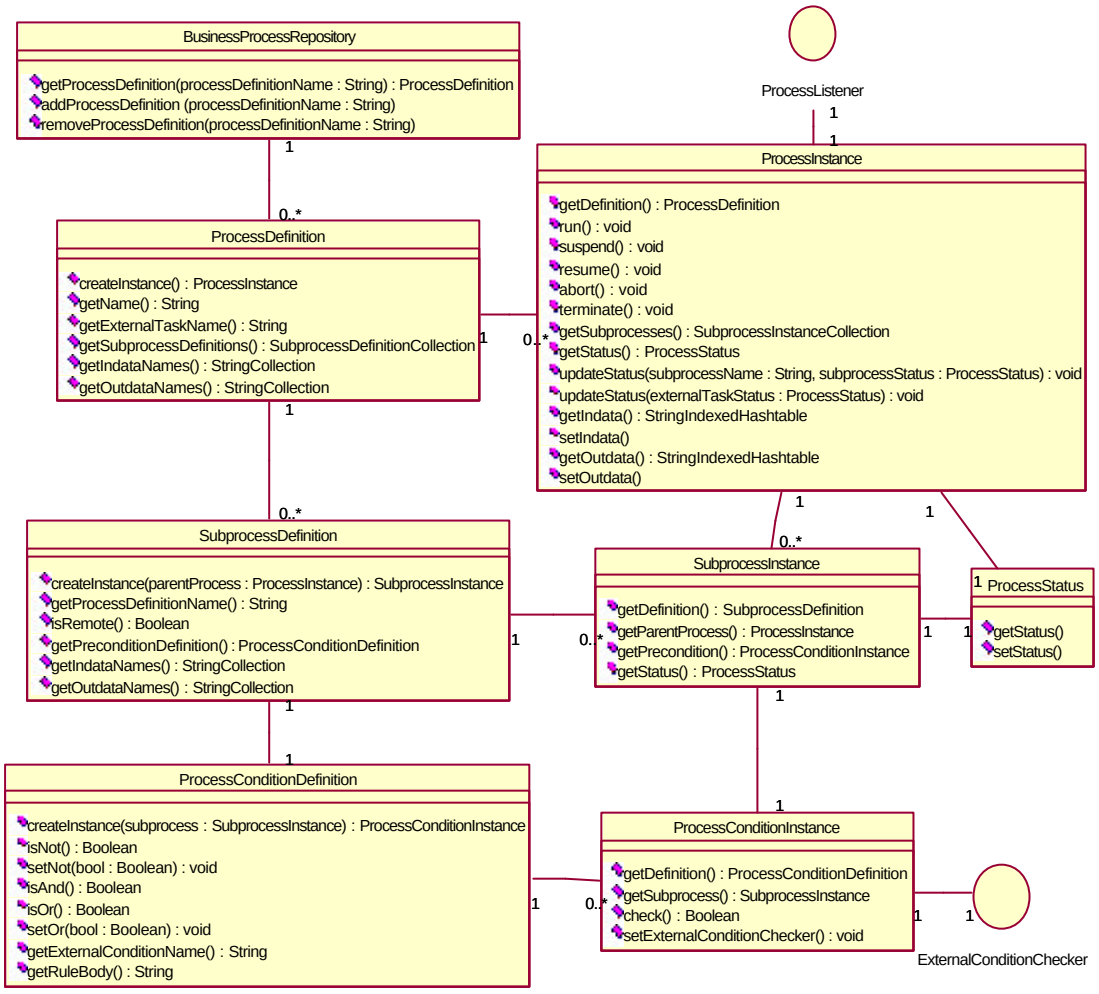

Figure 57: Workflow Engine Class Model

The operations of the Business Process Listener Interface are divided into three major categories, namely the operations related to the parent WPA or DR, to the sub-processes and to the atomic processes.

The operation related to parent WPA or DR is the:

- notifyProcessStatusChanged: the status of the process has changed. The WE notifies the agent that the process instance has changed status and thus, the agent should inform its parent about that. The Decision Manager generates an inform message with the new status and sends it to the parent WPA or DR. In case that the new status is completed, then the output parameters and values are extracted by the WE and inserted into the message.

The operations related to sub-processes are:

- notifySubprocessNeedsToRun: The WE notifies the agent that the given sub-process should start its execution because its pre-conditions have been evaluated to true. The Decision Manager creates a new WPA agent, composes a request message with the name of the sub-process, the input parameters and the input values, and sends it to the newly created WPA.

The input parameters and the values of the new sub-process are provided by the WE.

- notifySubprocessNeedsToSuspend: The WE notifies the agent that the given sub-process should suspend its execution. The Decision Manager composes a *suspend* request message and sends it to the corresponding sub-process WPA responsible for this sub-process.
- notifySubprocessNeedsToResume: The WE notifies the agent that the given sub-process should resume its execution. The Decision Manager composes a *resume* request message and sends it to the WPA responsible for this sub-process.
- notifySubprocessNeedsToTerminate: The WE notifies the agent that the given sub-process should terminate its execution. The Decision Manager composes a *terminate* request message and sends it to the WPA responsible for this sub-process.
- notifySubprocessNeedsToAbort: The WE notifies the agent that the given sub-process should abort its execution. The Decision Manager composes an *abort* request message and sends it to the WPA responsible for this sub-process.

The operations related to the external tasks are similar to the previous ones. These operations are:

- notifyTaskNeedsToRun: The WE notifies the agent that the given atomic process should start its execution. The Decision manager creates a RPA agent, composes a request message with the name of sub-process, the input parameters and the input values and sends it to the newly created RPA. The input parameters and the values of them are provided by the WE.
- notifyTaskNeedsToSuspend: The WE notifies the agent that the given atomic process should suspend its execution. The Decision manager composes a *suspend* request message and sends it to the RPA responsible for this atomic process.
- notifyTaskNeedsToResume: The WE notifies the agent that the given atomic process should resume its execution. The Decision manager composes a *resume* request message and sends it to the RPA responsible for this atomic process.
- notifyTaskNeedsToTerminate: The WE notifies the agent that the given atomic process should terminate its execution. The Decision manager composes a *terminate* request message and sends it to the RPA responsible for this atomic process.
- notifyTaskNeedsToAbort: The WE notifies the agent that the given atomic process should abort its execution. The Decision manager composes an *abort* request message and sends it to the RPA responsible for this external task.

When the WPA is created, it gets the first request message from its parent WPA or the DR to start a given process, with a given name, instance id and given input parameters and values. The WPA gets from the message the process name, the input parameters and the values of these parameters. Afterwards, the WPA retrieves from the Business Process Repository the Definition Model for this process and instantiates the Workflow Engine by creating a process instance. From now on, the WE will control the status of the process and all of its sub-processes. After the instantiation of the WE, the WPA inserts the values of the input parameters into the Process Instance object and calls the run method. The previously described steps are depicted in the following sequence diagram.

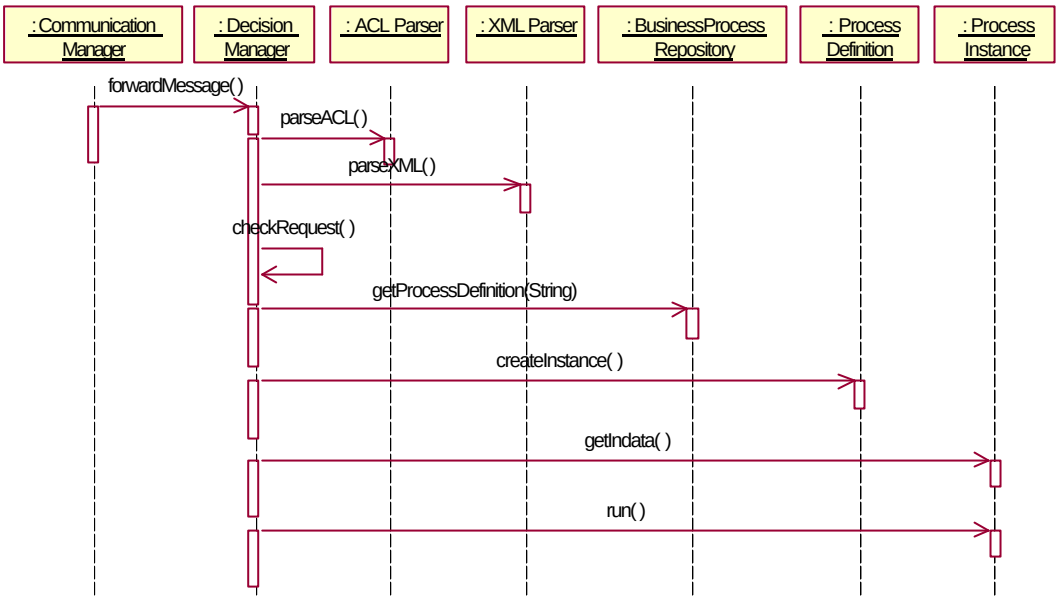

Figure 58: Instantiation of the Workflow Engine

When the run method is called, the status of the process instance is changing to "running". Automatically and in an asynchronous way, the WE creates an event notifying the decision manager about that. The WE actually invokes the *notifyProcessStatusChanged* operation of the Process Listener Interface. The Decision Manager, upon receipt of the notification, gets from the LAP the name of the parent of the process, generates a status inform message and sends the message to him. The *inform* message contains the status of the process which in that case is *running.* The same procedure is followed for all the process status changes, i.e. when the process is suspended, resumed, terminated or completed. In all cases the WE invokes the corresponding *notifyProcessStatusChanged* method of the Process Listener Interface and then the Decision Manager functions accordingly. In case that the new status of the process is completed, then the output parameters and values should be sent to the parent process too. The output parameters and values are extracted from the Workflow Engine. When the new status of the process is terminated, aborted, or completed the WPA dies out after he sends the message to his parent. The steps involved in this process are depicted in the following sequence diagram.

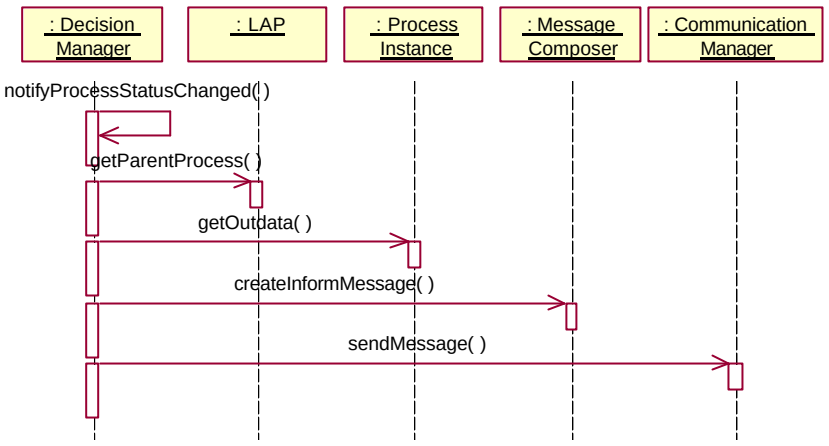

Figure 59: Creation of Process Status Inform Message

All the previously described operations explain what the WPA does in order to send a message to other agents. On the contrary, when a WPA gets a message it functions as follows: the WPA initially parses the ACL/XML message, gets the content of the message, and checks the type of the message. Two types of messages might arrive:

- **request messages**: these messages are sent from the parent of the WPA to the WPA. Potential messages in this category are request for run, suspend, resume, abort, or terminate. This is actually how the parent process forwards to the lowest levels of the process any type of events created in higher levels, e.g. the end-user requested to suspend the process.
- **inform messages**: these messages are sent from the sub-processes or atomic process to the WPA when their process status has changed. Potential messages in this category are status changes like suspended, resumed, aborted, terminated, or completed. This is actually how the sub-processes inform its parent about any type of events occurring in the lowest levels.

In the first case, when a WPA gets a request like suspend, resume, terminate or abort message from its parent WPA, it invokes the corresponding method provided by the process instance object. The following sequence diagram explains the steps involved in this type of operations.

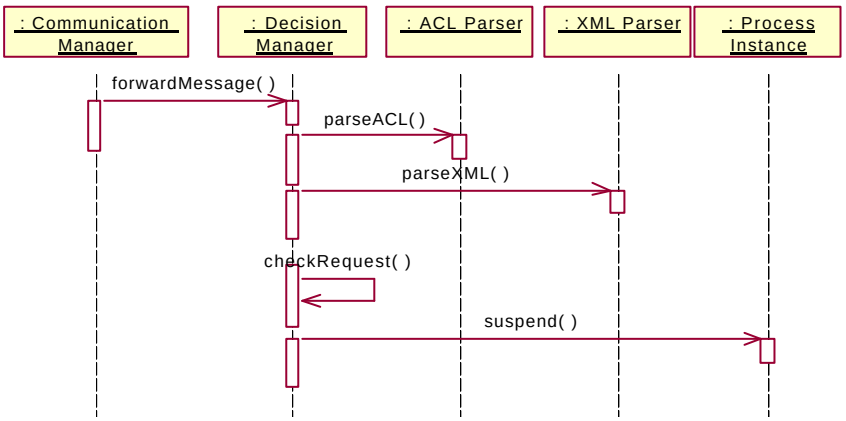

Figure 60: Suspension of a WPA

When a WPA gets a suspend request from its father, it informs the WE. In that case the WPA should suspend all of each sub-processes. In that case, the WE notifies the WPA by invoking the *notifySubProprocessNeedsToSuspend* that all active sub-processes of this process should be suspended. Then the Decision Manager, that asynchronously receives the notification, locates the names of the sub-process WPAs from the LAP, composes an inform message, and sends it to all the sub-process WPAs. The same steps are followed when a resume, terminate or abort request is received. The following sequence diagram shows how these steps are provided by the different entities.

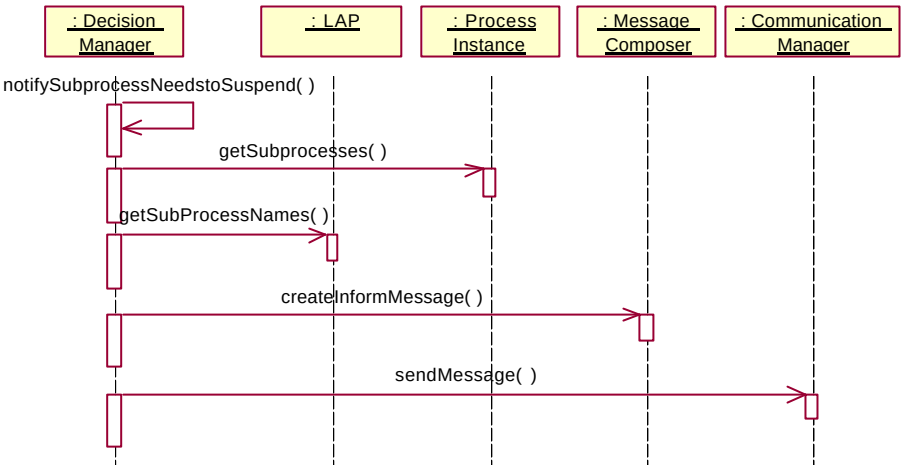

Figure 61: Suspension of a sub-process WPA

In the second case, when a WPA gets an inform process status message like suspended, resumed, terminated or aborted from its sub-processes, it always invokes the *updateStatus* method provided by the process instance object. Additionally, if the inform message is completed, the output parameters and values are extracted from the incoming message and inserted into the WE. The following sequence diagram explains the steps involved in this type of operations.

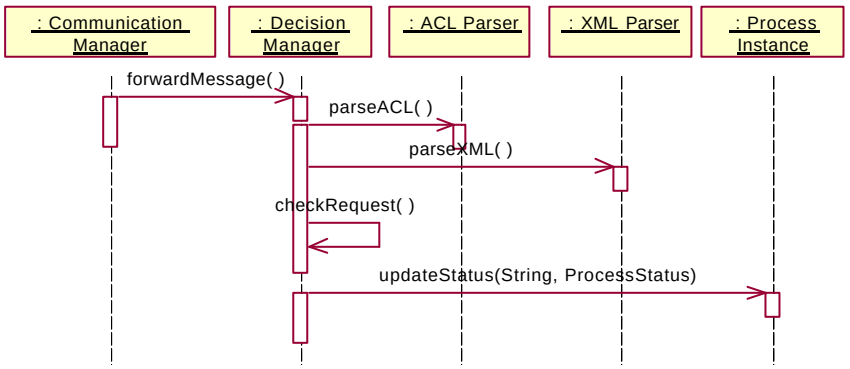

Figure 62: Update Status Sequence Diagram

Based on the above description, it is clear that the WE plays a significant role in the execution and management of a business process. In general, the WE maintains for the process instance and all the sub-processes the current status and values of input and output parameters. Therefore,

when the status of the process changes then the WE generates the corresponding events. When the conditions related to a sub-process are evaluated to true, then this sub-process should start its execution. This is performed by the generation of the appropriate event, i.e. *notifySubProcessNeedsToRun*. The evaluation of the pre-conditions is performed from a third party Condition Checker.

The External Condition Checker is actually a module that evaluates logical conditions included in the business process specification. The logical conditions have been specified in terms of logical operators, input and output parameters and certain values. The conditions are expressed in a specific language that the External Condition Checker can understand. In the context of this thesis, the Java Expert System Shell (JESS) has been used. JESS is an easy to use expert system written totally in Java that provides all the basic operations of expert systems. JESS has a well-defined condition specification language that is being used for the specification of conditions. However, a general interface has been built among the WE and the External Condition Checker, so as different Condition Checkers can be integrated. The **External Condition Checker Interface** (ECCI) enables the WE to update the values of parameters within the JESS database. The conditions have been specified during process specification and of course, cannot change dynamically. When a new process instance is created, the conditions related to the sub-processes are inserted into the JESS database. Then, during the process execution, the evaluation of conditions is performed from the Condition Checker.

The External Condition Checker Interface is the link between the external Rule Engine and the Workflow Engine through the process condition instances. Classes implementing this interface should do the following things:

- Add rules to the rule engine whenever necessary. In practice, this has to be done every time a new process condition instance representing an external condition is created and the rule engine should take care of evaluating the rule,
- Subscribing the process condition instance to changes in the rules it is interested in. To do this, the external condition checker provides a method the process condition instance can call to notify its interest in a rule, and the external condition checker has to provide a mechanism internally to make sure that any notifications are forwarded to the correct process condition instance.
- If the rule engine allows so, the external condition checker should provide a method to do backward chaining on a rule and return the result of it back to the process condition instance. If the rule engine can't handle backward chaining, the external condition checker has to throw an exception of type *CannotCheckConditionException.*

The following figure clarifies the design and implementation issues.

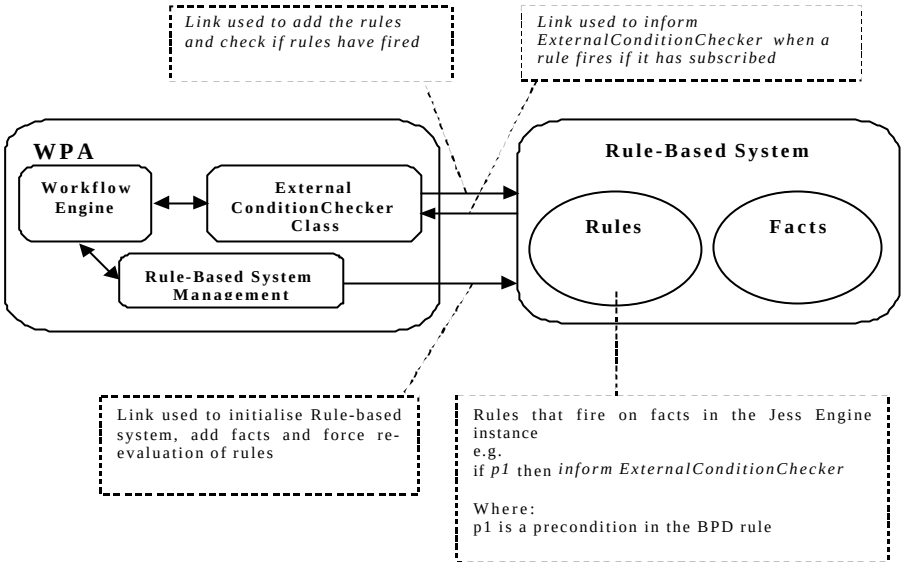

Figure 63: Interactions between the WPA and the Rule-Based System

The Rule-Based system must be able to store *'if p then a'* type rules, where 'a' is an action, and 'p' is a precondition. This pre-condition must be specified in terms of facts that are also stored in fact database. When a fact is inserted into the fact database all rules should then be evaluated. If any pre-condition becomes true due to this new fact, then the action should be executed. It is very important that this rule can only be fired once, the pre-condition should only be evaluated once. It is also important that rules for each workflow engine instance are unique within the Rule-Based system. This can be achieved either by, having multiple instances of the Rule-Based system, for example one rule-based system instance for one process instance, or having one instance of the Rule-Based system and marking the rules and facts as belonging to a particular process instance as they are entered.

The Rule-Based system must provide methods to implement the following functionality:

- initialise the Rule-Based system,
- adding a rule,
- adding a fact and re-evaluating all rules by either Informing the ExternalConditionChecker class, when a rule fires or if the Rule-Based system implements backward chaining checking if a rule has fired

The *ExternalConditionChecker* Interface provides the functionality needed for the Workflow Engine to be able to add rules to the Rule-Based System. It can also inform the Workflow Engine when a rule fires within the Rule-Based System only if the Workflow Engine has requested this using the *notifyNeedsChangeUpdates* method. A rule fires, when its pre-condition becomes true. Additionally, the Workflow Engine is able to find out if a rule has fired by asking the external condition checker class. Therefore, a method *check* must be implemented to provide this functionality. If the Rule-Based System is unable to implement backward chaining then the *ExternalConditionChecker* class must throw a *CannotCheckConditionException* whenever *check* is called. When a Workflow Engine asks the *ExternalConditionChecker* to add a rule, the rule

body is retrieved from the BPD. The *ExternalConditionChecker* informs the Workflow Engine when a rule fires only if the Workflow Engine has first called the *notifyNeedsChangeUpdates* method. The JessExternalConditionChecker class implements the *IExternalConditionChecker* interface. It provides the functionality needed for the process engine to be able to add rules to the JESS Engine. It also informs the process engine when the precondition of a rule within the JESS Engine becomes true, causing a rule to fire.

In order to clarify the way the WE is cooperating with the Condition Checker an example is given. Lets assume that a process has two sub-process A and B. Sub-process A has two output parameters namely, *"requested_items"* and *"payment_method"*. Additionally, sub-process B has the following condition: *"requested_items"* > "10" AND *"payment_method"* = "VISA". This means that when this condition becomes true then the B sub-process should start. Initially, during the execution of A, both parameters have no values and thus, the condition is evaluated from the condition checker to false. Therefore, the B sub-process cannot start. When the sub-process A completes, it forwards its output parameters to the parent WPA, which in the sequel informs its WE. The WE gets the values of the parameters and passes them into the Condition Checker for evaluation. The Condition Checker evaluates the condition and if it is true, it informs the WE accorndigly. The WE locates the name of the process associated with the condition and generates the event *notifySubProcessNeedsToRun* that the Decision Manager will further process. This means that whenever the process or one of its sub-processes and atomic processes change status, the WE is informed and accordingly, the Condition Checker starts the evaluation of the conditions.

When a new sub-process should start its execution, the Decision Manager always checks if the sub-process is local or remote. If the process is local then the previous described steps are followed, i.e. a new WPA is created and a request to start the sub-process is sent. However, when the process has been specified as remote, then a different procedure is followed.

In case that the sub-process is remote, the Decision Manager creates a Requestor Negotiation Agent (RNA) and informs him about the remote process name, the input and output parameters, the initial input parameter values and default constraints assigned to them during specification phase. The RNA is responsible for "moving" into the marketplace, locate all the potential VE candidate partners and start a negotiation process among them to locate the best VE partner. The steps involved within the WPA are depicted in the following sequence diagram. However, details about how the RNA agent is functioning during negotiation process are provided in the section concerning RNA.

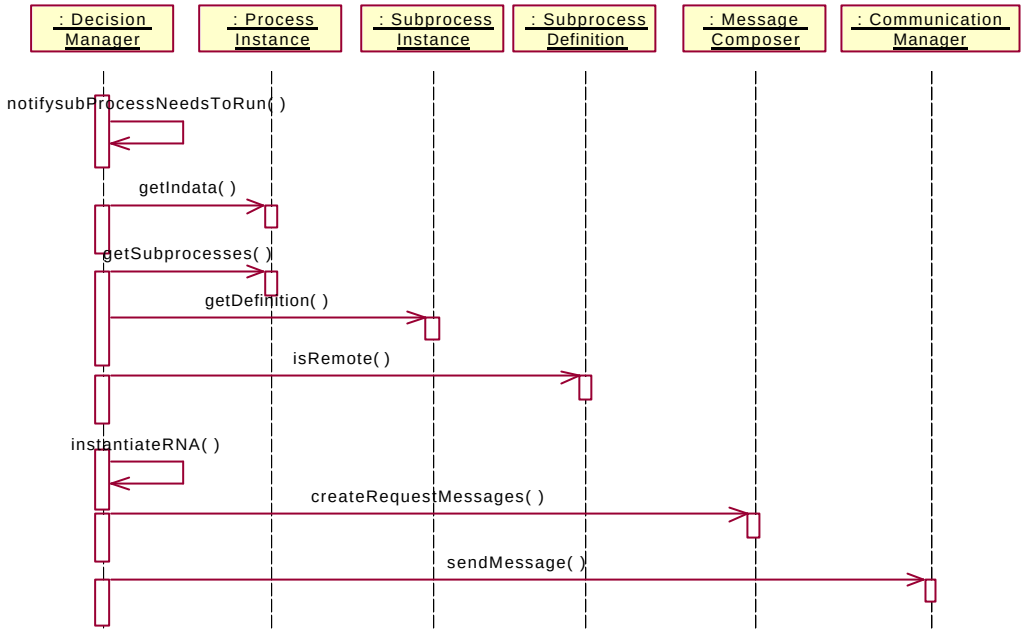

Figure 64: Instantiate Requestor Negotiation Agent

The above specifications of WPA operations and functionality are related to the sub-processes. However, there is a direct one-to-one correspondence with the atomic processes. This means that the WE, Decision manager and the external condition checker are functioning in the same way for the atomic processes as in the case of sub-processes. In the following section more details about the atomic processes and the Resource Provider Agent (RPA) are provided.

8.5 Resource Provider Agent

The Resource Provider Agent (RPA) is responsible for the execution of an atomic process. An atomic process is an elementary unit of process inside a business process. Whenever an atomic process is specified within a business process, the business object name that will be deployed is also given. The RPA can be instantiated only by a WPA agent when an atomic process needs to run. In that case, the WPA instantiates a RPA agent and provides him the name of the business object and the input parameters and values. In the sequel, the RPA locates the business object and invokes the operation provided by the business object. The input parameters of the business object are the input parameters of the RPA while the output parameters of the business object are the output parameters of the RPA agent. This means that the RPA agent has no sub-processes or other atomic processes. Direct consequence to this, is that the RPA has no embedded Workflow Engine and should not control the execution of any other sub-processes. Therefore, the complexity of the RPA is rather smaller in comparison to the WPA.

The RPA agent is a vital part of the business process execution and it can also be in the same potential states as the WPA agent. The states that the RPA agent might be are:

- running: when the RPA agent has been instantiated and the invocation of a business object has started,
- suspended: when the RPA agent got a suspend message from his parent WPA to suspend its

operations,

- resumed or running again: when the RPA agent got a resume message from his parent to resume its operations,
- terminated: when the RPA agent got a terminate message from his parent to terminate its operations,
- aborted: when the RPA agent got a abort message from his parent to abort its operations, or it can not accomplish its mission and generate an aborted message to indicate that event to his parent.
- completed: when the RPA agent has completed its mission and forwarded the output results to the WPA agent.

The RPA agent communicates only with his parent WPA. The request messages that the agent can get from the WPA are to start his execution, to suspend it , to resume it, to abort it, or to terminate it. In any of the above requests, the RPA agent changes its status to a corresponding status and sends back to his parent WPA a response with the new status. However, there are two cases where the RPA might send a response to his parent WPA without prior request. The first case is when the RPA agent has completed its mission and sends back a completed message with the output parameters and values. The second case is when the RPA agent cannot continue his execution and sends back an abort messages. An abort message might be generated when the agent cannot locate the business object, or when the business object has generated an exception.

The communication of RPA agent with his parent WPA is based on message exchanges specified in standard FIPA ACL/XML format while the content of the messages has been specified in the intra-domain ontology.

In addition to the basic modules of the FIPA compliant agent, the following entities are specified for the RPA agent:

- **INDO XML Parser**: responsible for parsing the content of the FIPA ACL messages based on the intra-inter domain ontology,
- **INDO Message composer**: responsible for composing the appropriate response FIPA ACL-XML messages related to the parent WPA. The structure of the messages is based on the interdomain ontology,
- **Decision manager**: responsible for controlling the basic operations of the agent, communicating with the Business Object manager and the other entities
- **Business Object manager**: responsible for locating the business object, invoking its operations and returning back its output results.

In the following picture the internal structure of the WPA agent and the relationships among the basic modules is depicted.

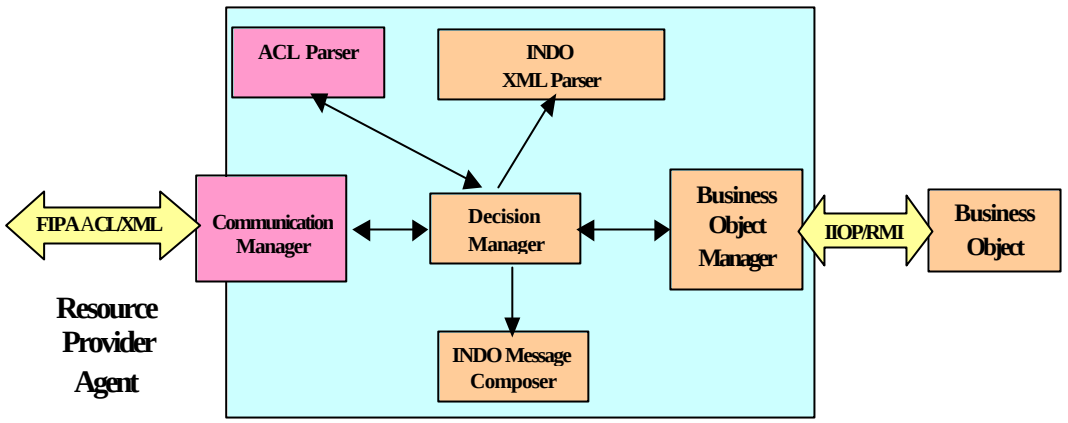

Figure 65: Resource Provider Agent Internal Architecture

Although the DR agent and the WPA are generic agents related to the execution and management of business processes, the RPA agent is always associated with a particular business object, i.e. there are different RPAs within a single domain. In general, every RPA is related to a particular business object and has the appropriate knowledge to deploy it. Additionally, the RPA agent should communicate with the WPA agents and participate in different business process instances. This means that the modules of the RPA agent can be categorized into a generic part, related to the business process execution and management, and a specific one, related to the business process object. The Business Object Manager is the specific part related to the business process, while all the other modules are generic ones. The developer of a RPA agent should only develop the Business Object Manager by extending the functionalities of the generic RPA class operations.

The main operation of the Business Object Manager is the *execute* method. This is actually the business logic of the RPA. When a RPA agent starts, the Decision Manager invokes the *execute* method of the Business Object Manager. This means that the business logic of the agent starts. At the same time, the status of the RPA is changing to running. The decision manager generates a response message and informs its parent WPA that the external task is now running. As soon as the business logic started, the Business Object Manager first gets the input parameters and values from the Decision Manager, locates the corresponding business object, using probably the CORBA or Java naming service, and invokes its operation. The business object might be any type of object located anywhere inside the domain. The location of the object and the invocation of the method by the business object manager is the responsibility of the RPA programmer. However, since the Business Object Manager is a java class, it can invoke either remote CORBA objects, or Java objects by deploying correspondingly the CORBA-IIOP or the RMI protocols. The previously described steps are depicted in the following figure.

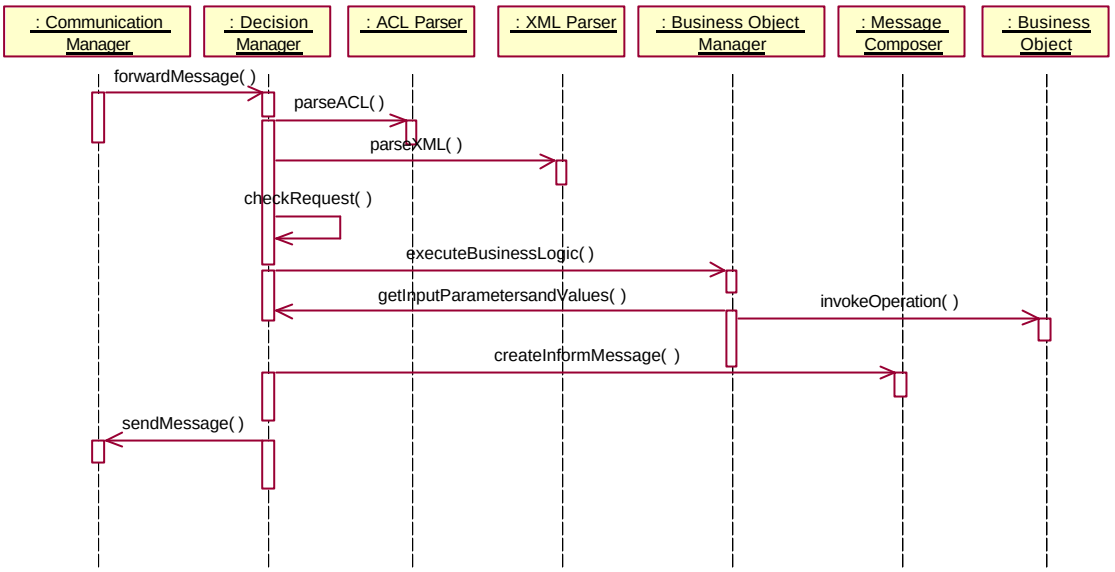

Figure 66: Start RPA Agent

When the business object completes its operation, returns back to the Business Object Manager the output parameters which forwards them to the Decision Manager. This situation interpreted by the Decision Manager as the completion of the business logic. In that case, the Decision Manager composes a completed message and informs its parent WPA. The new status of the RPA agent is now completed. If during the invocation of the business object by the Business Object Manager an exception occurs, then the Business Object Manager informs the Decision Manager about that. In that case the Decision Manager generates an *aborted* status message and sends it to his parent WPA agent while the RPA agent dies out. The previously described steps are explained better in the following sequence diagram.

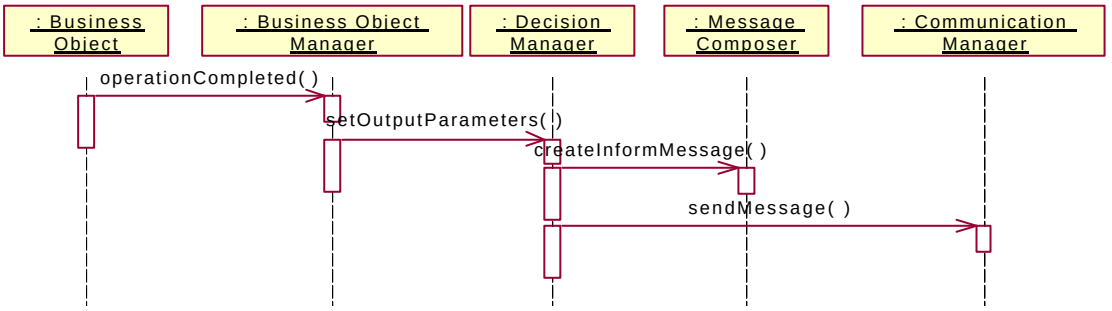

Figure 67: RPA Agent Completes

During the execution of the RPA agent certain requests might arrive from the parent WPA. These requests might be to suspend the execution of the agent, to resume it, to terminate or to abort it. When a request like this arrives in the RPA agent, the Decision Manager parses the message from the ACL and XML parsers and checks the type of the request. In the sequel, the Decision Manager fulfils accordingly the request and informs the parent WPA. If, for example, a

suspend request arrives in the Decision Manager then the Business Object Manager should be informed. One way to accomplish this is to inform the Business Object Manager about this event by invoking an operation provided by the Business Object Manager. Then, it is the responsibility of the RPA agent programmer to handle the event. The other way is to have the Business Object Manager as an extra thread running under the control of the Decision Manager. In this case, the Decision Manager suspends the operation of the thread without notifying the Business Object Manager. In the context of this thesis the second approach has been adopted and implemented. The main reason is that the programmer of the Business Object Manager should not handle any type of business process execution requests. On the contrary, the events should be handled and managed in a transparent to the Business Object Manager way by the Decision Manager. The following sequence diagram shows how the RPA agent modules are cooperating to support a request like this.

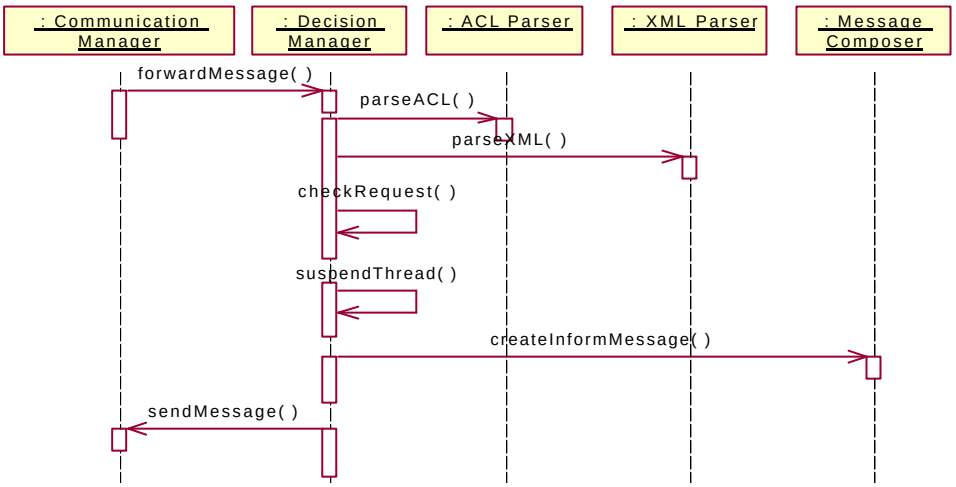

Figure 68: Suspend RPA Agent

In the following section the way the Requestor Negotiation Agent is functioning and the main operations that the agent provides are described and further analysed.

8.6 Requestor Negotiation Agent

The Requestor Negotiation Agent (RNA) is responsible for negotiating with potential VE candidate partners about a particular business process instance. The RNA is instantiated dynamically by a WPA agent when a remote business process needs to be executed. Initially, the WPA provides him the name of the process, the input parameters and values, and requests from him to find a suitable VE partner. Based on this request, the RNA locates from the Offer Repository the negotiation parameters specified for this remote process and migrates into the virtual marketplace. Based on the name of the process, the input, output and negotiation parameters and values, the RNA agent composes a query message that will be sent to the Service Offer Retrieval (SOR) agent. The SOR agent checks the Service Offer Repository, locates all the offers that satisfy the constraints, composes a response message with all the VE candidate partners, and sends it back to the RNA. The RNA interprets the messages and locates the names of the VE candidate domains, i.e. the names of the Provider Negotiation Agents. At this point the negotiation process among the RNA and all the PNAs is starting.

The negotiation process is based on the FIPA compliant Contract-Net protocol. The messages exchanged among the agents are FIPA compliant ACL/XML while the ontology used is the negotiation ontology. The result of the negotiation process is the selection of the best VE partner that can provide the remote process. In technical terms, this agreement is regulated by a particular unique contract. When the negotiation process completes, the RNA informs the parent WPA agent about the VE partner that has been selected and the contract that has been established. At the same time, the PNA stores the contract that has been agreed on into the Contract Repository on the VE candidate partner domain. In the sequel, the WPA composes a request for a remote process execution and sends it to the remote DR of the VE partner by referring to the contract id. The DR agent checks the contract repository, authenticates the WPA with the contract id, and proceeds to the execution of the business process by creating a WPA agent that will execute the requested process. In the following paragraphs, all the steps involved in this complex process are further explained.

In addition to the basic modules of the FIPA compliant agent, the following entities are specified for the Requestor Negotiation Agent (RNA):

- **INDO, VMP and Negotiation XML Parser**: responsible for parsing the content of the FIPA ACL messages based on the intra-inter domain, marketplace and negotiation ontology,
- **INDO, VMP and Negotiation Message composer**: responsible for composing the appropriate response FIPA ACL-XML messages. The structure of the messages is based on different ontologies used,
- **Decision manager**: responsible for controlling the operations of the agent,
- **Offer Repository**: responsible for managing information about all the remote business processes and the input, output and negotiation parameters for each one of them. Specification of the Offer Repository has been provided in the Business Process Specification section,
- **Strategy Manager**: responsible for selecting the best VE candidate partner based on a simple strategy.

In the following Figure 69 the internal structure of the WPA agent and the relationships among the basic modules is depicted.

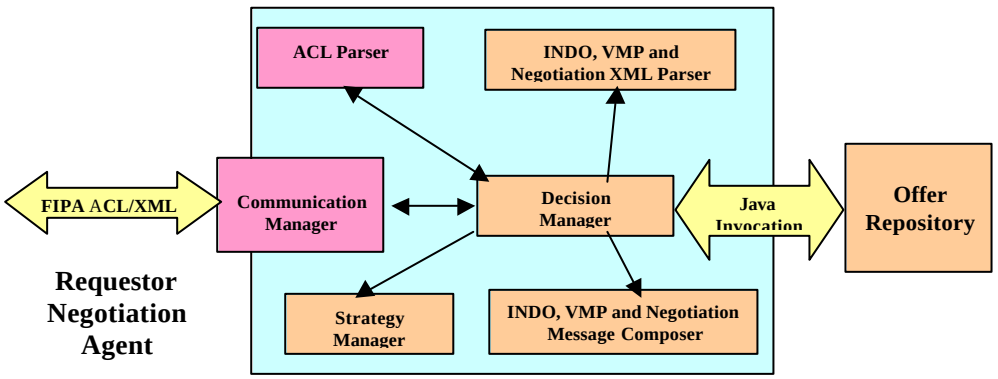

Figure 69: Requestor Negotiation Agent Internal Architecture

Initially, a WPA instantiates an RNA agent and sends to him a request message to find a VE partner for this process name, with the corresponding input parameters and values. The RNA interprets the message and locates the name of the process and the input parameter and values. In the sequel, the RNA conducts the Offer Repository, gets the output and negotiation parameters, migrates to the virtual marketplace, composes a FIPA compliant ACL/XML message based on the virtual marketplace ontology and sends it to the SOR agent. The above mentioned steps are further explained in the following sequence diagram.

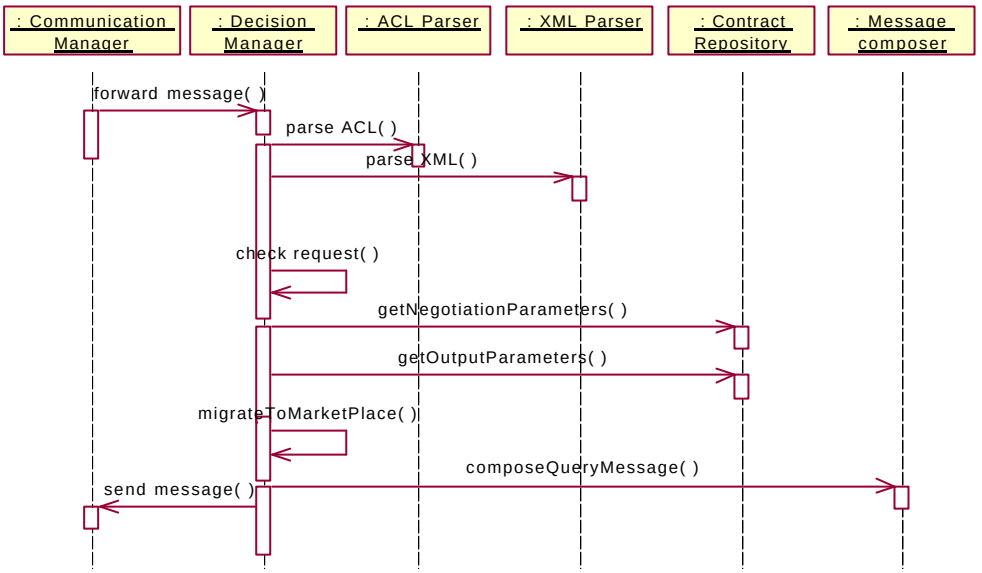

Figure 70: Locate VE Candidate Partners

When the SOR agent gets the query request from the RNA agent, searches the service offer repository, gets the list of VE candidate partners that satisfy the constraints of the query, composes a response ACL/XML message and sends it back to the RNA. The RNA agent parses the message, locates the VE candidate domains and especially the PNA agents, and migrates back to his original domain. At this point of time, the RNA knows all the VE candidate partners and can start the negotiation process with the corresponding PNA agents. In the following picture the steps involved in the selection of the VE candidate partners are depicted.

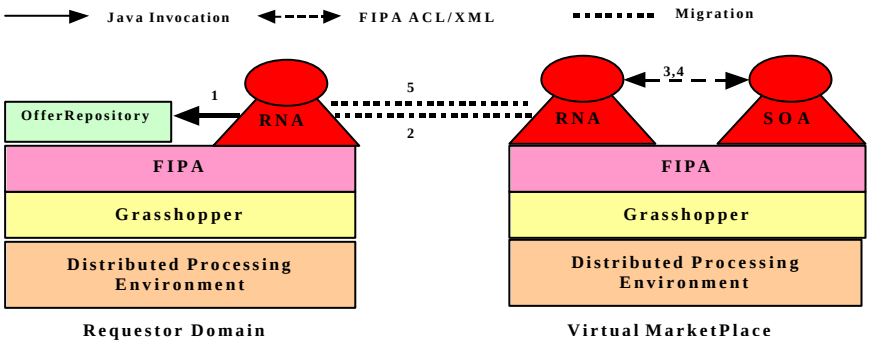

Figure 71: Selection of VE Candidate Partners from the Virtual Marketplace

The negotiation process among the RNA and PNAs agents is based on the FIPA compliant Contract-Net protocol. According to this protocol, two main roles are involved, namely the requestor and a set of providers. In our case, the RNA agent plays the role of the requestor, while the PNA agent plays the role of the provider. The negotiation process starts when the RNA sends a **Call for Proposals (CFP)** to the PNAs requesting from them to make proposals related to the CFP. In our case, the CFP message contains the name of the process and a set of input and negotiation parameters together with values and constraints. For example "Can you provide me the process A with *price*< $20 and *deliveryday*=2" where A is the service name and *price* and *deliveryday* are the negotiation parameters associated with certain constraints. All the PNAs, that can provide process A with such requirements, prepare a **Proposal** message and send it back to the RNA. The Proposal message contains the name of the process and the negotiation parameters with certain values, e.g. "price=$12" and "deliveryday=1". If a PNA cannot provide the process with such requirements, prepares and sends back a **Refuse** message. The meaning of the Refuse message is that this VE candidate provider can not provide the process under these constraints.

All the PNAs agents should respond to the CFP message within a given time period which is called the *negotiation time interval.* Messages arriving later than this period will not be considered. If all PNAs have responded to the CFP or the negotiation time interval has passed, then the RNA evaluates the proposals received and, based on a simple strategy provided by the Strategy Manager, selects the best one. In the sequel, the RNA sends back to the selected PNA an **Accept_Proposal** message with a contract template containing the three parts of a contract, namely the technical session, the administrative session and the pricing session, a contract id and the date of issue. The combination of domain name and contract id is unique and corresponds to only one contract. For all the other PNAs that have not been selected, the RNA sends back a **Reject_Proposal** with a certain reason. When the selected PNA receives an Accept_Proposal message, fills in the contract with its own information regarding the technical, administrative and pricing session, sends it back to the RNA in the form of an **Inform** message, and finally creates a contract and stores it into the contract repository. The requestor domain can now easily invoke the local proposal by only referring to the contract id that has been generated during the negotiation process. In addition to the above key messages sent by the RNA and PNA agent, the following messages might sent during the negotiation process:

- **not-understood-message:** sent by the PNA when the agent can't understand the CFP-message and requires a new one to be sent,
- **failure-message**: sent by the PNA when the values included into the Accept_Proposal are different with the ones that have been agreed upon, i.e. the RNA have changed the values sent with the Proposal,
- **cancel-message**: sent by the RNA when the values included into the Inform message are different with the ones that have been agreed upon i.e. the PNA have changed the values specified within the Accept_Proposal.

Based on the negotiation protocol and the virtual marketplace, VE partners can dynamically locate VE candidate partners, negotiate with them, and establish dynamic VE links. The above described steps and messages are presented in the following figure.

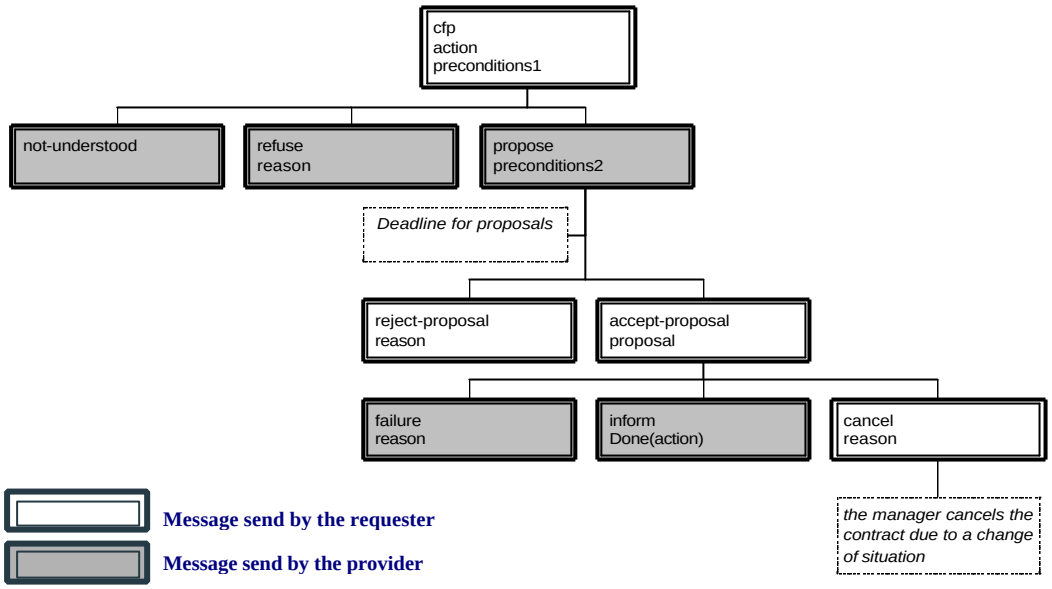

Figure 72: Standard FIPA 97-Contact-net Specification

When the Provider Negotiation Agent (PNA) gets a CFP message from a Requestor Negotiation Agent (RNA), it first parses the message and checks the type of the request. Afterwards, the agent conducts the Offer Repository and checks the offer specified for this local process. If there is no offer specified for this process, the PNA generates a Refuse message and sends it back to the RNA. If there is an offer stored into the Offer Repository, the PNA gets all the stored values of the input, output and negotiation parameters and compares them one by one with the constraints included into the CFP message. If the existing offer does not satisfy the constraints imposed by the CFP, then the PNA agents generates a Refuse message and sends it back to the RNA agent. Otherwise, it generates a Proposal message with the values stored into the Offer Repository and sends it back to the RNA agent. The Proposal message contains the name of the process, the input, output and negotiation parameters and certain corresponding values. It should be noted that both private and public negotiation parameters are included in the Proposal message. The previous described steps followed by the PNA agent, depicted in the following sequence diagram.

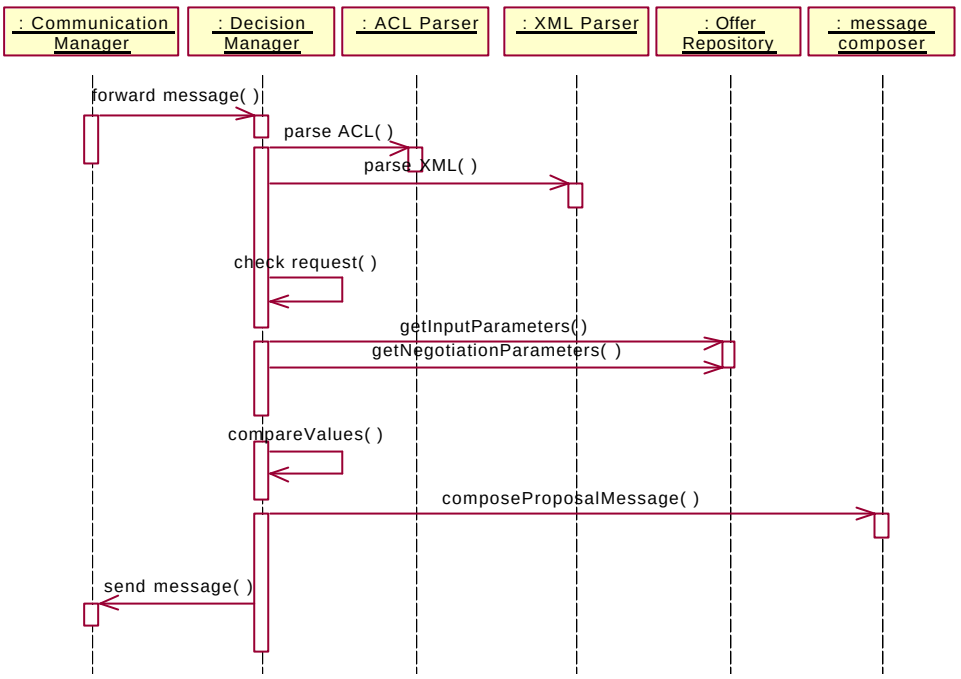

Figure 73: PNA - Create Proposal Message

At the same time, after the RNA agent has sent the CFP messages, it waits for the incoming messages from the VE candidate partners. Two types of messages might arrive, namely a *Proposal* or a *Refuse*. The RNA maintains a List of Active VE candidate Providers (LAVEP). When a Refuse message arrives, the corresponding VE candidate partner is removed from the list of active providers. This means that this domain will not participate in the selection phase. When a Proposal message arrives, the content of the message is stored into the LAVEP. The RNA waits until all the PNAs have sent one message, either Refuse or Proposal, or the negotiation time interval has passed. Then, the Decision Manager notifies the Strategy Manager to evaluate the proposals stored into the LAVEP and to select the best offer.

The List of Active VE candidate Providers (LAVEP) is a dynamic list with all the VE candidate providers that take part in a specific negotiation process. If one VE candidate domain sends a Reject-Proposal or the Proposal message arrives after the negotiation time interval, then this domain is removed from this list. This list actually contains all the proposals send by the different domains. The main operations provided by LAVEP are to insert a VE candidate partner proposal and to delete one. Modification of the lists elements and consequently of the proposals is not allowed for obvious reasons. The class diagram of the LAVEP module is depicted in the following Figure 74.

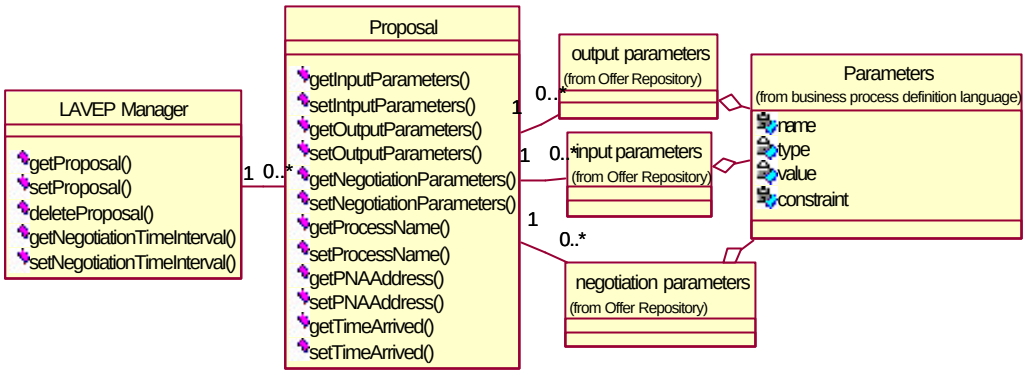

Figure 74: List of Active VE Candidate Providers Class Diagram

The Strategy Manager is a generic component that provides a generic interface for the provision of specialized selection mechanisms for the selection of best proposals and consequently, the best VE candidate partner. Due to the fact that every remote business process under negotiation has probably different negotiation parameters, a specialized selection mechanism is needed for each remote business process. The Strategy Manager Interface (SMI) deploys the different proposals stored into the LAVEP and enables the developer of the customized selection mechanism to develop and integrate its own strategy. In short, the Decision Manager invokes the specialized selection module for this process through the Strategy Manager Interface. The specialized module gets as input the LAVEP list of proposals and then evaluates the different proposals by accessing the input, output, and negotiation parameters and values in order to select one. It is the responsibility of the developer of the specialized selection component to specify and develop its own strategy. The specification and development of a generic strategy mechanism for the selection of VE candidate partners is considered out of the scope of this thesis. However, the interesting reader can get more about automated negotiation strategy algorithms in (see negotiation part).

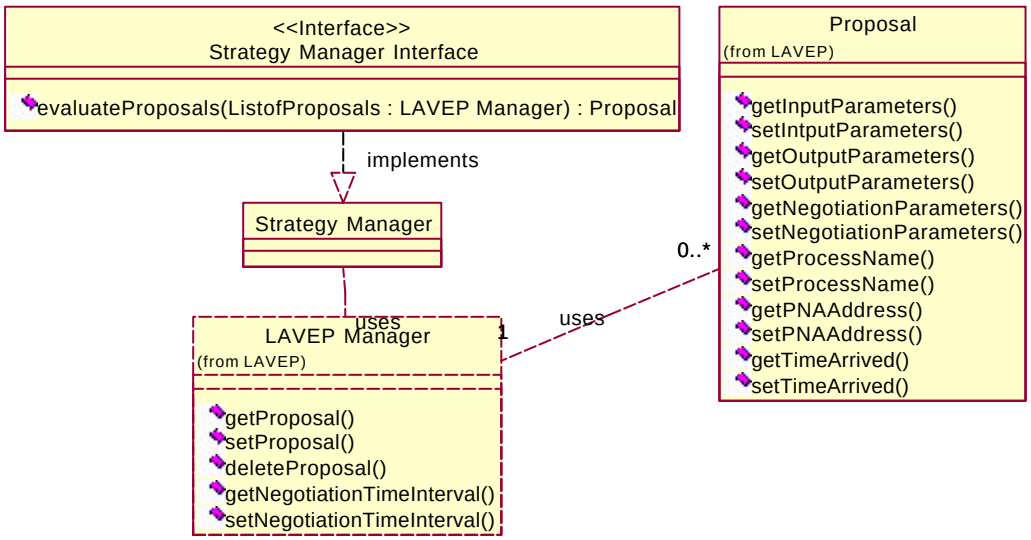

Figure 75: Strategy Manager Interface Class Diagram

In the context of this thesis, only simple selection mechanisms have been developed and tested. More complex selection mechanisms and algorithms can easily be specified, developed, and integrated for experimentation within the Strategy Manager. In the following sequence diagram the previously described steps that the RNA agent follows are shown.

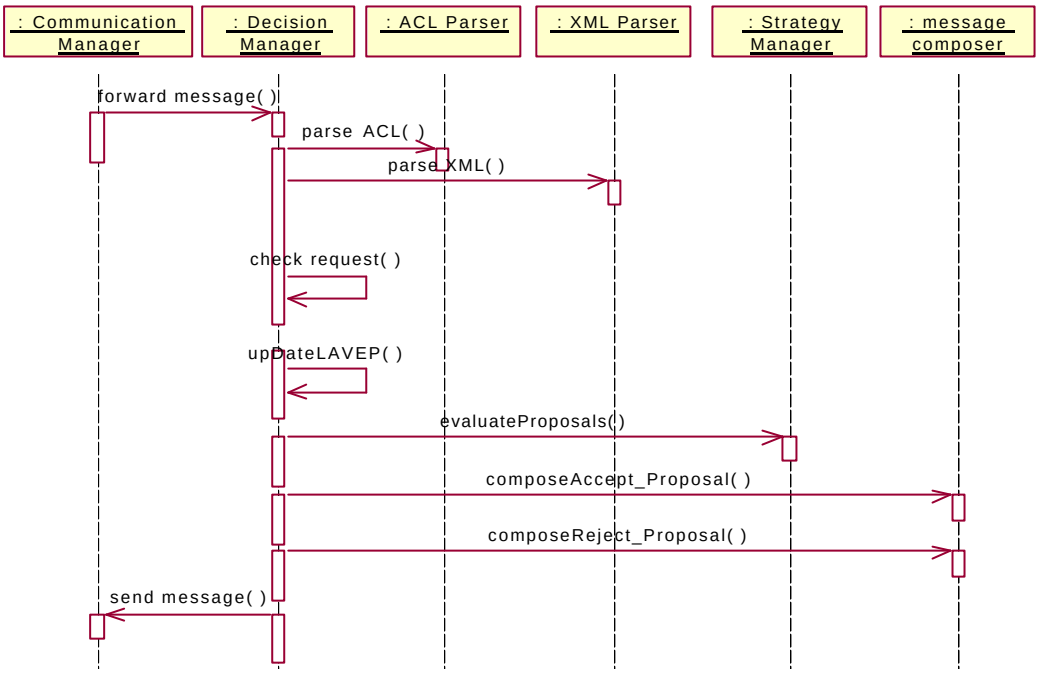

Figure 76: Evaluate Proposal Message

The result of the selection process will be only one Proposal, i.e. one PNA that satisfies the constraints and is the most competitive one in comparison with the others. In that case, the selected PNA will get back an Accept_Proposal message while the remaining ones will get a Reject_Proposal. The Accept_Proposal contains a contract id and a draft contract that should be filled in by the PNA. The draft contract contains all the administration session, technical session, and pricing session information of the RNA domain. The PNA gets the Accept_Proposal message, fills in his own corresponding part, stores it into the Contract Repository, generates an Inform message with the completed contract and sends it back to RNA. At this point of time a contract has been agreed upon and the invocation of the remote process can start. The contract for this domain is characterized as local because this domain will provide the business process to the remote domain. In case that the draft contract contains values different than the ones that have been agreed, the PNA generates a **Failure** message and sends it to the RNA indicating the fact. The previous described steps are depicted in the following sequence diagram.

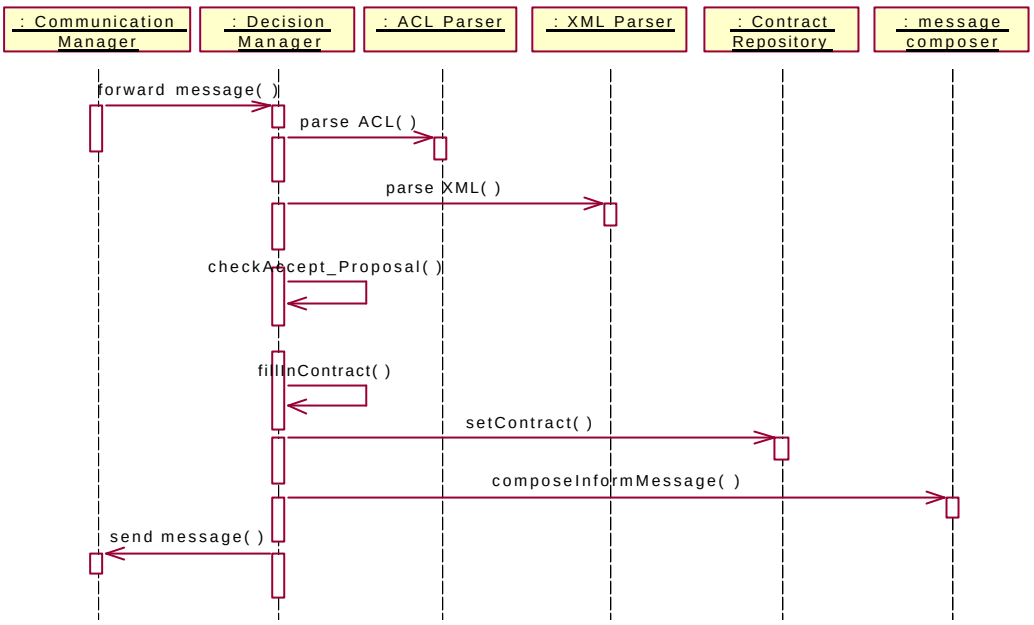

Figure 77: Generate Inform Message

Finally, the RNA agent receives the Inform message from the PNA with the fully completed contract inside. The PNA parses the message, checks the values contained inside, generates an Inform message with the technical characteristics of the VE partner that will provide the remote process, stores the contract into the Contract Repository, sends the message to the initial WPA that generated the negotiation process and dies out. The stored contract is characterized as remote because this domain will deploy the business process specified in the contract. The Inform message contains the name of the remote agent, in our case the DR, the FIPA address of this agent, the ontology used, in our case inter-intra domain, and the protocol used, in our case FIPA Request-Response protocol. This information used from the WPA for the generation of a request message that will be sent to the remote DR agent. The message actually requests from the remote DR agent to start a given process, with given input parameters values and a given contract id. The DR authenticates the request by checking the contract id and starts the execution of the requested process. When the requested process has started, the DR generates an Inform message and sends it back to the WPA to inform him that the execution of the requested process has started. The following sequence diagram describes the steps involved and the activities among the agents.

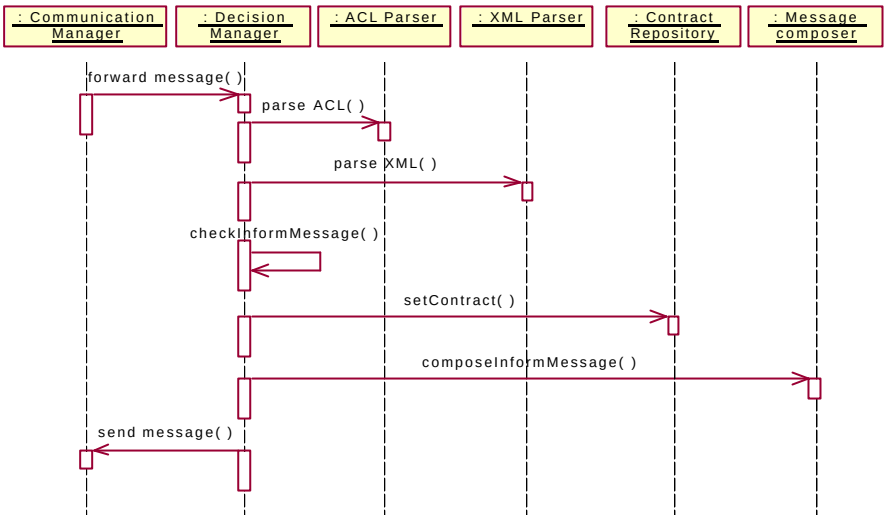

Figure 78: Inform WPA about VE Partner

8.7 Summary

This chapter presents the detailed specification and design of the business process execution and management phases. More specifically, five FIPA compliant agents are introduced and analysed, namely the Personal User Agent (PUA), the Domain Representative (DR), the Workflow Provider Agent (WPA), the Resource Provider Agent (RPA) and the Requestor Negotiation Agent (RNA). The PUA agent is responsible for the provision of the shared business process to the end-user through a web-based interface, while the DR is responsible for initiating and managing the business processes provided by a business domain by conducting the existing contracts. The WPA is responsible for the execution and management of shared business processes in an autonomous, distributed and co-operative way, while the RPA is responsible for the provision of resources and third party business objects and legacy systems during the process execution. Finally, the RNA is responsible for the selection of potential partners and the negotiation among them in order to select the best one. For every agent, the internal architecture, the internal modules, the relationships among them and a set of sequence diagrams are given and discussed. Additionally, for the execution and management of shared business process the inter- and intra-domain ontology are specified and described. Finally, the negotiation protocol used for the automated negotiations and negotiation ontology are further explained and analysed.

Chapter 9: Implementation, Testing, Validation, and Assessment

9.1 Implementation

The agent-based platform for the management of dynamic Virtual Enterprises has been fully implemented and tested. The implementation of the platform has been done following the overall architecture and the specifications and designs provided in the previous chapters. In general, the development of the platform and the different agents has been done using open, interoperable and standard technologies.

The development of the different agents that support the main operations of the platform has been performed in Java programming language. The underlying agent platform deployed was the Grasshopper agent platform with extra OMG-MASIF and FIPA compliant services. To implement a FIPA compliant agent, the agent class has to inherit from the *FIPAAgent* class provided by Grasshopper. In general, the class *FIPAAgent* itself extends from the *StationaryAgent* class, i.e. the basic grasshopper stationary agent. The extension of a stationary, FIPA compliant agent, mainly consists of two methods. These are:

- public void message(FIPAACLMessage msg): this method has to be overwritten by a FIPA agent in order to be able to receive FIPA ACL messages sent by other agents through the ACC.
- public void send(ACLMessage msg): this method enables an agent to send a message to another agent though the ACC. Calling this methods results in establishing a connection to the local ACC by means of grasshopper communication and sending a forwarding request with the message *msg* to the agent addressed in the message.

Apart from the regular Grasshopper agent methods, which have to be implemented for each Grasshopper agent such as the *live* method, the agent developers have to implement or overwrite the *message* method, whereas the *send* method can be simply used.

In principle, FIPA compliant agents should be in position to send and receive ACL/XML messages. For that reason, when a message has been sent to an agent, the agent must first parse the incoming ACL/XML message by deploying a standard ACL parser. For that reason a FIPA ACL Parser is provided by the Grasshopper platform. The parser gets as input an ACL/XML message string, parses it and produces a query object called FIPAACLMessage. In general, the ACLMessage class provides operations for getting and setting the type of message, the sender, receiver, content, etc. As soon as the incoming messages have been parsed from the ACL parser, the content of the message, which is described in XML, should also be parsed. For that reason, three specialised XML parsers have been developed. These XML parsers correspond to the three ontologies, namely the inter-intra domain ontology, the virtual marketplace ontology and the negotiation ontology. The XML parser provides all the necessary operations for interpreting and retrieving information from XML content.

Additionally, when an agent wants to send an ACL/XML message to another agent he should always compose a legitimate ACL/XML message. In general, the responsibility of the message composer is to produce a legitimate ACL/XML string based on the corresponding ontology. For that reason, three message composers have been developed and used corresponding to the inter-intra domain ontology, the virtual marketplace ontology and the negotiation ontology. Using these message composers, the different agents of the platform can easily create legitimate ACL/XML messages following the specified ontologies.

Furthermore, the communication of agents obeys certain FIPA compliant protocols. In the context of this thesis, three FIPA compliant protocols have been specified and developed. These are the FIPA compliant Request-Response, Request-Query and Contract-Net. Additionally, the status of the agents and the internal synchronisation of the provided operations are managed and controlled by the *Decision Manager* module. The *Decision Manager* is a specialised module tailored to the functionality of each agent that manages and controls the requests and responses of the agent with the agents. It is actually the entity that synchronises the internal operations and entities of an agent in order to respond to different requests.

In the following class diagram, the main classes involved in the development of a standard FIPA agent are provided.

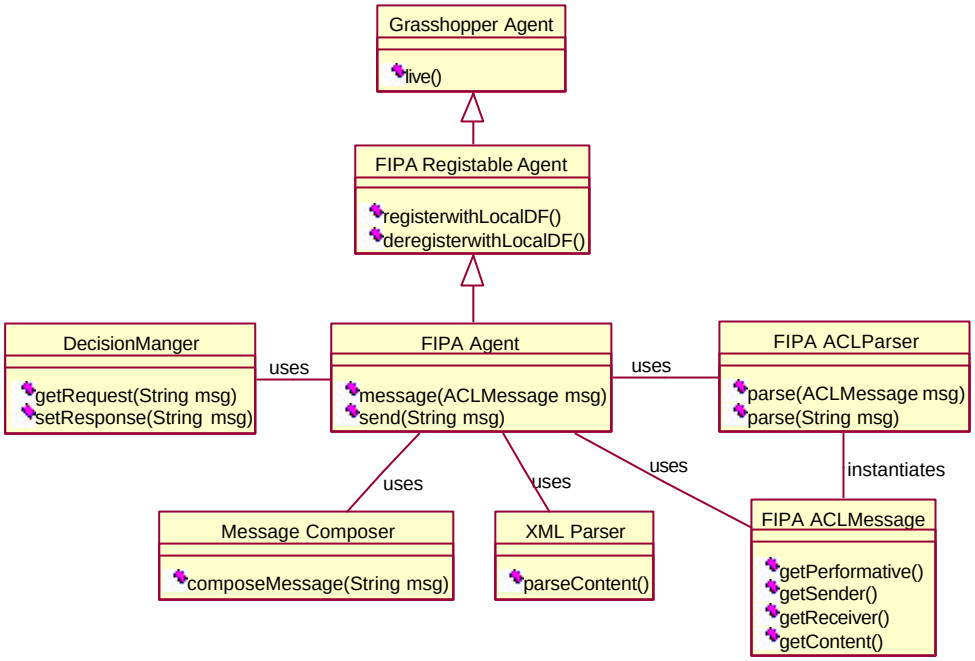

Figure 79: Generic FIPA compliant agent class diagram

Additionally, the virtual marketplace agents deploy and integrate directly a standard OMG-Trader. In that case, the three agents have the appropriate access to the corresponding CORBA objects of the OMG-Trader. More specifically, whenever an agent deploys an object provided by the OMG-Trader, the agent gets a reference to the corresponding object, by using the Interoperable Object Reference (IOR), and, using the CORBA IIOP protocol, access the different methods provided by the object. This interaction is a typical case of using CORBA objects. The OMG-Trader objects deployed by the virtual marketplace agents are the Service Type Repository, the Service Offer Repository and the OMG Constraint Language Parser objects.

Furthermore, for the persistent storage of local and remote business process offers, contracts, and business process specifications, certain XML based storage modules have been developed. The Offer, Contract, and Business Process Repository are XML based persistent modules. In all cases, the different entities of the repositories, i.e. the Offers, Contracts and Business Processes, are stored as separate ASCII files with XML content in conventional file systems. For every persistent storage module a configuration file with references to the individual files of the entities is maintained and configured.

Finally, the execution and management of VE business process by the user is done through the Web. For that reason, special mechanism based on TCP/IP and HTTP protocols and the standard Java Servlets technology have been developed. More specifically, the interface for the management of the business processes is web-based. This means that the user needs only to have a standard Web browser. Every request by the user initiates the corresponding Java servlet at the web server of the VE representative domain. A Java servlet is actually a normal Java object that formulates the appropriate XML request, connects to the TCP/IP server of the PUA agent, and sends the request to the agent. When the request of the user has been fulfilled, the PUA agent,

through the TCP/IP server, informs the Java servlet which in the sequel, informs the user by generating a dynamic html web page.

In Figure 80, a screen shot of the web-based management system provided to the end-user of the VE is provided. This figure depicts the main operations that the user can perform, i.e. to start a process, to suspend, to resume, or to query a process. Additionally, the current status of the process is also depicted.

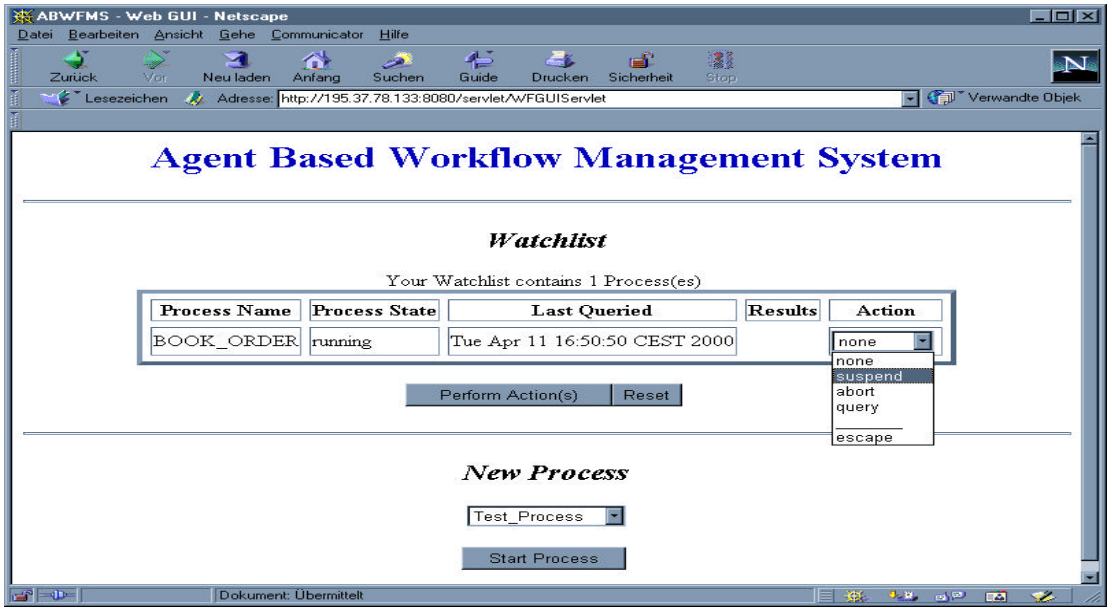

Figure 80: Web-Based End-User Interface

In Figure 81, a screen shot of the business process execution and management system is provided. This figure depicts the main FIPA agents (AMS, DF and ACC) and the agents of the platform, i.e. the Domain Representative (DR), the Workflow Provider Agent (WPA) and the PUA agent.

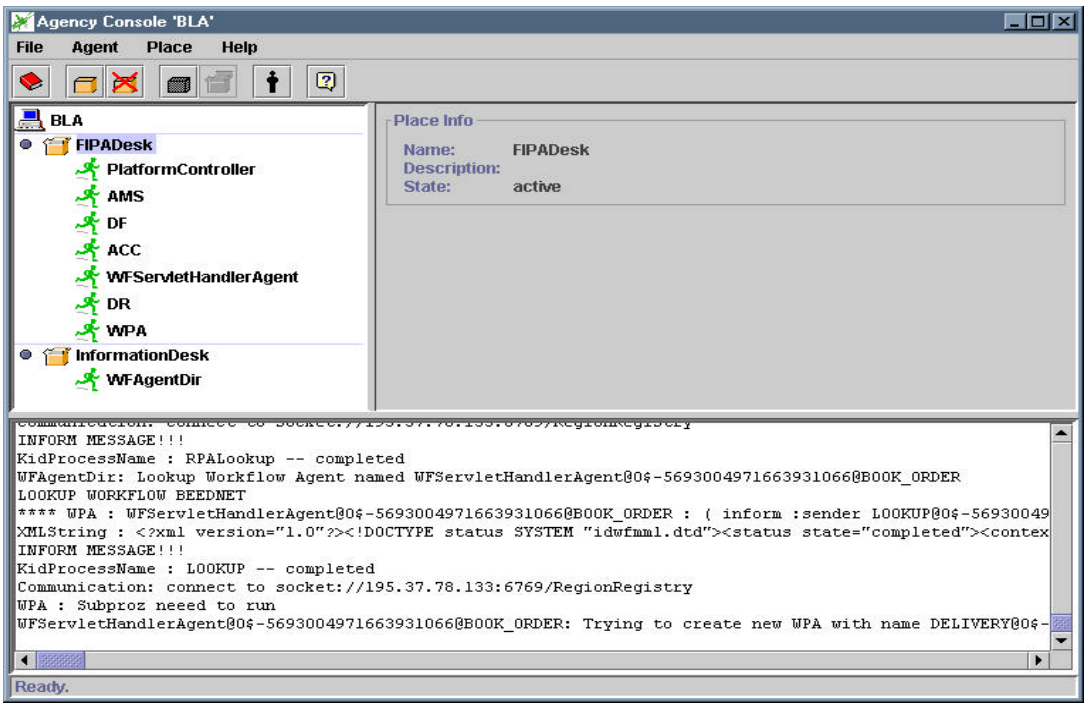

Figure 81: Instantiation of business process

In Figure 82, a screen shot of the business process execution and management system is presented. In that scenario, the end-user has requested to suspend the currently running process. This request has been resulted in the suspension of the corresponding agents. Additionally, the message sent by the PUA agent to the DR agent is depicted. This is actually an *inform* ACL/XML message where the content of the message contains the status of the process, i.e. suspended.

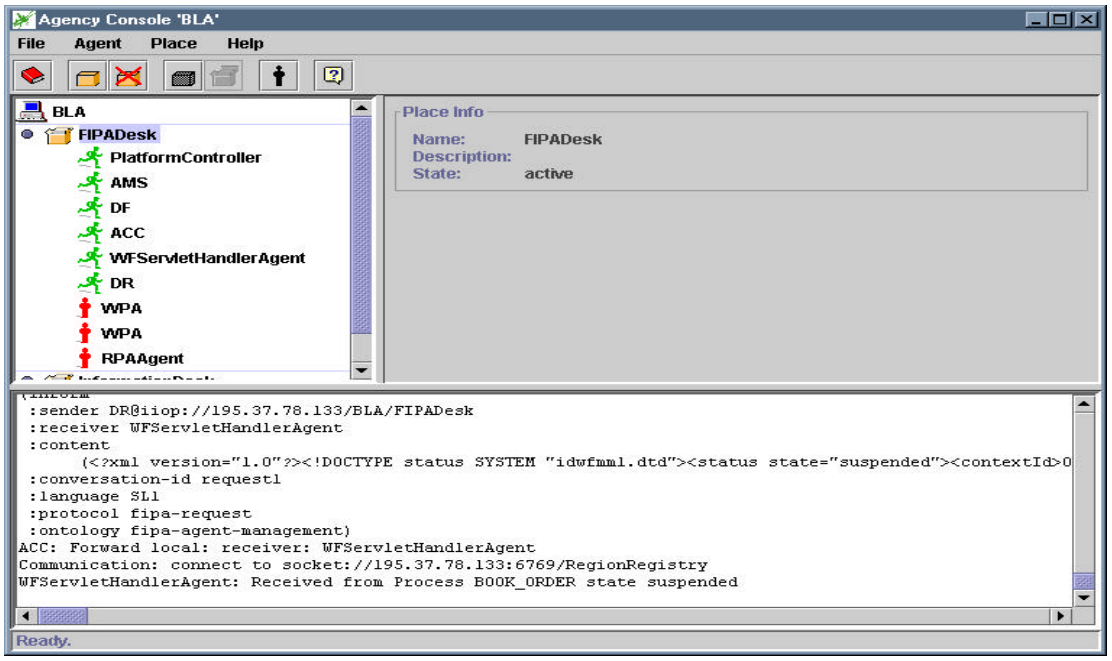

Figure 82: Suspension of business processes

In Figure 83, the Virtual marketplace administration GUI is depicted. This is actually the GUI that the administrator of the Virtual Marketplace uses to administer the service types of the marketplace. In particular, this figure depicts the *add new service type* operation and especially to insert new properties with a pre-defined type, i.e. Boolean, String, etc.

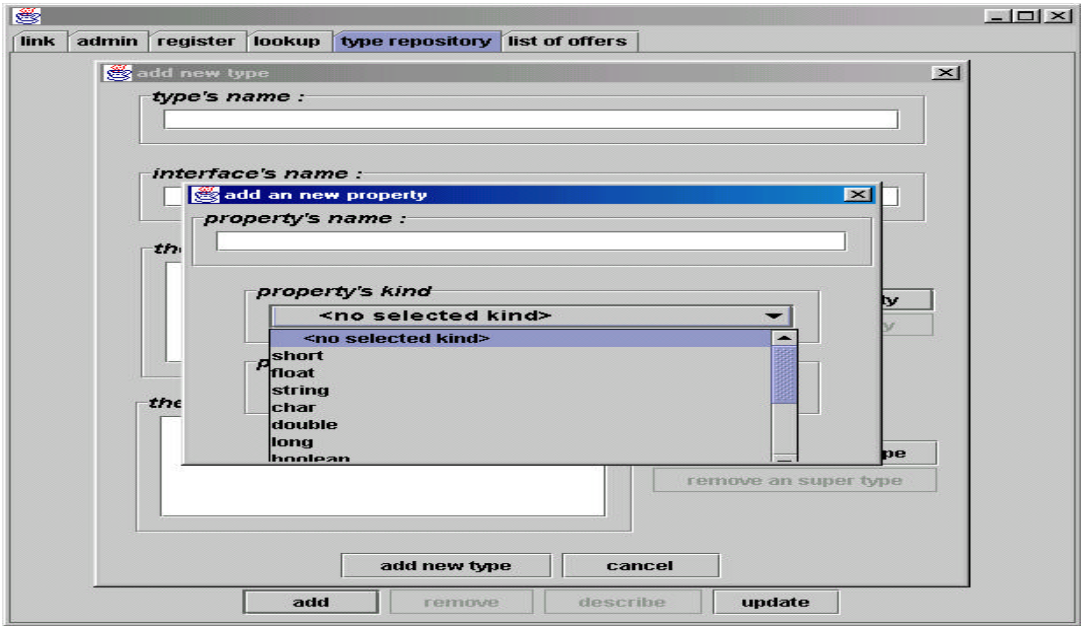

Figure 83: Virtual Marketplace Administration GUI

9.2 Testing

In addition to the development of the different agents and components of the platform, extending testing has also been performed. The testing activities have been conducted by developing certain scenarios that testify certain functionality and features of the system. In every testing scenario, a particular agent, component or interface is being tested. In general, five main testing categories have been specified and developed. These are:

- **testing of the virtual marketplace ontology and agents**: different testing scenarios have been developed for testing the functionality of the virtual marketplace agents and marketplace ontology. These scenarios test actually the different services of the virtual marketplace, i.e. the management of service types, the management of service offers, and the management of retrieval requests. Through the testing scenarios, the specification of the virtual marketplace ontology and the different agents has been improved. In addition to that, the testing scenario improved the integration and deployment of the Service Type Repository and Service Offer Repository. More specifically, effective ways, based on CORBA IIOP, to access objects provided by the OMG-Trader have been identified and improved.
- **testing of the intra-domain business process execution and management**: different scenarios have been developed for testing the intra-domain execution and management of business processes. Emphasis has been placed to the improvement of the intra-domain ontology, the access to Business Process Repository, and the integration of the Workflow Engine and JESS-based Condition Checker. Additionally, the three key interfaces, i.e. the Business Process Listener Interface, the External Condition Checker Interface and the Business Object Manager Interface, have also been tested and improved. Finally, certain synchronisation problems due to the autonomous and distributed execution and management of business processes have been resolved. Synchronisation of intra-domain business processes was a serious problem occurred during the implementation and testing of the system. However, by introducing timestamps and special mechanisms inside the WPA and RPA agents the synchronisation problems have been eliminated.
- **testing of the inter-domain business process execution and management**: different scenarios have been developed for testing the inter-domain execution and management of business processes. Emphasis has been placed in the integration of the negotiation process during the business process execution, the access control and authorisation of remote requests based on contracts, and the overall process management process. More specifically, the two key negotiation agents, i.e. RNA and PNA, have been fully integrated and tested with the WPA agents. Additionally, the inter-domain ontology and Contract and Offer Repositories have been fully tested. The testing scenarios resulted in improvements on the specifications and designs of the ontology, the Offer and Contract Repository and the inter-operation among WPA and RNA agents.
- **testing of the web-based access of user to the business process management**: different scenarios have been also implemented for the testing of the web-based access to business process management. The emphasis was on the interoperation among the Java Servlets and the TCP/IP module located in the PUA, the testing of the Active User Repository (AUR) and the interoperation of the PUA with the Domain Representative (DR) agent. The testing scenarios have improved the specifications and designs of the AUR module and the interoperation interface among the PUA and DR. Additionally, the deployment of Java

Servlets proved to be a very good solution that enabled an efficient interoperation among the web server modules, namely Java Servlets, and agents, namely PUA and DR.

- **testing of the negotiation and contract establishment process**: different scenarios have been also implemented for the testing of the negotiation and contract establishment process. The key entities tested were the negotiation protocol, the negotiation ontology, the Contract Manager and the Strategy Manager Interface (SMI). More specifically, the negotiation protocol and the ontology have been improved due to the fact that certain extreme cases of the protocol have been tested, like the negotiation time interval, the refusal of a proposal, the modification of a contract prior to contract establishment phase from the PNA agent, etc. Through these testing scenarios, the Strategy Manager Interface and the List of Active VE candidate Providers (LAVEP) entities have also been improved.

In general, the testing of the platform and the different entities has significantly improved the initial specifications and designs of the ontologies, the agents, and the internal modules of them.

9.3 Validation

The validation of the platform for the management of dynamic Virtual Enterprises has been done by the development of four independent business and application scenarios. The first two scenarios have been developed in the context of the ACTS-MIAMI project (ACTS 99), the third one in the context of the EURESCOM P815 (P815 99) project and the fourth one individually independent from a particular project.

The first validation scenario is a dynamic VE for the provision of on-line document translation and certification services to remote users. The VE representative is a consulting company that provides document translation and certification services to different users. The translation and certification services are individual processes that can be provided by different VE partners. For that reason, document translation and document certification providers register their service offers in the virtual marketplace. The document translation process is a local process for the corresponding provider and remote process for the VE representative. In similar way, the document certification process is local for the corresponding provider and remote for the VE representative. When a user wants to translate and certify a document, he/she uses the web-based service provided by the VE representative. The user actually specifies the document that will be translated, the initial language that the document has been written, and the target language that the document will be translated. This is actually a request for a business process execution that will be served by the VE representative. When the VE representative gets a request like this, it first checks the input parameters and values provided by the user and starts a negotiation process with potential document translation providers in order to select the best one. When the document translation provider has been selected, the remote document translation process starts. This is actually an inter-domain business process execution. As soon as the document translation is ready, the VE representative starts the negotiation process to locate a document certification provider. This provider will get the translated document and will certify that the translation of the document is correct. This is another inter-domain process execution among the VE representative and the corresponding provider. When the certification process is ready, the translated and certified document will be delivered to the user. This means that the VE business process has been finished and the user can now access the translated document. During the execution of the VE business process, the user can suspend, resume, or even terminate the process.

The second validation scenario is a dynamic network management solution. In that case, a third party network provider called the *Active Virtual Pipe* plays the role of the VE representative and provides network management services to potential corporations. More specifically, the Active Virtual Pipe can establish network connections from one physical location to another with certain Quality of Service (QoS) characteristics. In principle, the Active Virtual Pipe deploys the capabilities of different Connectivity Providers (CP) to establish physical network connections from A to B. When a user requests a network connection, the Active Virtual Pipe conducts the virtual marketplace and negotiates with different Connectivity Providers about the potential connection. When a suitable Connectivity Provider has been selected, the network connection will be established dynamically. During the business process provision, the Active Virtual Pipe monitors the established network connection by querying the status of the process, i.e. the network connection. When a problem occurs, like network fault or performance degradation, i.e. the process aborts, the Active Virtual Pipe conducts the virtual marketplace, negotiates with other potential Connectivity Providers and finally selects another suitable Connectivity Provider. Based on this scenario, the Active Virtual Pipe is the VE representative while the Connectivity Providers are the different VE partners. The VE partners provide physical connections from one physical location to another with some QoS properties. The provision of the network management services to the user from the Active Virtual Pipe is the VE business process while the network connections from A to B locations are local processes of the different Connectivity Providers. The provision of network connections through the deployment of different Connectivity Providers is totally transparent to the user.

The third validation scenario is an International Leased Line service. In a similar way like the previous scenario, a corporation wants to establish an international leased line from one physical location in one country to another physical location in another country. The leased line will be provided to a set of telecom operators that will co-operate in the establishment, configuration and management of an international leased line. This scenario deals not with the actual network connections, like the previous one, but with the management of activities and operations that different departments, teams, and people should do in order to design, configure and establish the requested international leased line. In general, the user conducts a telecom provider and requests an international leased line from one physical location A to another physical location B. This provider is the VE representative while the VE process is the international leased line provision process. The VE representative can provide leased lines only within its country boundaries and for that reasons is looking for other providers that can provide the corresponding segments of the international line until the final destination. However, different telecom providers can provide these network segments of the leased line, and thus a negotiation process starts among the different providers. These telecom providers play the role of the VE partner while the services that they offer are domestic leased lines. For the user, the provision of the international leased line is done by only one domain. However, in reality, different providers are being involved for the provision of the international leased line to the user.

The fourth validation scenario is an on-line book portal system. The portal system sells books on-line to different users all over the world and collaborates dynamically with logistic companies for the delivery of books to the users and with banks for the management of payments. In that respect, the on-line portal system is the VE representative while the logistic and bank partners are VE partners. In principle, the selection of the logistic and bank VE partners is performed dynamically through the virtual marketplace based on the location and the requirements of the user. The delivery of the book to a customer is a local process for a logistic company and remote for the on-line portal system. In a similar way, the payment management is

a local process for a bank company and remote for the on-line portal system. Based on this dynamic selection of logistic and bank domains, different dynamic VEs can be established based on the location and requirements of the user.

The above described validation scenarios have been developed and demonstrated publicly to different events and conference with great success. The diversity in scope and business context of the validation scenarios underline the generality and openness of the proposed approach and the applicability of the it in solving existing, every day, business problems based on dynamic Virtual Enterprises concepts.

9.4 Assessment

Based on the development, testing and validation of the agent-based platform of the management of Dynamic Virtual Enterprises, the assessment phase has been conducted. The assessment phase focused on three individual areas, namely assessment of:

- XML-based workflow management standards,
- emerging FIPA standards,
- proposed solution.

In the following sections, these three individual assessment phases are presented and discussed.

9.4.1 Assessment of Workflow Management Standards

As part of the assessment phase, an analytical comparison of the proposed approach with the current Interoperability Wf-XML Binding standard specified by the Workflow Management Coalition has been performed (WfMC 98-00). The main emphasis of this comparison is on the usage of XML and FIPA ACL for the communication among agents located in different agent platforms. More specifically, this assessment will identify the similarities and differences between the proposed approach and the approach described in the Workflow Management Coalition Workflow Standard announced in January 2000 and labelled WFMC-TC-1023, which is working standard.

The main difference between these two approaches is on the scope and the technology used for the development of the XML-based workflow management systems. The WfMC's standard specifies an interoperable, XML-based interface for the execution and management of business processes by different workflow management systems and products. Therefore, the proposed standard neither address the technology used for the development of the workflow management systems, the workflow engine, and the execution and management mechanisms, nor the underlying transport protocols. The main emphasis is on the syntactic and semantic specification of the XML messages that can be exchanged among the different workflow entities. On the contrary, the proposed solution addresses a complete agent-based workflow management solution using open, emerging, agent standards like FIPA and FIPA-ACL. Additionally, the proposed solution is focusing on dynamic VEs and thus, dynamic selection of VE partners, negotiation and integration of negotiation with workflow management agents during process execution are very important aspects of the platform. In general, the proposed solution deploys a virtual marketplace for the dynamic selection of VE partners, i.e. registration of local business processes to virtual marketplace is a very important feature of the proposed solution. The

WfMC's proposed standard has a narrow view, is focusing on the execution and management of intra-domain business processes, and puts emphasis on the interoperability issue. Therefore, the WfMC proposed standard could be considered as an intra-domain ontology for the execution and management of business processes. For that reason, the assessment will be done on this basis.

In more details, the core of the proposed approach is built around the same logical concepts as the one described in the Wf-XML Binding standard. The *ProcessDefinition* and *ProcessInstance* maps directly to the corresponding *ProcessDefinition* and *ProcessInstance* models specified in the context of this thesis The Observer interface specified in Wf-XML Binding standard corresponds directly to the Business Process Listener Interface (BPLI) that has been specified and developed in the proposed approach.

The overall message structure of the inter-domain ontology of the proposed approach is simpler. The main choice was to use FIPA-ACL as the basic agent communication language, in order to maintain interoperability among other agent platforms and to use XML in the content field of the FIPA-ACL messages, in order to describe any data required for the execution and management of business processes. As a result, the messages exchanged between the platforms of different administrative domains are a combination of ACL messages with XML content inside. On the contrary, WfMC specifies its own envelope in XML for the messages exchanged among the different entities of the workflow management system. More specifically, the *CorrelationData* element is used to connect the request and response messages together in the Wf-XML Binding. This element already exists in the standard FIPA-ACL message and it is called c*onversation-id* field and thus, there was not clear need to define a field like this. However, there was an obvious need to connect agent conversations together, i.e. to define a common context for two or more agent conversations during a business process execution. Therefore, the proposed approach includes a *contextId* element that facilitates this role. Additionally, the message header and the functions of the message header were already provided by the FIPA ACL performatives, so there was no need to include them into the proposed XML inter-domain ontology. Finally, the message body is spread over the FIPA ACL part of the messages and the XML part of the messages. This is due to the fact that the proposed model is based on FIPA compliant agent communication. The option of having standard FIPA-ACL as a communication language among agents gives the benefit of deploying the FIPA compliant communication protocols like the Request-Response, Contract-Net, etc. However, in WfMC proposed standard these protocols should be extensively specified and developed.

FIPA assumes that the communication among agents is asynchronous and loosely coupled. The asynchronous communication is a feature provided by the FIPA platform and especially by the Agent Communication Channel (ACC). Therefore, the inter-domain ontology and the communication of different agents during the execution and management of business processes have been designed for asynchronous and loosely coupled communication. In a similar way, the WfMC also specifies that the communication model for the management of processes should also be asynchronous and loosely coupled. However, the WfMC does not specify any particular transport protocol or mechanism in order to achieve asynchronous and loosely coupled communication. On the contrary, FIPA is based on CORBA as its transport layer and defines, through the ACC agent, a single CORBA interface for all agents that want to comply to the FIPA standard. The interface consists of only one method that allows an agent to receive a single string from another agent. One of the consequences of having CORBA as transport protocol is that any parameter that should be included in a message has to be serialized to a string, and the agent that reads in the message should be able to reassemble the object from the serialized string.

Additionally, the representation of process context and input and output parameters in this thesis have been specified as (name, value, type) triplets. The same approach has been used in version 1.0a of the standard dated June1999. The main reason for this choice was simplicity and flexibility. However, complex input and output parameters can also be specified within process specifications. In that case, during process execution the complex input and output objects are passed as serialized objects in string format. This means that special interpretation mechanisms should be included for the serialisation and deserialisation of the complex objects into the WPA and RPA agents. However, this approach is rather complex and results in loss of generality of the whole concept. Therefore, the complex object structure for input and output parameters specified in the WfMC proposed standard is a better way but results in complex message definitions and descriptions.

Concerning process management operations both the proposed approach and the WfMC standard specify the same operations, i.e. start, suspend, resume, abort, terminate, and complete business process operations. Based on these operations, the potential states of a business process instance are the same in both concepts. However, there are two basic differences between the two approaches. The first one is related to the execution of remote business processes. The WfMC standard does not specify any mechanism for access control and authorisation of domains based on contracts and contracts ids. This means that the current standard has not been designed and specified explicitly for inter-domain process execution. The second one is related to the status of business processes. In the proposed model, querying the current status of the process results only in the provision of the current status of a process and not on any other additional data, as in the case of the WfMC.

9.4.2 Assessment of FIPA Standards

As part of the assessment phase, validation of the FIPA related concepts and standards has been performed. One of the key requirements of this thesis was the deployment of FIPA compliant agents by using the standard FIPA-Agent Communication Language (ACL) and the underlying FIPA platform. Based on the experience gained during the design, implementation and testing of the system, a set of conclusions regarding the maturity and efficiency of FIPA have been drawn.

The FIPA standards evolve every year. Although this approach results in significant improvements of the provided specifications, at the same time introduces a set of interoperability problems among different versions of the standards. For example, the Agent Communication Channel (ACC) specification of FIPA97 is different to the FIAP98 specifications. The main problem is the introduction of the *forward* performative that has been introduced in the FIPA98 series for the ACC-ACC communication. This means that in principle, the ACC specification of FIPA98 is not compatible to the ACC specification of FIPA97.

The Agent Communication Language (ACL) specification is very generic and rather simplified. The currently specified ACL envelope is rather simple and contains a rather limited set of fields. In most of the cases, these fields are not being used, while the semantic meaning of them is rather ubiquitous. At the same time, the execution and management of business processes imposed certain fields to be included into the messages exchanged among agents. These fields need to be included into the content of the message due to the absence of generic fields that could have been used instead. Examples of such fields are the *instance id*, the *contract id*, and the *conversation id*.

Additionally, the standard FIPA protocols are simple and generic. Although this decision has certain benefits, at the same time it is not easy to extend these protocols by specifying specialised performatives. In general, the syntax of ACL envelope is static with pre-defined performatives. The FIPA protocols have been specified based on these performatives. Therefore, when a new protocol needs to be specified, then new performatives need to be defined also. However, a flexible mechanism for defining new performatives does not exist and thus the extension of existing protocols or the definition of new ones based on FIPA-ACL is rather impossible.

Finally, FIPA does not specify the content language that will be used for the description of the content of the message. This option increases the generosity and openness of the approach. However, it introduces certain performance degradations. This is due to the fact that for every incoming message two parsing activities needs to be done, one for the ACL message as such and one for the content of the message. In particular, when the content language has been specified in XML then the deployment of two different syntactic and semantic languages with different formats makes the message format complex and difficult to parse. Instead, a unified message description format would have been better.

In general, the FIPA standards and specifications are rather unstable, generic and in some cases ubiquitous. However, the yearly evolution of the different specifications will definitely improve the specifications and will finally provide a standard, interoperable platform for multi-platform, multi-domain agent systems. Agent standards are very important for the large-scale deployment and acceptance of agent systems and solutions.

9.4.3 Assessment of the proposed solution

Based on the development, testing, and validation of the agent-based platform for the management of dynamic Virtual Enterprises, the following important characteristics and features have been identified:

- **Openness.** This is achieved due to the deployment of the flexible, XML-based ontologies for the management of shared business processes and the negotiation process. Additionally, the specification of generic interfaces like the Business Process Listener Interface, the External Condition Checker, the Strategy Manager Interface, and the Business Object Manager enable the easy integration of third party components and thus contribute to the openess of the system. Moreover, the integration of existing legacy systems, like the OMG-Trader, distributed objects, and JESS rule engine also contribute to the openness of the system. Finally, the usage of open, interoperable, standard technologies like XML, FIPA, FIPA-ACL, and Java also increases the openness of the system.
- **dynamicity, flexibility and evolution.** This is achieved due to the dynamic selection of VE partners and the automated negotiation during business process execution and management. In principle, the proposed approach has been designed and developed with emphasis on flexibility and evolution due to the dynamic Virtual Enterprise concept. The usage of the virtual marketplaces, the registration of local business processes, and the negotiation and dynamic selection of VE partners are special mechanisms that enable and support evolution and flexibility. These concepts in conjunction with the generic communication mechanisms offered by FIPA-ACL increase the levels of flexibility.

- **asynchronous and loosely coupled communication.** This is achieved due to communication mechanisms supported by FIPA platform. In general, the intelligent agents communicate asynchronous and loosely coupled by message exchanges through the FIPA ACC. Asynchronous communication is a key requirement for inter-domain business process execution and management because the different administrative domains should not have static references among them, like in Distributed Component based System. On the contrary, the communication mechanism is based on message exchanges while the content of the message is described in open and flexible ontologies.

- **distribution and scalability**. This is achieved due to the autonomous and distributed execution and management of shared business processes. In principle, the execution and management of business processes is performed by different intelligent, autonomous agents located in different administrative domains. The agents are located in different physical nodes and communicate with message exchanges. Additionally, the interaction between the web server of the VE representative and the PUA agent is also distributed in the sense that the communication is done through the TCP/IP protocol. This means that the web server and the business process management system can be located in different physical locations. The same principle has been adopted for the virtual marketplace. In that case, the three agents of the virtual marketplaces are located in different physical location while the agents that want to deploy their services should migrate to them. Finally, scalability is another feature of the platform. This is achieved due to the autonomous execution of the processes. For every sub-process a specialised agent is created to execute and manage the sub-process. Therefore, as the business process instances running on the system increase, the number of WPA and RPA agents for serving them increase. This concept improves the scalability of the system in the sense that specialised agents are being dynamically created for serving the business processes.

- **autonomy**. This is achieved due to the asynchronous and loosely coupled communication of agents during the execution and management of business processes. Autonomy and decentralisation is a key requirement for the management of dynamic VEs. In the context of this thesis, the agents are autonomous and communicate by exchanging messages specified in FIPA ACL, while the content of the requests and responses is specified by the inter-domain, negotiation and virtual marketplace ontologies. The autonomy of the system is also improved by the deployment of the FIPA compliant protocols.

- **intelligence**. This is achieved due to the deployment of artificial intelligence techniques during the business process execution and management. For that reason, special mechanisms have been developed and tested for the integration of rule engines like the JESS rule engine for the assertion of conditions related to the flow of control in business processes. Additionally, for that reason, two generic interfaces have been specified and developed, namely the External Condition Checker Interface (ECCI) and the Strategy Manager Interface (SMI). The ECCI enables the generic integration of third party condition checkers like JESS while the SMI enables the easy integration of selection and negotiation algorithms during the negotiation process. This means that the intelligence of the different agents can be improved by using the interfaces to incorporate third party intelligent modules.

- **efficient management of network and computational resources**. This is achieved due to the migration of agents to the virtual marketplaces. In general, the migration of agents should be used carefully in order to bring the appropriate results. In general, when a rather

big in terms of size agent is moved from one physical location to another, it may require more bandwidth and network resources than to send a message in ACL/XML format. Therefore, migration of agents is advised only when the communication among the agents is heavy and continuous. In the context of this thesis, migration of agents has been used only for the deployment of virtual marketplace agents. In that case, the PNA and RNA agents migrate to the virtual marketplace in order to register business processes or to select VE partners that can provide a particular business process. In both cases, heavy interaction among the agents occurs and thus the migration of agents is a good choice because it results in efficient management of network and computational resources.

- **easy integration of business objects**. This is achieved due to the specification and deployment of the generic Business Object Manager Interface (BOMI). The BOMI enables the easy and flexible integration of distributed business objects and the creation of specialised Resource Provider Agents (RPA). In general, the developer of the RPA agent should only extend the generic RPA class and implement the methods specified by the BOM Interface. Then, the integration of RPA agent with third party distributed business objects can be easily and effectively performed in a transparent to the developer way. These specialised RPA agents are then used in the different business process specifications.
- **generality and applicability in various applications areas**. This is achieved due to the generality of the different entities of the platform. In principle, the Business Process Definition Language (BPDL) and the service type are generic concepts for describing and specifying processes and process templates. Additionally, the three inter-domain ontologies are generic and can be used in different business sectors or application domains. Finally, deployment of XML as a meta-language for ontology description enables the easy customisation and extension of the different entities. The generality and applicability of the proposed approach is proved by the fact that different validation scenarios from different business sectors and application domains have been developed, tested and demonstrated successfully.

In addition to the previously described benefits, one key drawback has been identified. This drawback is performance and it is related to certain entities of the platform. The main reasons for the performance limitations are:

- **parsing of the messages**. The format of the messages exchanged among agents is specified in FIPA compliant ACL/XML format. Therefore, for every incoming message, parsing of the ACL envelope and parsing of the XML content are required. However, the usage of ACL/XML messages enables the autonomous and loosely coupled communication of agents and thus the performance problem introduced is unavoidable.
- **asynchronous message transportation**. The transportation of messages exchanged among agents is done in an asynchronous way through the FIPA Agent Communication Channel (ACC). Every message sent from one agent to another is forwarded initially to the ACC, which checks whether the destination agent is local or remote to the platform and forwards the message to him. The involvement of the ACC in every agent-agent communication decreases the performance of the system. However, the ACC is a standard entity specified by FIPA standardization committee and the one that guarantees the asynchronous delivery of messages. Therefore, the performance degradation is also unavoidable in that case too.
- **migration of agents**. The migration of agents from one physical location to another also decreases the performance of the platform. The performance problem is introduced when

the migrating agent is rather big enough in terms of bytes. However, when the message exchanges with the remote agent increases, then the migration technique can be profitable. In the context of this thesis, mobility of agents has been used in a reduced way and only when the circumstances require it. For example, mobility is used only with the interaction with the virtual marketplace and especially during the registration of business processes and the selection of VE partners. In both cases, the number of messages exchanged among the agents is high and thus, the mobility of the agent can improve the performance of the platform.

- **agent platform and third party module overhead**. The agent-based platform for the management of dynamic Virtual Enterprises is based on a standard mobile agent platform with FIPA and OMG-MASIF support. All the agent life-cycle management services, mobility services and FIPA compliant services introduce delays and complexity which is inherited into the platform. However, this is also unavoidable due to the fact that the platform is based on emerging agent standards. The same is happening with the JESS engine, the OMG-Trader and the IBM XML parser. However, this is also unavoidable cost due requirement for integration of legacy systems and third party business objects.

9.5 Summary

This chapter presents the implementation, testing, validation and assessment of the proposed approach. More specifically, certain details regarding the implementation of the different agents and components are provided. Main emphasis is given in the generic FIPA compliant agent and the internal modules of this class. The ACL and XML parsers and composers are being explained and discussed and short description of the Decision Manager module incorporated in every agent is provided. Additionally, implementation details concerning the three ontologies, namely the inter-domain, the negotiation, and the virtual marketplace ontology are also provided. Furthermore, description and discussion of the implementation of the Offer, Contract and Business Process Repository are provided. Finally, a short number of screen shots of the implemented platform and the different agents are provided and further discussed. In addition to the implementation details, an extensive analysis of the testing activities are discussed and analysed. The testing activities have been focused on five respective areas, namely the virtual marketplace, the intra-domain process execution and management, the inter-domain process execution and management, the web-based management of processes, and the negotiation and contracting phase. The validation of the platform has been done by developing, testing and demonstrating four different scenarios. The first scenario is related to the ACT-MIAMI project and deals with document translation and certification. The second is also related to the ACT-MIAMI and deals with the dynamic management of network resources. The third scenario is related to the EURESCOM P815 project and deals with international leased lines, while the fourth scenario deals with an on-line book portal system with dynamic selection of logistic and banking partners. After the validation, assessment of the proposed approach is performed. The assessment is done in three respective phases, namely assessment of emerging XML-based workflow standards, assessment of emerging FIPA standards, and finally assessment of the platform and the proposed solution. In both cases, certain key features and characteristics of the proposed solution are discussed and analysed.

Chapter 10: Conclusions

10.1 Conclusions

The penetration of Internet and the web in accordance with new technological advances urged companies to seize the opportunities offered by electronic commerce and to establish a strategic position in the new global networked world. In order to do that, companies should co-operate in different product development phases and share critical business processes, resources, core competencies, skills and know how with each other. In a global, competitive marketplace, companies are continuously seeking for new ways to address competitive pressure. Recognizing the need to shorten development and manufacturing cycles, reduce time to market and operational costs, increase customer satisfaction, operate on global scale and reach, and rapid adaptation to new market changes has historically led companies to automation, collaboration and distribution.

This new business model leaded to the concept of Virtual Enterprises (VE) that is the foundation of the networked economy. The original goals for virtual enterprise business systems were to enable deployment of distributed business processes among different partners, to increase the efficiency of existing provided services, to decrease the cost for the provision of these services, and to adapt rapidly to new market changes. Two broad, well-defined categories have been identified so far, namely the static Virtual Enterprises (SVEs) and the Dynamic Virtual Enterprises (DVEs).

Dynamic Virtual Enterprises improve significantly the Static VEs and take full advantage of the open, global, opportunities offered by the Internet and the global economy. Dynamic Virtual Enterprises feature very short lifetimes, the business relationships among the partners are dynamic and flexible enough for alterations, modifications and evolution. Dynamic Virtual Enterprises exhibit low process integration, high degree of autonomy and scalability between different partners. The number of partners and thus the structure of the network can change dynamically upon demand and supply and based on the requirements of the individual members

of the marketplace. Additionally, Dynamic Virtual Enterprises feature more autonomicity because the business relationships among the partners are dynamic and thus, any changes to the internal business processes can easily be done. Finally, in Dynamic Virtual Enterprises there are no tightly coupled interfaces among the partners and thus scalability and business evolution is a key issue.

Due to the open mechanisms of Internet economy, flexible Dynamic Virtual Enterprises that take advantage of the global market conditions are preferred. Although from business point of view Dynamic Virtual Enterprises are the most promising business model, from technical point of view, the required solutions and systems are more complex, sophisticated and distributed (Ducroux, 1998). Current technologies and scientific results are not addressing in a consistent and coherent way certain key requirements of Dynamic VEs. Open issues like specification and storage of business processes in the context of dynamic VEs, flexible and dynamic mechanisms for autonomous, distributed, and loosely coupled execution and management of business processes across organizational boundaries, registration and management of core business process that can be offered to potential VE partners in an open virtual marketplace, dynamic selection of VE partners based on automated negotiations and business process offerings stored in virtual marketplaces, flexible and adaptable ontologies for business process execution and management across organisational boundaries, flexible and adaptable ontologies for virtual marketplace deployment from both, business process providers and requestors, flexible and easy adaptable ontologies for automated partner selection and negotiation, and finally integration of existing legacy systems and business components with business processes in the context of dynamic VEs.

This thesis analysed, designed, developed, tested, and validated a platform for the management of dynamic virtual enterprises that supports the whole life cycle model, namely the business process specification and registration and business process execution and management. The platform is based on FIPA compliant, autonomous, intelligent agent technologies, emerging agent-based workflow management concepts for cross-organisational business process execution and management, virtual marketplaces with emphasis on OMG Trader integration, and automated negotiation for dynamic partner selection.

More specifically, this thesis defined, developed and validated the following key entities:

- an agent-based, FIPA compliant virtual marketplace that provides service types management, service offer management and service retrieval management based on standard OMG Trader concepts,
- a XML-based virtual marketplace ontology that enables the registration and administration of business process, the management of processes offers, and the dynamic selection partner of partners,
- a negotiation ontology and protocol for the automated and dynamic selection of VE candidate partners based on the FIPA compliant Contract-Net protocol,
- an XML-based business process definition language for the specification of business processes in the context of dynamic Virtual Enterprises and a business process repository for the storage and administration of business processes,
- a distributed, autonomous, agent-based, FIPA compliant, workflow management system for the execution and management of shared business processes across different organizational boundaries based on a flexible and adaptable inter-domain ontology,

- an XML-based intra- and inter-domain ontology for the execution and management of cross-organizational, agent-based, business process execution and management,
- a generic mechanism for the deployment and integration of distributed objects within shared business processes

The agent-based platform for the management of dynamic Virtual Enterprises has been fully implemented and tested. The development of the platform and the different entities of it have been done using open, interoperable and standard technologies. More specifically, the specification and development of the virtual marketplace ontology, the negotiation ontology, and the inter- and intra-domain ontology have been in done in XML. The development of the different agents that support the main operations of the platform has been performed in Java. The underlying agent platform deployed was Grasshopper, an OMG-MASIF and FIPA compliant platform. The communication and interaction among the different agents were based on standard FIPA-ACL while the protocols used were the FIPA compliant Request-Response, Request-Query and Contract-Net. Finally, the platform integrated directly a standard OMG-Trader and developed a special mechanism for the execution and management of business processes through the www by using the TCP/IP and HTTP protocols and the standard Java Servlets technology.

The validation of the agent-based platform for the management of dynamic Virtual Enterprises has been done by the development of four independent business and application scenarios. The first two scenarios have been developed in the context of the EU funded ACTS-MIAMI project, the third one in the context of the EURESCOM P815 project and the fourth one individually. The first scenario is a dynamic VE for the provision of on-line document translation and certification services to remote users. The second scenario is a dynamic network management solution provided by a third party provider called the Active Virtual Pipe. The third scenario is the development of an International Leased Line Scenario through a set of network and telecom providers. Finally, the fourth scenario is an on-line book portal system that collaborates dynamically with a logistic company to delivery the books to the customers and a bank to manage the payment from the customer.

The assessment of the agent-based platform for the management of dynamic Virtual Enterprises has been focused in thee respective areas. An analytical comparison of the proposed approach with the current Interoperability Wf-XML Binding standard specified by the Workflow Management Coalition has been performed. Additionally, as part of the assessment phase, validation of the FIPA related concepts and standards have been performed. One of the key requirements of this thesis was the deployment of FIPA compliant agents by using the standard FIPA-Agent Communication Language (ACL) and the underlying FIPA platform. Based on the experience gained during the design, implementation and testing of the system, a set of conclusions regarding the maturity and efficiency of FIPA have been drawn. Finally, assessment of the proposed model has been performed. Based on the development, testing, validation, the agent-based platform for the management of Dynamic Virtual Enterprises reveals the following important characteristics and attributes:

- **openness** due to the flexible ontologies used for the management of shared business processes, the integration of existing legacy systems, like the OMG-Trader and distributed objects, and the open interfaces used, like the Business Process Listener Interface and External Condition Checker Interface,

- **dynamicity, flexibility and evolution** due to the dynamic selection of partners, the automated negotiation during business process execution and management, and the standard protocols and technologies used,
- **asynchronous and loosely coupled coordination** due to the fact that the communication among the intelligent agents is performed by message exchanges through the FIPA ACC that supports asynchronous and loosely coupled communication,
- **distribution and scalability** due to the distributed execution and management of shared business processes from different intelligent agents across different administrative domains,
- **autonomy and decentralization** due to the loosely coupled coordination of business process execution and management through autonomous intelligent agents and the message exchange approach based on FIPA compliant ACL, protocols and XML-based ontologies,
- **intelligence** due to the deployment of artificial intelligence techniques during business process execution and management, like the JESS engine, and easy integration of specialized selection algorithms through the Strategy Manager Interface,
- **efficient management of network and computational resources** due to the migration of agents to the virtual marketplaces during the business process registration and the VE partners selection phases,
- **generality and applicability in various applications areas and business sectors** due to the generality of the business process definition language, the service types, the inter- and intra-domain ontologies, negotiation ontologies and virtual marketplace ontologies.

In addition to the previously described benefits, one key drawback has been identified. This drawback is performance limitation and it is related with certain entities and features of the platform. The main reasons for the performance limitations are the double parsing of the ACL/XML messages, the asynchronous message transportation through the ACC agent, the migration of agents to different physical locations and the overhead introduced from the agent platform and the third party modules. However, most of the performance deficiencies occurred are strongly related with certain useful features of the platform. Thus, improving the performance of the system results in loss of certain attributes and features of the system. For example, the deployment of both ACL and XML for the description of messages decreases the performance of the system due to the double parsing required in every incoming message. However, at the same time, due to the deployment of the ACL and XML-based ontologies the system reveals high degree of autonomy, openness and loosely couple communication.

Dynamic Virtual Enterprises is an emerging and very attractive business model that enables the dynamic collaboration among different administrative domains based on market oriented approaches and automated negotiation and can constitute the foundation of future electronic business. This thesis tried to provide a systematic, coherent and state of art solution for the management of dynamic Virtual Enterprises based on standard, intelligent agent concepts and technologies. The acceptance and penetration of solutions like this will depend of the success and adoption of the intelligent agent approach and the specification and development of generic business process templates and service types from standardization committees. As the Internet and the new digital economy will urge companies to collaborate and share critical business processes dynamically, solutions like this will become more important, applicable and effective for day to day business operations.

10.2 Future Work

Although the presented work tried to provide a coherent solution for the management of dynamic VEs, certain issues are subject to further improvements and research. These are:

- **negotiation strategy algorithms**. The automated negotiation was one of the key requirements for the selection of VE partners. The thesis provided the basic infrastructure, i.e. protocol, ontology and standard, open interfaces, for the automated negotiation and selection of partners and the integration of negotiation mechanisms with the process execution and management. In principle, different negotiation strategies that can be developed and adopted during the negotiation processes. Since strategy algorithms were not a key requirement, this issue has not been addressed by this thesis. Therefore, a potential improvement would have been the integration and experimentation of different negotiation strategies for the selection of VE partners. The specification of the Strategy Manager Interface enables the easy and flexible integration of negotiation manager modules.
- **fault tolerance and exception handling**. During the execution of business processes certain unpleasant situations might arise. In any case, when the execution of a running business process cannot continue anymore, the process should abort. In the current specification and implementation, unpleasant situations are managed with specific exception handling processes specified during the business process specification. When a process aborts, then the exception handling process starts automatically to bring the system in a stable state. However, this approach solves the problem only in intra-domain process execution but not in inter-domain processes. Therefore, a potential improvement of the platform would have been the provision of fault tolerance and exception handling features for the inter-domain processes.
- **secure inter-domain communication**. In general, the execution and management of inter-domain business processes is performed using the native security features of the underlying platform, i.e. the Grasshopper. The provided solution addressed the problem of access control and authorization to local business processes from remote domains and users. In principle, the Grasshopper security services can be used only for distributed inter-domain Multi-Agent Systems developed in Grasshopper agent platform. However, when different agent platforms are involved, the FIPA-ACL as an interoperability mechanism for agent communication should be used. In that case, the FIPA recommendations do no address explicitly how secure inter-domain communication among agents can be done. Therefore, a potential improvement would have been the introduction of security mechanisms for inter-domain agent communication. However, this is a feature mostly related with the underlying agent platform and not directly with the proposed result.
- **mobility and inter-domain business process execution**. The execution and management of inter-domain business processes is performed in an asynchronous and loosely coupled way by the exchange of messages. The migration of WPA agents from one domain to another has been avoided due to the fact that the performance of the system is worsening. The main reason is that the size of the WPA agents is bigger in comparison to the string-based ACL/XML messages. This means that it takes less time and resources to send a string message than to send a whole agent. As the performance of migration services provided by the mobile agent platforms might be improved in the future, migration of agents to different domains for the coordination and management of business processes would have been an alternative option and thus an issue for further research and investigation.

10.3 Summary

This chapter presents the results achieved in this thesis and identifies a list of issues for further research, experimentation and development.

Chapter 11: Glossary

Agent	An entity that combines one or more service capabilities into a unified and integrated execution model that can include access to external modules, human users, and communication facilities.
Agent Cloning	The process by which an agent creates a copy of itself on a local or remote Agent Platform
Agent Communication Channel (ACC)	Is the agent responsible for routing messages between agents within the platform and to agents resident on other platforms
Agent Communication Language (ACL)	A language with precisely defined syntax, semantics, and pragmatics that is the basis of communication between independently designed and developed software agents.
Agent Creation	The process by which an instance of an agent in being created on an Agent Platform
Agent Domain	is a logical space that provides a context within which Agents may organize and locate each other
Agent Life-Cycle	The finite steps an agent may go through over its entire life history. The agent life cycle consists of the following five states: *Initiated*. The agent is created. *Active*. The agent is currently executing. *Suspended.* The agent has been suspended. *Waiting*. The agent is waiting for some event. *Transit.* The agent is in the process of moving (only applies to mobile agent)
Agent Management System (AMS)	is an agent that manages the creation, deletion, suspension, resumption, authentication, and migration of agents on the agent platform and provides a 'white pages' directory service for all agents resident on an agent platform. It stores the mapping between Globally Unique Agent Names (GUID) and local transport addresses used by the platform. Only one AMS exists in a single Agent Platform.

Agent Migration	The process by which an agent transports itself from one Agent Platform to another.
Agent Name	Uniquely identifies an agent within the network. Stationary and mobile agents communicate mainly through agent names.
Agent Platform (AP)	Provides an infrastructure in which agents can be deployed. An agent must be registered on a platform in order to interact with other agents on that or other platforms.
Atomic Process	Are computational elements, i.e. they provide special elementary operations into the business process. The atomic process has always a well-defined functionality and is always associated with an external business object.
Business Object	a network accessible object that represents a real world business entity or performs a real world business process.
Business Process (BP)	"A procedure where documents, information or tasks are passed between entities of the workflow according to defined sets of rules to achieve, or contribute to, an overall business goal" (WfMC, 1996).
Business Process Analyst (BPA)	The human role responsible for the specification of business processes using a business process definition language.
Business Process Definition Language (BPDL)	Is an XML-based language that enables the specification of complex business processes. The language enables the specification of processes, sub-processes, atomic processes, conditions, input and output parameters
Business Process Execution	Is the process of interpreting and instantiating business processes specified with the help of a business process specification. The execution of the process is performed based on the flow of control and data specified in the specification of it.
Business Process Management	Is the process of managing the status of a business process. The main operations are the suspension, termination, resumption, or completion of a process
Business Process Registration Phase	Is the process of registering business process offers to the virtual marketplace. In that way, different VE partners can find the process providers and select them for dynamic co-operation.
Business Process Repository (BPR)	Is the permanent storage system that stores, administers, and interprets the business process specification.
Business Process Specification	Is a representation of real-world activity in a machine readable format. Conceptually, a business process specification is a directed acyclic graph in which nodes represent steps of execution and edges represent the flow of control and data among the different steps.
Business Process Specification Phase	Is the process that the business process analyst undertakes for the specification of a business process by using the business process definition language.
Business Sub-Process	Is each step within a process. A sub-process has a name, a set of input and output parameters, and start pre-conditions. A sub-process

	consists of one atomic process or one or more sub-processes. A sub-process can be either local, when the current domain can execute the whole sub-process, or remote, when the execution of the sub-process can be performed by another domain.
Communicative Act	A special class of actions that correspond to the basic building blocks of dialogue between agents. A communicative act has a well-defined, declarative meaning independent of the content of any given act. Communicative Act 's are modelled on speech act theory.
Conditions	Specify the circumstances under which certain events will happen. When a condition becomes true then the corresponding sub-process should start its execution. Conditions can be either atomic or composite.
Constraint Language Parser (CLP)	Is the parser that interprets constraints that have been specified using the OMG Constraint Language.
Content Language (CL)	The language used to describe the ontologies and the content of the messages that the agents exchange.
Content of a message	Is part of a communicative act that represents the domain-dependent component of the communication
Contract Repository (CR)	Is the permanent storage system that stores and administers the electronic contracts that have been established as a result of the negotiation process
Conversation	An on-going sequence of communicative acts exchanged between two or more agents relating to an on-going topic of discourse. A conversation may, perhaps implicitly, accumulate context that is used to determine the meaning of later messages in the conversation.
Directory Facilitator (DF)	Provides 'yellow pages' services to other agents. It stores descriptions of the agents and the services they offer. The DF is a mandatory, normative agent that is the trusted, benign custodian of the directory within a single domain.
Domain Representative (DR)	Is the autonomous, intelligent agent that represents one administrative domain and is responsible for the management of the business process requests coming from remote domains or the end-user.
Dynamic VE	Is a dynamic constellation of different administrative domains that co-operate in order to execute and manage share business processes. The form and the relationships among the partners are built dynamically and after negotiation.
Electronic Contract	Is the electronic representation of the agreement reached between two administrative domains for the sharing of a particular business process. It regulates the terms and conditions of the partnership.
FIPA Protocols	Are the set of protocols specified by the FIPA standardization organization in order to achieve standard behaviour and interoperation among different multi-agent systems
FIPA-ACL	Is the agent communication language that has been specified by the FIPA standardization organization. It is based on the speech-act theory

	and is considered the successor of KQML
Flow of Control	Is actually the order in which activities are being scheduled and executed. The Flow of Control is specified by special logical conditions assigned to the sub-processes or atomic processes.
Flow of Data	Is a series of mappings between output data and input data to allow activities to exchange information. Actually, the output parameters of one process can be input parameters of another process.
Home Agent Platform (HAP)	Is the platform that creates and manages the life-cycle of agents or is the initial platform that agents have registered onto.
Inter-Domain Ontology	The set of messages that different autonomous agents, belonging to different administrative domains, exchange during the execution and management of shared business processes.
Intra-Domain Ontology	The set of messages that different autonomous agents, belonging to the same administrative domains, exchange during the execution and management of local business processes.
Life-Cycle Model	The finite steps a system may go through over its entire life history. The different life cycle phases define types of activities which are pertinent during the life cycle of the entity, ISO/DIS 15704 (ISO)
Local Agent Platform	Is the Agent Platform to which an agent is attached. The Local Agent Platform represents an ultimate destination for messages directed to that agent.
Local Business Processes	Are the business processes that can be fully and consistently provided by one administrative domain. The specification of local business processes is stored and managed by one administrative domain.
Message	An individual unit of communication between two or more agents. A message corresponds to a communicative act, in the sense that a message encodes the communicative act for reliable transmission between agents. Communicative acts can be recursively composed.
Message transport service	Is an abstract service provided by the agent management platform to which the agent is currently attached. The message transport service provides reliable delivery of messages to their destination agents, and also a mapping from agent logical names to physical transport addresses.
Mobile Agent	An agent that is not reliant upon the Agent Platform where it was created and can subsequently transport itself between different Agent Platforms.
Mobility	The property or characteristic of an agent that allows it to travel between different Agent Platforms.
Negotiation	Is a process by which a joint decision is made by two or more parties. The parties first verbalize contradictory demands and then move towards agreement by a process of concession making or search for new alternatives (Bicher and Siera, 1997)
Negotiation Ontology	The set of messages, with well-defined syntax and semantic meaning,

	exchanged by the autonomous agents during the negotiation process.
Negotiation Protocol	A common pattern of conversations used by different agents during the negotiation process.
Offer Repository (OR)	Is the persistent storage system that maintains information regarding the registration of local and remote business processes into the virtual marketplace and the negotiation process during partner selection.
OMG Constraint Language (OMG-CL)	Is the standard OMG language for the specification of logical constraints related to the service types and the OMG Trader. It is used for the specification of constraints related to the retrieval of service offers stored into the OMG Trader.
Ontology	Gives meaning to symbols and expressions within a given domain language. In order for a message from one agent to be properly understood by another, the agents must ascribe the same meaning to the constants used in the message. The ontology performs the function of mapping a given constant to some well-understood meaning. For a given domain, the ontology may be an explicit construct or implicitly encoded with the implementation of the agent.
Personal User Agent (PUA)	Is the autonomous agent responsible for the provision of shared business process management and execution operations to the customers of the VE.
Protocol	A common pattern of conversations used to perform some generally useful task. The protocol is often used to facilitate a simplification of the computational machinery needed to support a given dialogue task between two agents. In the context of this thesis, protocol refers to the dialogue patterns between agents, and networking protocol refers to the underlying transport mechanisms.
Provider Negotiation Agent (PNA)	Is the autonomous agent responsible for the registration of local business processes into the virtual marketplace and for controlling the negotiation process on behalf of the provider domain.
Remote Business Processes	Are the business processes that can not be fully provided by one administrative domain. In that case, the administrative domain needs to deploy remote business processes provided by other administrative domains. The specification of remote business processes is stored and managed by other administrative domains.
Requestor Negotiation Agent (RNA)	Is the autonomous agent responsible for the selection of potential VE candidate partners from the virtual marketplace and for controlling the negotiation process on behalf of the requestor domain.
Resource Provider Agent (RPA)	Is the autonomous agent that provides a simple, elementary processing activity into the business process. It has always a well-defined mission and it is related to an external business object.
Role	A well-defined business activity which can not be further subdivided between a number of players.
Service Offer Agent (SOA)	Is the autonomous agent responsible for the management and administration of the service offer requests in the virtual marketplace.

Service Offer Management	Is the process of managing the service offers stored into the virtual marketplaces in relation to certain local business processes of different administrative domains.
Service Offer Repository (SOR)	Is the persistent storage system that stores the different service offers registered in the virtual marketplace.
Service Offer Repository (SOR)	Is the persistent storage system that stores the service offers that have been registered in the virtual marketplace concerning local business processes.
Service Offer Retrieval Agent (SOR Agent)	Is the autonomous agent responsible for the management and administration of the service offer retrieval requests in the virtual marketplace. It is being used for the selection of VE candidate partners.
Service Offer Retrieval Management	Is the process of managing the service offer retrieval requests in the virtual marketplace.
Service Type Agent (STA)	Is the autonomous agent responsible for the management and administration of the service types stored into the virtual marketplace.
Service Type Management	Is the process of managing and administering the service types in the virtual marketplace.
Service Type Repository (STR)	Is the persistent storage system that stores the different service types that have been created in the virtual marketplace.
Speech Act Theory	A theory of communications which is used as the basis for ACL. Speech act theory is derived from the linguistic analysis of human communication. It is based on the idea that with language the speaker not only makes statements, but also performs actions. A speech act can be put in a stylized form that begins 'I hereby request ' or 'I hereby declare ' In this form the verb is called the performative, since saying it makes it so. In speech act theory, communicative acts are decomposed into locutionary, illocutionary, and perlocutionary acts.
Static VE	Is a static constellation of different administrative domains that co-operate in order to execute and manage pre-defined and statically specified business processes. The form and the relationships among the partners are built statically and before the provision of the process to the customer.
Stationary agent	An agent that executes only on the Agent Platform where it was created and is reliant upon it.
VE Business Process	is a business process where the different steps involved are provided by different administrative domains.
VE Candidate Partner	Is the administrative domain that registers its local business processes to the virtual marketplace and is willing to establish dynamic relationships with other domains.
VE Partner	Is the domain that has been engaged itself into a business relationship with another domain through negotiation. The VE partner offers for a very short period of time a specific business process with specific

	terms and conditions to another domain.
VE Representative	Is the administrative domain that represents the VE to the external world and provides the shared business processes to the different customers.
Virtual Enterprises (VE)	A network of different administrative business domains that co-operate by sharing business processes and resources to provide a value-added service to the customer. Each partner of the virtual enterprise will contribute primarily what it regards as its core competencies, i.e. business processes and resources. There is a time limit on the existence of the virtual enterprise caused by fulfilment of its business purpose. From the viewpoint of an external observer, i.e. a customer, the virtual enterprise appears as a unitary enterprise.
Virtual Marketplace (VMP)	Is the set of matchmaking services for the dynamic selection of partners. The matchmaking services consist of the service type management, service offer management, and service retrieval management.
Virtual Marketplace Administrator	Is the human operator of the virtual marketplaces that performs alternatively the service type management operations, like create service type, etc.
Virtual Marketplace Domain	Is the third party administrative domain that provides the matchmaking services to different VE candidate and partner domains.
Virtual Marketplace Ontology	The set of messages, with well-defined syntax and semantic meaning, exchanged by the autonomous agents during the deployment of the virtual marketplace services.
Workflow Engine (WE)	Is the module of the workflow management system responsible for the execution and management of different business process instances.
Workflow Management System (WFMS)	Is the set of tools used to design, define, and specify business processes utilising a business process definition language, widely known as business process modelling tools, the environment or workflow engine in which these processes are executed and managed, widely known as workflow engine, and the set of interfaces to the users and applications involved in the workflow process, widely known as application interfaces and tasklists.
Workflow Provider Agent (WPA)	Is the autonomous agent responsible for the execution and management of a single step within a business process. A set of WPAs and RPAs agent co-operate in an asynchronous and autonomous way to execute business processes.

Chapter 12: Acronyms

ACC	Agent Communication Channel
ACL	Agent Communication Language
AMS	Agent Management System
AP	Agent Platform
API	Application Programming Interface
AUR	Active User Repository
BO	Business Object
BOM	Business Object Manager
BP	Business Process
BPD	Business Process Definition
BPDL	Business Process Definition Language
BPLI	Business Process Listener Interface
BPR	Business Process Repository
C/S	Client/Server
CA	Communicative Act
CFP	Call For Proposals
CL	Content Language
CORBA	Common Object Request Broker Architecture
CR	Contract Repository
DF	Directory Facilitator

DM	Definition Model
DPE	Distributed Processing Environment
DR	Domain Representative
DTD	Document Type Definition
ECCI	External Condition Checker Interface
EDI	Electronic Data Interchange
FIPA	Foundation for Intelligent Physical Agents
GUID	Global Unique Identifier
HAP	Home Agent Platform
HTML	Hyper Text Mark-up Language
HTTP	Hypertext Transmission Protocol
IDL	Interface Definition Language
IDWfMML	Inter-Domain Workflow Message Mark-up Language
IIOP	Internet Inter-Orb Protocol
INDO	Inter- and Intra domain Ontology
IP	Internet Protocol
JESS	Java Expert System Shell
LAP	List of Active Processes
LAVEP	List of Active VE Providers
MASIF	Mobile Agent System Interoperability Facilities
MAT	Mobile Agent Technologies
OMG	Object Management Group
OR	Offer Repository
PNA	Provider Negotiation Agent
PUA	Personal User Agent
RMI	Remote Method Invocation
RNA	Requestor Negotiation Agent
RPA	Resources Provider Agent
RPC	Remote Procedure Call
SL	Semantic Language
SOA	Service Offer Agent
SOR	Service Offer Repository
SORA	Service Offer Retrieval Agent

STA	Service Type Agent
STR	Service Type Repository
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/ Internet Protocol
UML	Unified Modeling Language
VMP	Virtual Market Place
WE	Workflow Engine
WFM	Workflow Management
WfMC	Workflow Management Coalition
WFMS	Workflow Management System
WPA	Workflow Provider Agent
WWW	World Wide Web
XML	Extensible Mark-up Language

Chapter 13: ANNEX

Virtual Marketplace Ontology Specification in XML-DTD Format

```
<!ELEMENT    VMPMessage   (STAMessage | SOAMessage | SORMessage | (VMPException,
YourMsg))>
<!ELEMENT  YourMsg (#PCDATA)      >

<!--    STA    -->
<!-- Request and Respond messages for the STA agent -->

<!ELEMENT STAMessage (STARequest+ | STAResponse)>

<!--    STARequest    -->

<!ELEMENT  STARequest  (AddType  |  DescribeType  |  MaskType  |  ListTypes  |
UnmaskType | RemoveType)>
<!ATTLIST STARequest RequestId ID #REQUIRED>
<!ELEMENT AddType (type, if_name, propdescs, super_types)>
<!ELEMENT DescribeType (type, fully?)>
<!ELEMENT fully EMPTY>
<!ELEMENT MaskType (type)>
<!ELEMENT UnmaskType (type)>
<!ELEMENT ListTypes ((since, incarnation) | (all))>
<!ELEMENT RemoveType (type)>

<!--    STAResponse    -->

<!ELEMENT  STAResponse  (VMPException  |  AddTypeRes  |  DescribeTypeRes  |
MaskTypeRes | ListTypesRes | UnmaskTypeRes | RemoveTypeRes)>
<!ATTLIST STAResponse RequestId ID #REQUIRED>
<!ELEMENT AddTypeRes (incarnation)>
<!ELEMENT  DescribeTypeRes  (if_name,  propdescs,  super_types,  (masked  |
unmasked), incarnation) >
```

```
<!ELEMENT masked EMPTY>
<!ELEMENT unmasked EMPTY>
<!ELEMENT MaskTypeRes EMPTY>
<!ELEMENT UnmaskTypeRes EMPTY>
<!ELEMENT RemoveTypeRes EMPTY>
<!ELEMENT ListTypesRes (types)>

<!--    SOA    -->
<!ELEMENT SOAMessage (SOARequest+ | SOAResponse)>

<!--    SOARequest    -->

<!ELEMENT   SOARequest   (ExportOffer   |   DescribeOffer   |   ModifyOffer   |
WithDrawOffer)>
<!ATTLIST SOARequest RequestId ID #REQUIRED>
<!ELEMENT ExportOffer (type, agent, properties)>
<!ELEMENT DescribeOffer (offerid)>
<!ELEMENT ModifyOffer (offerid, delete?, modify?)>
<!ELEMENT WithDrawOffer (offerid)>

<!--    SOAResponse    -->

<!ELEMENT  SOAResponse  (VMPException  |  ExportOfferRes  |  DescribeOfferRes  |
ModifyOfferRes | WithDrawOfferRes) >
<!ATTLIST SOAResponse RequestId ID #REQUIRED>
<!ELEMENT ExportOfferRes (offerid)>
<!ELEMENT DescribeOfferRes (type, properties, agent)>
<!ELEMENT ModifyOfferRes EMPTY>
<!ELEMENT WithDrawOfferRes EMPTY>

<!--     SOR    -->

<!ELEMENT SORMessage (SORRequest+ | SORResponse)>
<!--    SORRequest    -->
<!ELEMENT SORRequest (Query)>
<!ATTLIST SORRequest RequestId ID #REQUIRED>
<!ELEMENT Query (type, constraint?)>

<!--    SORResponse    -->

<!ELEMENT SORResponse (QueryRes | VMPException)>
<!ATTLIST SORResponse RequestId ID #REQUIRED>
<!ELEMENT QueryRes (offers)>

<!--    VMPException    ->

<!ELEMENT VMPException (reason)>
<!ELEMENT reason (#PCDATA)>


<!-- Specification of Common Entities Used in the DTD VMP FILE -->

<!ELEMENT if_name (#PCDATA)>
<!ELEMENT propdescs (propdesc*)>
<!ELEMENT   propdesc   (prop_name,   (normal   |   readonly   |   mandatory   |
readonly_mandatory),  (string  |  integer  |  float  |  boolean  |  stringseq  |
integerseq | floatseq |  booleanseq) )>
<!ELEMENT prop_name (#PCDATA)>
<!ELEMENT normal EMPTY>
<!ELEMENT readonly EMPTY>
```

```
<!ELEMENT mandatory EMPTY>
<!ELEMENT readonly_mandatory EMPTY>
<!ELEMENT string EMPTY>
<!ELEMENT integer EMPTY>
<!ELEMENT float EMPTY>
<!ELEMENT boolean EMPTY>
<!ELEMENT stringseq EMPTY>
<!ELEMENT integerseq EMPTY>
<!ELEMENT floatseq EMPTY>
<!ELEMENT booleanseq EMPTY>
<!ELEMENT super_type (#PCDATA)>
<!ELEMENT super_types (super_type*)>
<!ELEMENT types (type*)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT since EMPTY>
<!ELEMENT all EMPTY>
<!ELEMENT incarnation (low, high)>
<!ELEMENT low (#PCDATA)>
<!ELEMENT high (#PCDATA)>

<!ELEMENT properties (property*)>
<!ELEMENT property (pname, pvalue)>
<!ELEMENT pvalue (((string | integer | float | boolean ), value) | (stringseq |
integerseq | floatseq | booleanseq), seq_value))>
<!ELEMENT seq_value (value+)>
<!ELEMENT pname (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT offerid (#PCDATA)>
<!ELEMENT delete (prop_name*)>
<!ELEMENT modify (properties)>
<!ELEMENT offers (offer*)>
<!ELEMENT offer (properties, agent)>
<!ELEMENT agent (#PCDATA)>
<!ELEMENT constraint (#PCDATA)>
```

Business Process Definition Language Specification in XML-DTD

```
<!-- Entry point for business process definition in XML -->

<!ELEMENT    business-process-definition    (process-definition    |    condition-
definition | parameter-definition)*>

<!-- Generic class for representing a business process definition -->

<!ELEMENT   process-definition   (process-name,   comment?,   in-data*,   out-data*,
(atomic-process | composite-process))>
<!ELEMENT process-name (#PCDATA)>

<!ELEMENT atomic-process (external-task-name)>
<!ELEMENT external-task-name (#PCDATA)>

<!ELEMENT   composite-process   (composite-process-element+,   exception-handling-
process-element*)>
<!ELEMENT    composite-process-element    (precondition-name?,    sub-process-name,
time-allowed-to-complete?,   (is-remote   |   to-be-negotiated)?,   in-data*,   out-
data*)>
<!ELEMENT   exception-handling-process-element   (exception-source+,   sub-process-
name,   time-allowed-to-complete?,   (is-remote   |   to-be-negotiated)?,   in-data*,
out-data*)>
```

```
<!ELEMENT precondition-name (#PCDATA)>
<!ELEMENT exception-source (#PCDATA)>
<!ELEMENT sub-process-name (#PCDATA)>
<!ELEMENT is-remote EMPTY>
<!ELEMENT to-be-negotiated EMPTY>
<!ATTLIST to-be-negotiated method (directly | vmp) #IMPLIED>
<!ELEMENT in-data (#PCDATA)>
<!ELEMENT out-data (#PCDATA)>
<!ELEMENT time-allowed-to-complete (#PCDATA)>
<!ATTLIST time-allowed-to-complete
              unit (seconds | s | minutes | m | hours | h | days | d) #REQUIRED
              trials CDATA "1">



<!-- Generic class for representing a condition definition -->

<!ELEMENT condition-definition (condition-name, comment?, (atomic-condition |
composite-condition))>

<!ELEMENT condition-name (#PCDATA)>

<!ELEMENT rule (rule-language, rule-body)>
<!ELEMENT rule-language (#PCDATA)>
<!ELEMENT rule-body (#PCDATA)>

<!ELEMENT composite-condition (is-not?, (is-or | is-and), branch-condition-
name+)>
<!ELEMENT is-not EMPTY>
<!ELEMENT is-or EMPTY>
<!ELEMENT is-and EMPTY>
<!ELEMENT branch-condition-name (#PCDATA)>

<!ELEMENT atomic-condition (((is-not?, process-status?) | rule), external-
condition-name)>
<!ELEMENT process-status (process-name)>
<!ATTLIST process-status process-state (running | notStarted | suspended |
aborted | terminated | completed) #REQUIRED>

<!-- Generic class for representing parameter definition -->

<!ELEMENT external-condition-name (#PCDATA)>

<!ELEMENT parameter-definition (parameter-name, comment?, parameter-type,
parameter-value)>
<!ELEMENT parameter-name (#PCDATA)>
<!ELEMENT parameter-type (#PCDATA)>
<!ELEMENT parameter-value (#PCDATA)>

<!ELEMENT comment (#PCDATA)>
```

Inter Domain Ontology Specification in XML-DTD Format

```
(request
       ...
       :protocol fipa-request
       :conversation-id request1
       :content "
              <?xml version=\"1.0\">
```

```
              <!DOCTYPE action SYSTEM \"idwfmml1.dtd\">
              <action command=\"run\">
                     <contextId>SLA-1</contextId>
                     <process>
                            <processid>Build</processid>
                            <paramater>
                                   <parameter-name>serviceName</parameter-name>
                                   <paramater-type>String</parameter-type>
                                   <parameter-value>ILLP</parameter-value>
                            </parameter>
                            <paramater>
                                   <parameter-name>customer</parameter-name>
                                   <paramater-type>String</parameter-type>
                                   <parameter-value>BNT</parameter-value>
                            </parameter>
                            <paramater>
                                   <parameter-name>supplier</parameter-name>
                                   <paramater-type>String</parameter-type>
                                   <parameter-value>NT</parameter-value>
                            </parameter>
                            ...
                     </process>
              </action>"
)
```

"I want you to run the provisioning process for this SLA"

```
(request
       ...
       :protocol fipa-request
       :conversation-id request2
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE action SYSTEM \"idwfmml1.dtd\">
              <action command=\"resume\">
                     <contextId>SLA-1</contextId>
                     <processId>Build</processId>
              </action>"
)
```

"I want you to resume (suspend | abort | terminate) process SLA-1:Build"

```
(not-understood
       ...
       :protocol fipa-request
       :conversation-id request3
)
```

"I don't understand request3"

```
(refuse
       ...
       :protocol fipa-request
```

```
       :conversation-id request2
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE reason SYSTEM \"idwfmml1.dtd\">
              <reason>Don't want to!</reason>"
)
```

"I refuse to perform request2, and here's a reason"

```
 (failure
       ...
       :protocol fipa-request
       :conversation-id request2
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE reason SYSTEM \"idwfmml1.dtd\">
              <reason>Failed miserably</reason>"
)
```

"Although I agreed, I failed to perform request2, and here's a reason"

```
(agree
       ...
       :protocol fipa-request
       :conversation-id request1
)
```

"I agree to perfrom request1"

```
(inform
       ...
       :protocol fipa-request
       :conversation-id request1
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE status SYSTEM \"idwfmml1.dtd\">
              <status state=\"running\"/>"
)
```

"The process for request1 is now running, as requested"

```
(inform
       ...
       :protocol fipa-request
       :conversation-id request1
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE status SYSTEM \"idwfmml1.dtd\">
              <status state=\"suspended\"/>"
)
```

"The process for request1 is now suspended (resumed | aborted | terminated), as requested"

```
(query
       ...
       :protocol fipa-request
       :conversation-id query1
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE status SYSTEM \"idwfmml1.dtd\">
              <status>
                     <contextId>SLA-1</contextId>
                     <processId>Build</processId>
              </status>"
)
```

"I wish to query the status of process SLA-1:Build"

```
(not-understood
       ...
       :protocol fipa-request
       :conversation-id query2
)
```

"I don't understand query2"

```
(refuse
       ...
       :protocol fipa-request
       :conversation-id query1
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE reason SYSTEM \"idwfmml1.dtd\">
              <reason>Don't want to!</reason>"
)
```

"I refuse to perform query1, and here's a reason"

```
 (failure
       ...
       :protocol fipa-request
       :conversation-id query1
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE reason SYSTEM \"idwfmml1.dtd\">
              <reason>Failed miserably</reason>"
)
```

"Although I agreed, I failed to perform query1 , and here's a reason"

```
(inform
       ...
       :protocol fipa-request
       :conversation-id query1
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE status SYSTEM \"idwfmml1.dtd\">
              <status state=\"running\"/>"
)
```

"The process queried in query1 is now running (notStarted | suspended | completed | aborted | terminated)"

```
(inform
       ...
       :protocol fipa-request
       :conversation-id inform1
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE status SYSTEM \"idwfmml1.dtd\">
              <status state=\"running\">
                     <contextId>SLA-1</contextId>
                     <processId>Design</processId>
              </status>"
)
```

"I want you to know that the state of process SLA-1:Design has changed to running (notStarted | suspended | completed | aborted | terminated)"

```
 (inform
       ...
       :protocol fipa-request
       :conversation-id inform1
       :content "
              <?xml version=\"1.0\">
              <!DOCTYPE status SYSTEM \"idwfmml1.dtd\">
              <status state=\"completed\">
                     <contextId>SLA-1</contextId>
                     <process>
                            <processid>Build</processid>
                            <paramater>
                                   <parameter-name>QoS</parameter-name>
                                   <paramater-type>Integer</parameter-type>
                                   <parameter-value>76</parameter-value>
                            </parameter>
                            ...
                     </process>
              </status>"
)
```

"I want you to know that the state of process SLA-1:Build has changed to completed and that the output is 76 for QoS"

Negotiation Ontology Specification in XML Format

cfp-message (send by the requester)

```
(cfp
       :protocol fipa-contract-net
       :conversation-id cfp1
       :reply-by 19991101T120000+0100
       :content "
        <?xml version=\"1.0\"?>
        <!DOCTYPE process SYSTEM \"idwfmml1.dtd\">
              <process>
              <processId>processX</processId>
                     <parameter>
                            <parameter-name>parameterA</parameter-name>
                            <parameter-type>…</parameter-type>
              <parametervalue constraint=\"equal\">
              </parameter-value>
                     </parameter>
                     <parameter>
                            <parameter-name>parameterB</parameter-name>
                            <parameter-type>…</parameter-type>
              <parameter-value>…
                            </parameter-value>
                     </parameter>
                     <parameter>
                            <parameter-name>parameterC</parameter-name>
                            <parameter-type>…</parameter-type>
                            <parameter-value constraint=\"lessequal\">…
                            </parameter-value>
                     </parameter>
</process>
"
)
```

"I want you to make an offer to the business process 'processX' with some parameter-values and constraints in this specific time-out-period. ParameterA and ParameterB should be equal to the specified value, ParameterC should be less than the specified value."

not-understood-message (send by the provider)

When the provider can't understand the cfp-message it sends a not-understood-message back.

```
(not-understood
       :protocol fipa-contract-net
       :conversation-id cfp1
       :
)
```

"I didn't understand your call for proposal"

refuse-message (send by the provider)

```
(refuse
```

```
       :protocol fipa-contract-net
       :conversation-id cfp1
       :content “
        <?xml version=\”1.0\”?>
        <!DOCTYPE reason SYSTEM \”idwfmml1.dtd\”>
        <reason>Don’t want to!</reason>”
)
```

“I refuse to make an offer and here’s a reason”

propose-message (send by the provider)

```
 (propose
       :protocol fipa-contract-net
       :conversation-id cfp1
       :content “
        <?xml version=\”1.0\”?>
        <!DOCTYPE process SYSTEM \”idwfmml1.dtd\”>
       <process>
              <processId>processX</processId>
                     <parameter>
                            <parameter-name>parameterA</parameter-name>
                            <parameter-type>…</parameter-type>
              <parametervalue>…</parameter-value>
                     </parameter>
                     <parameter>
                            <parameter-name>parameterB</parameter-name>
                            <parameter-type>…</parameter-type>
              <parameter-value>…</parameter-value>
                     </parameter>
                     <parameter>
                            <parameter-name>parameterC</parameter-name>
                            <parameter-type>…</parameter-type>
                            <parameter-value>…</parameter-value>
                     </parameter>
                     <parameter>
       </process>
”
)
```

“Here is the offer to the cfp1 for the processX with the following concrete parameter-values …”

reject-proposal-message (send by the requester)

```
 (reject-proposal
       :protocol fipa-contract-net
       :conversation-id cfp1
       :content “
        <?xml version=\”1.0\”?>
        <!DOCTYPE reason SYSTEM \”idwfmml1.dtd\”>
        <reason>Not happy with your proposal</reason>”
)
```

“I reject to make an offer and here’s a reason”

accept-proposal-message (send by the requester)

```
 (accept-proposal
       :protocol fipa-contract-net
       :conversation-id cfp1
)
```

"I accept the proposal …"

failure-message (send by the provider)

```
 (failure
       :protocol fipa-contract-net
       :conversation-id cfp1
       :content "
        <?xml version=\"1.0\"?>
        <!DOCTYPE reason SYSTEM \"idwfmml1.dtd\">
        <reason>…</reason>"
)
```

"Although I proposed to offer a service and you accept on it, I failed to perform the service, and here's a reason"

inform-message (send by the provider)

```
 (inform
       :protocol fipa-contract-net
       :conversation-id cfp1
       :content "
        <?xml version=\"1.0\"?>
        <!DOCTYPE status SYSTEM \"idwfmml1.dtd\">
<status state=\"running\">
<contextId>SLA-5</contextId>
</status>"
)
```

"The service is running, as proposed by me and accepted by you"

cancel-message (send by the requester)

```
 (cancel
       :protocol fipa-contract-net
       :conversation-id cfp1
       :content "
        <?xml version=\"1.0\"?>
        <!DOCTYPE reason SYSTEM \"idwfmml1.dtd\">
        <reason>I've changed my mind</reason>"
)
```

"The service that you proposed and I accepted is cancelled and here's a reason"

Contract Repository Specification in XML-DTD Format

```
<!-- Contract Repository -->
<!ELEMENT     contract     (Contract_ID,     DateOfIssue,     RequestorSection,
SupplierSection)>

<!-- Contract Characteristics  -->
```

```
<!ELEMENT Contract_ID (#PCDATA)>
<!ELEMENT DateOfIssue (day_entity)
<!ELEMENT day_entity (day, month, year)>
<!ELEMENT day (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT year (#PCDATA)>

<!-- Contract Main Entities  -->

<!ELEMENT     RequestorSection     (Technical_Section,     Administrative_Section,
Pricing_Section)>
<!ELEMENT    Supplier    Section    (Technical_Section,    Administrative_Section,
Pricing_Section)>

<!-- Contract Technical section specification  -->

<!ELEMENT Technical_Section (agentName, FIPA_Address, Ontology, Protocol)>
<!ELEMENT agentName (#PCDATA)>
<!ELEMENT FIPA_Address (#PCDATA)>
<!ATTLIST  Technical_Section  Ontology  (Inter-Domain  |  Intra-Domain  )  'Inter-
Domain' #REQUIRED>>
<!ATTLIST  Technical_Section  Protocol  (Request-Response  |  Query-Response  |
Contrat-Net) 'Request-Response' #REQUIRED>>

<!-- Contract Administrative section specification  -->

<!ELEMENT Administrative_Section (DomainName, Address, telephone, fax, e_mail)>
<!ELEMENT DomainName (#PCDATA)>
<!ELEMENT Address (address_entity)>
<!ELEMENT address_entity (street, city, zipcode, country)
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT zipcode (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT telephone (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>

<!-- Contract Pricing section specification  -->

<!ELEMENT Pricing_Section (price, paymentMethod, paymentDateline, BankZIp)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST pricing_Section paymentMethod (visa | eu-card | bank-transfer) 'visa'
#REQUIRED>>
<!ELEMENT paymentDateline (day_entity)>
<!ELEMENT BankZip (#PCDATA)>
```

Business Process Specification Example in XML

```
<?xml version="1.0"?>
<!DOCTYPE business-process-definition SYSTEM "bp.dtd">

<business-process-definition>
<process-definition>
<process-name>BOOK_ORDER</process-name>
<in-data>BOOK_TITLE</in-data>
<in-data>AUTHOR</in-data>
```

```
<in-data>ISBN</in-data>
<in-data>PUBLISHER</in-data>
<in-data>CUSTOMER_NAME</in-data>
<in-data>STREET</in-data>
<in-data>ZIPCODE</in-data>
<in-data>CITY</in-data>
<in-data>EMAIL</in-data>
<in-data>CREDITCARD_NUMBER</in-data>
<in-data>CREDITCARD_TYPE</in-data>
<out-data>AVAILABILITY</out-data>
<out-data>PRICE</out-data>
<out-data>DELIVERY_DATE</out-data>
<out-data>PAYMENT_STATUS</out-data>
<composite-process>
<composite-process-element>

<precondition-name>toStartLookup</precondition-name>

<sub-process-name>LOOKUP</sub-process-name>
<in-data>BOOK_TITLE</in-data>
<in-data>AUTHOR</in-data>
<in-data>ISBN</in-data>
<in-data>PUBLISHER</in-data>
<out-data>AVAILABILITY</out-data>
<out-data>PRICE</out-data>
</composite-process-element>
<composite-process-element>

<precondition-name>ready-to-deliver</precondition-name>

<sub-process-name>DELIVERY</sub-process-name>
<in-data>CUSTOMER_NAME</in-data>
<in-data>STREET</in-data>
<in-data>ZIPCODE</in-data>
<in-data>CITY</in-data>
<in-data>EMAIL</in-data>
<out-data>DELIVERY_DATE</out-data>
</composite-process-element>

<composite-process-element>

<precondition-name>ready-to-pay</precondition-name>

<sub-process-name>PAYMENT</sub-process-name>
<in-data>CREDITCARD_NUMBER</in-data>
<in-data>CREDITCARD_TYPE</in-data>
<in-data>PRICE</in-data>
<out-data>PAYMENT_STATUS</out-data>
</composite-process-element>
</composite-process>
</process-definition>
<process-definition>
<process-name>LOOKUP</process-name>
<in-data>BOOK_TITLE</in-data>
<in-data>AUTHOR</in-data>
<in-data>ISBN</in-data>
<in-data>PUBLISHER</in-data>
<out-data>AVAILABILITY</out-data>
<out-data>PRICE</out-data>
<atomic-process>
```

```
<external-task-name>RPALookup</external-task-name>
</atomic-process>
</process-definition>
<process-definition>
<process-name>DELIVERY</process-name>
<in-data>CUSTOMER_NAME</in-data>
<in-data>STREET</in-data>
<in-data>ZIPCODE</in-data>
<in-data>CITY</in-data>
<in-data>EMAIL</in-data>
<out-data>DELIVERY_DATE</out-data>
<atomic-process>
<external-task-name>RPADelivery</external-task-name>
</atomic-process>
</process-definition>
<process-definition>
<process-name>PAYMENT</process-name>
<in-data>CREDITCARD_NUMBER</in-data>
<in-data>CREDITCARD_TYPE</in-data>
<in-data>PRICE</in-data>
<out-data>PAYMENT_STATUS</out-data>
<atomic-process>
<external-task-name>RPAPayment</external-task-name>
</atomic-process>
</process-definition>

       <condition-definition>
              <condition-name>toStartLookup</condition-name>
              <atomic-condition>
                     <process-status process-state ="notStarted">
                            <process-name>LOOKUP</process-name>
                     </process-status>
                     <external-condition-name>toStartLookup</external-condition-
name>
              </atomic-condition>
       </condition-definition>

       <condition-definition>
              <condition-name>lookup-end</condition-name>
              <atomic-condition>
                     <process-status process-state ="completed">
                            <process-name>LOOKUP</process-name>
                     </process-status>
                     <external-condition-name>lookup-end</external-condition-
name>
              </atomic-condition>
       </condition-definition>

       <condition-definition>
              <condition-name>deliver-not-started</condition-name>
              <atomic-condition>
                     <process-status process-state="notStarted">
                            <process-name>DELIVERY</process-name>
                     </process-status>
                     <external-condition-name>deliver-not-started</external-
condition-name>
              </atomic-condition>
       </condition-definition>

       <condition-definition>
```

```
              <condition-name>ready-to-deliver</condition-name>
              <composite-condition>
                     <is-and/>
                     <branch-condition-name>lookup-end</branch-condition-name>
                     <branch-condition-name>deliver-not-started</branch-
condition-name>
              </composite-condition>
       </condition-definition>

       <condition-definition>
              <condition-name>deliver-end</condition-name>
              <atomic-condition>
                     <process-status process-state ="completed">
                            <process-name>DELIVERY</process-name>
                     </process-status>
                     <external-condition-name>deliver-end</external-condition-
name>
              </atomic-condition>
       </condition-definition>

       <condition-definition>
              <condition-name>payment-not-started</condition-name>
              <atomic-condition>
                     <process-status process-state="notStarted">
                            <process-name>PAYMENT</process-name>
                     </process-status>
                     <external-condition-name>payment-not-started</external-
condition-name>
              </atomic-condition>
       </condition-definition>

       <condition-definition>
              <condition-name>ready-to-pay</condition-name>
              <composite-condition>
                     <is-and/>
                     <branch-condition-name>deliver-end</branch-condition-name>
                     <branch-condition-name>payment-not-started</branch-
condition-name>
              </composite-condition>
       </condition-definition>


<parameter-definition>
<parameter-name>BOOK_TITLE</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>AUTHOR</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>ISBN</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>PUBLISHER</parameter-name>
<parameter-type>STRING</parameter-type>
```

```
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>CUSTOMER_NAME</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>STREET</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>ZIPCODE</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>CITY</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>EMAIL</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>CREDITCARD_NUMBER</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>CREDITCARD_TYPE</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>AVAILABILITY</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>PRICE</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>DELIVERY_DATE</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
<parameter-definition>
<parameter-name>PAYMENT_STATUS</parameter-name>
<parameter-type>STRING</parameter-type>
<parameter-value/>
</parameter-definition>
</business-process-definition>
```

Chapter 14: References

ACTS-MIAMI Project (1999) http://www.fokus.gmd.de/research/cc/ecco/miami/

Adams, T. Dworkin 1997 "Workflow Interoperability Between Businesses". In: Lawerence P. (ed.) WfMC Workflow Handbook. John Wiley & Sons, New York, 1997, pp 211—221.

ADEPT-Next Generation Workflow Management Systems Project (1998)

Alliance Technical Specification (1998) http://www.extricity.com/products/alliance_3zero.html

Alonso, G. Agrawal, D. El Abbadi, A. Mohan, C. Günthör, R. Kamath, M. (1995) "EXOTICA/FMQW: A Persistent Message-based Architecture for Distributed Workflow Management" In IFIP/WG8.1 Working Conference on Information System Development for Decentralised Organisations, Trondheim, Norway, August 1995.

Alonso, G. Fiedler, U. Lazcano, A. Schuldt, H. Schuler, C. Weiler N. (1999) "WISE: An Infrastructure for E-Commerce." Proceedings of the Informatik'99 Workshop "Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications. Paderborn, Germany, October 6, 1999.

Alzaga, A., Martin, J. (1999), A Design Process Model to Support Concurrent Project Development in Networks of SMEs, in Infrastructures for Virtual Enterrises. Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999,pp. 307-318.

Ambrosch,W. D., et.al.: (1989), "The Intelligent Network", A Joint Study by Bell Atlantic, IBM and Siemens, ISBN:0-387-50897-X, Springer-Verlag, Heidelberg

Ambros-Ingerson, J. Steel, S (1988) "Integrating planning, execution and monitoring" In proceedings of the seventh National Conference on Artificial Intelligence (AAAI 88), Mineapolis

Applegate, L M, Holsapple C W, Kalakota R, Radermacher F J and Whinston A B (1996) Electronic Commerce: Building Blocks of New Business Opportunity. Journal of Organizational Computing and Electronic Commerce vol. 6, no. 1. pp. 1-10.

Assiss Silva, F.M., Krause, S., Loyolla, W., Magedanz, T., Mendes, M. (1996)"Agent Skills and their Roles in Mobile Computing and Communications", Mobile Communications - Technology, tools, applications, authentication and security, pp. 181-204, ISBN: 0-412-75580-7, Chapman&Hall, 1996

Banahan, E., Banti, M. (1999), Report on Workshop on Legal Aspects of Virtual Organisations, Brussels, 30 November 1999, http://www.ispo.cec.be/serist/

Barbuceanu, M. Singh N., Syed M. (1995) "COOL: A Language for Describing Coordination in Multi-Agent Systems" Proceedings 1st International Multiagent Systems (ICMAS 95), AAAI/MIT press, Cambridge, Mass., p. 17.24

Bargainfinder homepage (1998), http://bf.cstar.ac.com

Barry, J. Aparicio, M. Durniak, T, et. al. (1998) "NIIP-SMART: An investigation of Distributed Object Approaches to support MES development and deployment in a Virtual Enterprise" 2nd IEEE International Enterprise Distributed Computing Workshop (EDOC 98), Nov. 98

Bäumer, C., Magedanz, T. (1999), "Grasshopper - An Agent Platform for Mobile Agent-based Services in Fixed and Mobile Telecommunications Environments", in "Software Agents for Future Communication Systems", A.L.G. Hayzelden and J. Bigham (Eds), pp. 326-357, ISBN: 3-540-65578-6, Springer Verlag, April 1999

BEA MessageQ (1999) "Technical Specifications" http://edocs.bea.com/tuxedo/msgq/index.htm

Beam, T. Carrie, W. Segev J. (1996) "Electronic Catalogs and Negotiations" CITM Working paper 96-WP-1016, available at http://haas.berkeley.edu/~citm/wp-1016-summary.html

Bellifernine, F. Rimassa G. Poggi A. (1999) "JADE: A FIPA compliant Agent Framework", Proceedings fourth international conference and exhibition on the Practical Applications of Intelligent Agents and Multi-Agent systems (PAAM 99), London

Berners-Lee T, Cailliau R, Luotonen A, Nielsen H F and Secret A (1994) "The World-Wide Web" Communications of the ACM vol. 37, no. 8. pp. 76-82.

Bichler, M. Kumar, M., Jhingran, A. (1998) „Multi-Attribute competive Bidding", IBM T.J. Watson Reserach Report, August 98

Billington C (1994) "Strategic Supply Chain Management" OR/MS Today, April 1994. pp. 20-27.

BizTalk Framework, http://www.biztalk.org/

Block, M. Pigneur, Y. (1995) "The extended enterprise: a descriptive framework, some enabling technologies and case studies in Lotus Notes environment" Proceedings of the 2nd International Conference on Network Organisations Management, June 95

Bolcer, G. A. Kaiser, G. (1999) "SWAP: Leveraging the Web to Manage Workflow" IEEE Internet Computing Jan/Feb 1999

Borenstein, N.S. (1992) "Computational Mail as Network Infrastructure for Computer-Supported Cooperative Work", CSCW´92 Proceedings, Toronto

Borenstein, N.S. (1994) "EMail with a Mind of its own: The Safe-Tcl Language for Enabled Mail", ULPAA, Barcelona

Borghoff UM, Bottoni P, Mussio P, Pareschi R. (1997) "Reflective Agents for Adaptive Workflows". In: Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent

Botspot Homepage (1998) http://www.botspot.com

Bradshaw, J. M., (Eds) (1997) 'Software Agents' MIT Press, ISBN 0-262-52234-9.

Breugst, M., Hagen, L., Magedanz; T. (1998a) "Impacts of Mobile Agent Technology on Mobile Communication System Evolution", pp.56-69, IEEE Personal Communications Magazine, Vol. 5, No. 4

Breugst, M., Magedanz, T. (1998b), "On the Usage of Standard Mobile Agent Platforms in Telecommunication Environments", pp. 275-286, in: Lecture Notes of Computer Sciences 1430, Intelligence in Services and Networks: Technologies for Ubiquiteous Telecom Services, S. Trigila et al. (Eds.), ISBN: 3-540-64598-5, Springer Verlag 1998

Breugst, M., Magedanz, T. (1998c), "Mobile Agents - Enabling Technology for Active Intelligent Networks", IEEE Network Magazine, pp. 53-60, Vol. 12, No. 3, Special Issue on Active and Programmable Networks, May/June 1998

Breugst, M., Magedanz, T., (1998d) "Mobile Agents - Enabling Technology for Active Intelligent Networks", IEEE Network Magazine, pp. 53-60, Vol. 12, No. 3

Breugst, M., Magedanz, T., (1998e) GRASSHOPPER - A Mobile Agent Platform for IN Environments", pp. 279-290, ISBN: 0-7803-4905-9, IEEE Catalog No.: 98TH8364, IEEE IN Workshop 1998, Bordeaux, France, May 10-12, 1998

Cai, T. Gloor, P. Nog, S. (1996) "DartFlow: A Workflow Management System on the Web using Transportable Agents" Technical Report PCS TR 96-283. Department of Computer Science, Dartmouth College, USA

Camarinha-Matos, L. M., Afsarmanesh, H., (eds) (1999a), Infrastructures for Virtual Enterrises. Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999, Boston: Kluwer Academic Publishers

Camarinha-Matos, L.M., Afsarmanesh, H. (1999b), "The PRODNET Demonstrator" in Infrastructures for Virtual Enterrises. Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999, Boston: pp. 279-290

Camarinha-Matos, L.M., Afsarmanesh, H. (1999c), "The Virtual Enterprise Concept" in Infrastructures for Virtual Enterrises. Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999, Boston: pp. 3-30.

Carr J (1996) Intranets Deliver. InfoWorld vol. 18, no. 8. pp. 61-63.

CBL Specifications, http://www.commerceone.com/xml/cbl/docs/components.html

Ceri, S. Grefen, P. Sánchez, G. (1996) "WIDE - A Distributed Architecture for Workflow Management" RIDE 97, UK

Chabernaud, C., Vilain, B. (1990), "Telecommunication Services and Distributed Applications", IEEE Network Magazine, November 1990

Chess, D. (1995) "Itinerant Agents for Mobile Computing", IEEE Personal Communications Magazine, Vol.2, No. 5, pp. 34-59

Chess, D., Harrison, C.G., Kershenbaum, A. (1998) „Mobile agents: Are they a good idea?", in G. Vigna (ED.), Mobile Agents and Security, LNCS 1419, pages 25{47. Springer Verlag, 1998.

Choy, S., Breugst, M.,Magedanz, T. (1999), "Beyond Mobile Agents with CORBA - Towards Mobile CORBA Objects", 6th ACTS Conference on Intelligence in Services and Networks (IS&N), pp. 168-180, H. Zuidweg et.al (Eds.), IS&N 99, LNCS 1597, ISBN: 3-540-65895-5, Springer-Verlag, 1999

Christopher M (1993) Logistics and Supply Chain Management. Pitman Publishing: London.

Chung, P.E. Huang, Y. Yajnik, S. Liang, D. Shih, J.C. Wang, C.-Y. and Wang, Y.-M. (1999) "DCOM and CORBA Side by Side, Step by Step, and Layer by Layer."www.bell-labs.com/~emerald/dcom_corba/Paper.html.

Ciacarini, P. (1998) "Coordinating Multiagent Applications on the WWW: A Reference Architecture" IEEE Transactions in Software Engineering (special issue on mobility and network aware computing) Vol 24, No 3, 1998, pp.362-366

Ciara Byrne (1999) "FACTS and Fiction: Agents in the Electronic Travel Market", Communicate, Vol 4, issue 2,

CIMOSA Association (1998) "Enterprise Engineering and Association: Why and How" CIMOSA e.V.

CommerceNet homepage (1998) http://www.commerce.net/

Communications of the ACM Journal (1994) "Intelligent Agents", Vol.37, No.7, July 1994

Control and Coordination of Complex Distributed Services Project (1999-2001) http://www.inria.fr/Themes/

Cost, R. (1998) "Jackal: A Java-based Tool for Agent Development" Working Notes of the Workshop on Tools for Developing Agents, WS-98-10, AAAI TR 73-82.

Crossflow Project (1998) "ESPRIT Project 28635 Cross-Organisational Workflow Management", The Crossflow Consortium, http://www.crossflow.org

CXML specifications, http://www.cxml.org/home/

Davis T (1993) Effective Supply Chain Management. Sloan Management Review, Summer 1993. pp. 35-73.

Debenham (1998) "An Experimental Agent-based Workflow System". Proceedings of the Third International Conference on the Practical Application of Agent Technology (PAAM98), London, UK, 1998. The Practical Application Company Ltd, pp 101-109.

DOM Specifications, http://www.w3c.org/DOM/

Doz, Y.L., Hamel, G. (1998), "Alliance Advantage. The Art of Creating Value through Partnering", Boston: Harvard Business School Press.

Durfee, E. (1995) "Blissful ignorance: Knowing just enough to coordinate well". In proceedings of the first International Conference on Multiagent systems (ICMAS 95), pp. 406-413 California USA

Ebay Homepage (1998) http://www.ebay.com

Eder, J. Liebhart W. (1996) "Workflow Recovery" In Proceedigns of the 5th IFCIS International Conference on Cooperative Information Systems (CoopIS 96), IEEE Computer Society Press, Brussels, Belgium 1996

Eder, J. Liebhart, W. (1995) "The Workflow Activity Model WAMO" In proceedings of 3rd International Conference on Cooperative Information Systems, pp. 87-98, Vienna, Austria,

May 1995

ELSEWISE Project Homepage, http://www.lboro.ac.uk/elsewise/

Enterprise Java Beans –EJB Specification (1999) http://java.sun.com/products/EJB

ESPRIT (1998), AgentLink home page http://www.AgentLink.org/

Etzioni, E., Weld, D.(1994) "A Softbot-Based Interface to the Internet", pp.72-76, Communications of the ACM, Vol.37, No.7, July 1994

EvE, an Event-Driven Distributed Workflow Execution Engine Project, http://www.ifi.unizh.ch/dbtg/Projects/EVE/eve.html

Fenster, M. Kraus, S. Rosenschein J. (1995) "Coordination without communication: Experimental validation of focal point techniques". In proceedings of the first International Conference on Multiagent systems (ICMAS 95), pp 102-108 California USA.

Fielding R. (1998) "Support for the Virtual Enterprise: web-based Development of Complex Information Products" Communications of the ACM, Vol. 41, No 8, August 1998, pp. 84-92

Filos, E., Ouzounis, V. (2000) "Virtual Organisations: Technologies, Trends, Standards and the Contribution of the European RTD Programmes" International Journal of Computer Applications in Technology, Special Issue: "Applications in Industry of Product and Process Modeling Using Standards"

Finin, F. Labrou, Y., Mayfield, J. (1995) "KQML as an Agent Communication Language" in Software Agents. J. Bradshaw (ed) MIT Press, Cambridge, 1995

Finin, T. et.al. (1994) "KQML as an Agent Communication Language", 3rd International Conference on Information and Knowledge Management (CIKM´94), ACM Press

FIPA (1998a) – Part 1: Agent Management (V.2.0).

FIPA (1998b) – Part 2: Agent Communication Language (V.2.0).

FIPA (1998c) – Part 3: Agent/Software Integration (V.2.0).

FIPA(1999) http://www.fipa.org/spec/FIPA98.html.

Fisher M L, Hammond J H, Obermeyer W R and Raman A (1994) Making Supply Meet Demand in an Uncertain World. Harvard Business Review, May-June 1994. pp. 83-93.

Foundation for Intelligent Physical Agents (FIPA) home page: http://www.fipa.org/.

Fowler, J. (1995) "STEP for data management, exchange and sharing" UK, Technology Apparaisals, 1995, ISBN 1-871802-36-9

Franklin, S. Graesser, A. (1996) "Is it an Agent or just a Programm?: A taxonomy for Autonomous Agents" Proceedings of the 3rd International Workshop on Agent Theories, Architectures, and Languages. Springer Verlag, 1996

Frederix, F.L.M. (1998), "Agility and Human Factors in the Virtual Enterprise, in Innovation, Agility, and the Virtual Enterprise" Proceedings of the 10th International IFIP WG 5.2/5.3 Conference, PROLAMAT 98, Trento, Italy, 9-12 September 1998, pp. 737-748.

Fuggetta, A. Picco, G.P. Vigna, G. (1998) "Understanding Code Mobility" IEEE Transactions on Software Engineering, Vol. 24, No 5 May 1998, pp 342-361

Fünfrocken S. (1998) "Transparent Migration of Java-based Mobile Agents: Capturing and Re-establishing the State of Java Programs" Proceedings of the 2nd International Workshop, MA

98, Stuttgart, Germany, Sept. 98 Lecture Notes in Computer Science 1477 Springer-Verlag, 1998, pp. 26-37

Gaedke, M. et.al. (1998) "Web Content Delivery to Heterogeneous Mobile Platforms" Workshop on Mobile Data Access at 17th International Conference on Conceptual Modeling, Singapore 1998

Garcia-Molina, H. Gawlick, D. Klein, J. Kleissner, K. Salem, K. (1991) "Coordinating Multi-Transaction Activities" Proceedings IEEE Spring Compcon, 1991

Gellersen, H. Gaedke, M. (1999) "Object oriented Web Application Development" IEEE Internet Computing Jan/Feb 1999

GENIAL Project Homepage and Papers, http://cic.cstb.fr/ilc/publicat/list.htm or http://www.c-lab.de/genial/index.html

Georgakopoulos, D. (1998) "Collaboration Management Infrastructure for Comprehensive Process and Service Management" International Symposium on Advanced Database Support for Workflow Management, Enschede, The Netherlands, May 19th, 1998

Georgakopoulos, D. Hornick, M. Sheth, A. (1995) "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure" Distributed and Parallel Databses, 3(2):119-153, April 1995

Geppert, A. Kradolfer, M. Tombros, D. (1998a) "Market-Based Workflow Management" Int'l IFIP Conf. on Distributed Systems for Electronic Commerce, Hamburg, Germany, June 1998

Geppert, A. Kradolfer, M. Tombros, D.(1998b) "Workflow Specification and Event-Driven Workflow Execution" SI-INFORMATIK 4:2, April 1998

Geppert, A. Tombros, D. (1998c) "Event-based Distributed Workflow Execution with EVE" Middleware '98, The Lake District, England, September 1998

Gibon, P., Clavier, J.-F., Loison, S., (1999), "Support for Electronic Data Interchange, in Infrastructures for Virtual Enterprises" Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999, Boston, pp. 187-208.

Ginsberg, M. (1991) "Knowledge Interchange Format: the kif of death", AI Magazine, Vol.5, No.63

Goldman S L, Nagel R N and Preiss K (1995) "Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer" Van Nostrad Reinhold, New York.

Goldszmidt, G., Yemini, Y., (1995) "Distributed Management by Delegation". In 15 th International Conference on Distributed Computing Systems. IEEE Computer Society, June 1995.

Goldszmidt, G., Yemini, Y., (1998) "Delegated Agents for Network Management". IEEE Communications Magazine, Vol. 36, No. 3. March, 1998.

Grefen, J. Pernini, B., Sanchez G. (Eds, 1999) "Database support for workflow management: The WIDE project" Kluwer Academic Publishers, ISB7923-8414-8

Grefen, J. Vonk, E. Boertjes, P. Apers, P. (1998a) "Two-Layer Transaction Management for Workflow Management Applications" Proc. 8th International Conference on Database and Expert System Applications, Toulouse, France

Grefen, P. (1998b) "Advanced Transaction Management in WIDE" International Symposium on

Advanced Database Support for Workflow Management, Enschede, The Netherlands, May 19th, 1998

Guilfoyle, C., Warner, E. (1994) "Intelligent Agents: the New Revolution in Software", Technical Report, OVUM Limited

Guttman, R. Maes, P. (1998a) „Agent-mediated Integrative Negotiation for Retail Electronic Commerce" Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET 98), Minneapolis, Minnesota, May 98

Guttman, R.H. Moukas, A.G., Paes, P. (1998b) "Agent-Mediated Electronic Commerce: A Survey" Knowledge Engineering Report, 1998

H. Schuldt, H.J. Schek, G. Alonso (1999) "Transactional Coordination Agents for Composite Systems." Proceedings of the International Database Engineering and Applications Symposium (IDEAS'99). Montreal, Canada, August 1999.

Hamilton, S. (1997) "E-Commerce for the 21st Century" IEEE Computer, Vol. 30, No 5 May 1997, pp 44-47

Hammer, M. Champy, J. (1993) "Reengineering the Corporation: A Manifesto for Business Revolution" HarperBusiness, New York, 1993

Hardwick, M., Spooner, D.L., Rando, T. and Morris, K.C. (1996), "Sharing Manufacturing Information in Virtual Enterprises" Communications of the ACM, 39 (2).

Harker PT, Ungar LH. (1996) "A market-based approach to workflow automation". Proceedings of NSF. Workshop on Workflows and Process Automation in Information Systems: State of the Art and Future Directions. 8-10 May 1996

Harold, E.R: (1998) "XML:Extensible Markup Language", IDG Books, Foster City, California, USA.

Harrison, C.G. et.al. (1995) "Mobile Agents: Are they a good Idea", IBM Research Report, RC 19887

Hayzelden, A.L.G., Bigham, J. (Eds), (1999) "Software Agents for Future Communication Systems", ISBN: 3-540-65578-6, Springer Verlag

Hoffner, Y, Schade, A. (1998) "Co-operation, Contracts, Contractual Match-Making and Bindings" 2nd International Enterprise Distributed Object Computing Workshop (EDOC 98), 3-5 November, San-Diego, USA

Hoffner, Y. Crawford, B. (1997) "Using Interception to Create Domains in Distributed Systems". Joint International Conference on Open Distributed Processing (ICODP) and Distributed Platforms (ICDP), 27-30May 1997, Toronto, Canda

Hoffner, Y: (1999) "Supporting Contract Match-Making" IEEE 9th International Workshop on Research Issues on Data Engineering, RIDE-VE 99, Sydney, Australia, 1999

Holland C, Lockett G and Blackman I (1992) "Planning for Electronic Data Interchange" Strategic Management Journal vol. 13. pp. 539-550.

Hunt, I., Caskey, K., Browne, J. (1999), "SMEs in the Virtual Enterprise - LOGSME enables IT Support, in Infrastructures for Virtual Enterprises" Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999, pp. 333-342.

IBM San Francisco Technical Homepage, http://www-4.ibm.com/software/webservers/appserv/

IEEE Network Magazine (1998), Vol. 12, No. 3, Special Issue on Active and Programmable Networks.

IKV++ (1999) "Grasshopper: An Intelligent Mobile Agent Platform – volume 2- Programmers Guide version 1.1." IKV++ 1998

INDEMAND Project Homepage, http://www.cranfield.ac.uk/sims/cim/research/research_projects/indemand/indemand_home_page.htm

ISO/IEC (1991) 9596, Information Technology, Open Systems Interconnection, Common Management Information Protocol (CMIP) – Part 1: Specification, Geneva, Switzerland, 1991.

ISO/IEC (1994) "Information Technology – Open Distributed Processing – Reference Model – Open Distributed Processing – Part 1- Overview and Guide to Use" ISO/IEC JTC 1/SC 21 Sept. 1994

ITU-T (1995) "Open Distributed Processing – Reference Model –Part 3- Architecture" May 1995

J. Miller, A. Sheth, K. Kochut, and D. Palaniswami (1998) "The Future of Web-Based Workflows" International Workshop on Research Directions in Process Technology, Nancy, France, International Workshop on Research Directions in Process Technology, position paper, Nancy, France

Jacucci, G., Olling, G.J., Preiss, K., Wozny, M. (eds) (1998), "Globalization of Manufacturing in the Digital Communications Era of the 21st Century. Innovation, Agility, and the Virtual Enterprise" Proceedings of the 10th International IFIP WG 5.2/5.3 Conference, PROLAMAT 98, Trento, Italy, 9-12 September 1998, Boston: Kluwer Academic Publishers.

Jango Homepage (1998), http://jango.excite.com

Java 2 Specification and Reference Model (1998) http://java.sun.com/

Java Messaging System –JMS Specification (1998) http://java.sun.com/products/jms

Java Servlets Specification (1998) http://java.sun.com/products/servlets

Jennings NR, Faratin P, Johnson MJ, O'Brien P, and Wiegand ME.(1996) "Using intelligent agents to manage business processes". In: Proceedings of the First International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96), London, UK, 1996. pp 345-360.

Jennings, N. (1993) "Commitments and conventions: the foundations of coordination in multi-agent systems" Knowledge Engineering Review, 8 (3), pp. 223-250

Jennings, N. (1995) "Controlling cooperative problem solving in industrial multi-agent systems using joint intentions" Artificial Intelligence, 75 (2), pp. 195-240

Jennings, N. R., Wooldridge, M. J.(eds). (1998) "Agent Technology: Foundations, Applications and Markets". Springer-Verlag 1998, ISBN 3-540-63591-2

Johnston H R and Vitale M R (1988) "Creating Advantage with Interorganizational Information Systems" MIS Quarterly vol. 12, no. 2. pp. 153-166.

Judge DW, Odgers BR, Shepherdson JW, Cui Z. (1998) "Agent Enhanced Workflow". BT Technical Journal, 16:3, 1998, pp 79-85. http://www.labs.bt.com/projects/ibsr/index.htm

Kalakota R and Whinston A B (1996) "Frontiers of Electronic Commerce" Addison-Wesley Publishing Company, Inc.: Reading, MA.

Kalakota, R. (1998), "Joined at the Bit. The Emergence of the E-Business Community" in: Tapscott, D., Lowy, A., Ticoll, D. (eds) (1998). Blueprint to the Digital Economy. Creating Wealth in the Era of E-Business, New York: McGraw Hill.

Karmouch, A.(Ed.), (1998) Special Isssue on "Mobile Software Agents for Telecommunications", IEEE Communications Magazine, Vol. 36, No. 7

Karpinski R (1997) "Extranets emerge as next challenge for marketers" Netmarketing, April 1997. p. M-4.

Katzy, B. Obozinski, V. (1999) "Designing the Virtual Enterprise" Proceedings of ICE 99, 5th International Conf. On Concurrent Engineering, The Hague, Netherlands, Mar 99, ISBN 0 9519759 8 6

Khare, R., Rifkin, A. (1998) "Capturing the state of Distributed Systems with XML" World WideWeb Journal Special Issue on XML, Vol. 2, Number 4, Pages 207-218

Khare, R: Rifkin, A. (1997) "XML: A Door to Automated Web Applications" IEEE Internet Computing, Vol. 1 No4. July/August, 1999, pp 78-87

Kiniry, J., Zimmerman, D. (1997) "A Hands-On Look at Java Mobile Agents", IEEE Internet Computing, pp. 21-30.

Klingemann, J. Wäsch, J. Aberer, K. (1999a) "Adaptive Outscourcing in Cross-Organisational Workflows" Proceedings of 11th Conference on Advanced Information Systems Engineering, Heidelberg, Germany, June 1999

Klingemann, J. Wäsch, J. Aberer, K. (1999b) "Deriving Service Models in Cross-Organisational Workflows" Proceedings of RIDE-Information Technology for Virtual Enterprises, Sydney, Australia, 1999

Kotz, D., Gray, R., Nog, S., Rus, D., Chawla, S, Cybenko, G. (1997) "AGENT TCL: Targeting the Needs of Mobile Computers". IEEE Internet Computing, July - August, 1997. pp.58-67.

Kraus, S. Sycara K. Evenchik, A.(1998) "Reaching Agreement through negotiations: a logical model and implementation" Artificial Intelligence, 104 (1-2), 1-69

Krause, S., Magedanz, T. (1996) "Mobile Service Agents enabling "Intelligence on Demand" in Telecommunications", IEEE Global Telecommunications Conference (Globecom 1996), pp.78-85, IEEE Catalog No.96CH35942, ISBN: 0-7803-3336-5, IEEE Press, 1996

Krause, S., Magedanz, T. (1997) "MAGNA - A DPE-Based Platform for Mobile Agents in Electronic Service Markets", IEEE International Symposium on Autonomous Decentralized Systems (ISADS), pp.93-102, IEEE Catalog No.97TB100111, ISBN: 0-8186-7785-6, IEEE Press, 1997

Krishnakumar, N. Sheth, A. (1994) "Specifying Multi-System Workflow Applications In METEOR" Technical Report TM-24198, Bellcore, May 1994

Kuokka D. Harad L.(1995) "A Communication Infrastructure for Concurrent Engineering" Journal Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)

Lange, D. Chang, D.T. (1996) "IBM Aglets Workbench – Programming Mobile Agents in Java – A White paper" IBM Corporation, Sept. 96

Lee H L and Billington C (1992) "Managing Supply Chain Inventory: Pitfalls and Opportunities." Sloan Management Review, Spring 1992. pp. 65-73.

Lee H L and Billington C (1993) “Material Management in Decentralized Supply Chains. Operations” Research vol. 41, no. 5. pp. 835-847.

Lee H L, Billington C and Carter B (1993) “Hewlett-Packard Gains Control of Inventory and Service through Design for Localization” Interfaces vol. 23, no. 4. pp. 1-11.

Lee J, Gruninger M, Jin Y, Malone T, Tate A, Yost G. (1997) "PIF: Process Interchange Format v.1.2". PIF Working Group.

Lee, R.M. (1998) “Towards Open Electronic Contracting Using Electronic Trade Scenarios” 1st Conference on Electronic Commerce for Small and Medium Sized Enteprises, Venice, Italy, 26-27 February.

Leichsering, A.L. (1998) “Development of a Basic Agent Communication Service for Grasshopper Platform” Diplomarbeit, Technical University of Berlin, Berlin 1998

Levesque, H. Cohen, P. Hunes, J. (1990) “On acting together” In proceedings of the Eight National Conference on Artificial Intelligence (AAAI-90), Menlo-Park, California, USA.

Lin F (1996a) “Reengineering the Order Fulfillment Process in Supply Chain Networks: A Mutliagent Information Systems Approach” Ph.D. Thesis, University of Illinois at Urbana-Champaign.

Lin F, Tan G W and Shaw M J (1996b) “Multi-Agent Enterprise Modeling” University of Illinois at Urbana-Champaign, College of Commerce and Business Administration, Office of Research, Working Paper 96-0134.

Lingnau, A. Drobnik, O. (1996) “Making Mobile Agents Communicate: A Flexible Approach” in Proceedings of the 1st Annual Conference on Emerging Technologies and Applications in Communications (etaCOM96) USA, May 1996

Lomet, D. (1993, ed) “Special Issue on Workflow and Extended Transaction Systems” IEEE Data Eng. Bull, June 1993.

M. Reichert, C. Hensinger, P. Dadam (1998) “Supporting Adaptive Workflows in Advanced Application Environments” Proc. EDBT Workshop on Workflow Management Systems, Valencia, March 1998, pp. 100 - 109

M. Reichert, P. Dadam (1997) “A Framework for Dynamic Changes in Workflow Management Systems“ Proc. 8th Int'l Workshop on Database and Expert Systems Applications, DEXA '97, Toulouse, France, September 1997, pp. 42-48

Maes, P (1994a) “Agents that reduce work and information overload.” Communications of ACM.Vol.37. No.7. 1994. pp. 31-40

Maes, P. (1994b) “Modeling Adaptive Autonomous Agents”, Artificial Life Journal, eds by C.Langton, Vol. 1, No 1&2, pp. 135-162, MIT Press

Magedanz, T. (1995) "On the Impacts of Intelligent Agent Concepts on Future Telecommunication Environments", in: Lecture Notes on Computer Science 998 - "Bringing Telecommunication Services to the People - IS&N'95", pp. 396 - 414, A. Clarke et al. (Eds.), ISBN: 3-540-60479-0, Springer Verlag, 1995

Magedanz, T. (1999a), Telecommunications Information Networking Architecture Conference 1999 (TINA99), Oahu, Hawaii, USA - 12-15 April 1999, Tutorial P: IN Evolution - Impact of Internet, CORBA, TINA, and Mobile Agent Technologies, available through http://www.tinac.com/conference/tutorials.htm

Magedanz, T. (Ed.), (1999b) Special Issue on “Mobile Agents in Intelligent Networks and

Mobile Communication Systems“, in Computer Networks Journal, ELSEVIER Publisher, Netherlands, vol. 31, No. 10, July 1999

Magedanz, T. Karmouch, A. (Eds.), (2000) Special Issue on Mobile Agents for Telecommunication Applications, Computer Communications Journal, Elsevier Publishers, Vol. 27, No.1

Magedanz, T., Glitho, R. (Eds.) (1999c), Special Issue on “Mobile Agent-based Network and Service Management“ in Journal of Network and Service Management (JNSM), Vol. 7, No. 3, Plenum Press, New York, 1999

Magedanz, T., Krause, K. (1997) "Mobile Agents - Basics, Technologies, Standards, and Applications", Invited Tutorial at 1st International Workshop on Mobile Agents, Berlin, Germany, April 1997

Malone T W and Crowston K (1990) What is Coordination Theory and How Can It Help Design Cooperative Work Systems? In Proc. CSCW '90. pp. 375-388.

Malone T W and Rockart J F (1991) Computers, Networks, and the Corporation. Scientific American vol. 265, no. 3. pp. 128-136.

Malone T W, Yates J and Benjamin R I (1987) Electronic Markets and Electronic Hierarchies. Communications of the ACM vol. 30, no. 6. pp. 484-497.

Malone, T. Crowston, K. (1991) “Towards an interdisciplinary theory of coordination”. Technical Report CCS RT #120 SS WP#3294-91-MSA, Massachusetts Institute of Technology (MIT)

Manola F. (1999) “Technologies for a Web Object Model” IEEE Internet Computing Jan/Feb 1999

Mardesish, J. (1996) “Onsale Takes Auction Gavel Elections” Computer Reseller News, July 8, 1996

Mark E. Nissen (1999) “Supply Chain Process and Agent Design for E-Commerce” In Proc. 33rd Hawaii Int'l Conference on System Sciences (HICSS-33), Maui, Hawaii, January 1999.

Martensson, N., Mackay, R., Björgvinsson, S., (eds) (1998), “Changing the Ways We Work. Shaping the ICT-Solutions for the Next Century” Proceedings of the Conference on Integration in Manufacturing, Göteborg, Sweden, 6-8 October 1998, Amsterdam: IOS Press.

McCaffer, R., Garas, F. (eds) (1999), “eLSEwise: European Large Scale Engineering Wide Integration Support Effort, Engineering Construction and Architectural Management,” Special Issue, 6 (1), ISSN 0969 9988.

McCutcheon D M, Amitabh S and Meredith J R (1994) “The Customization-Responsiveness Squeeze” Sloan Management Review, Winter 1994. pp. 89-99.

Merrick, P. Allen, C. (1997) “Web Interface Definition Language (WIDL)” W3C Note, World Wide Web Consortium 1997, available at: http://www.w3c.org/TR/NOTE-WIDL

Merz, M. Lamersdorf, W. (1996) “Agents, Services, and Electronic Markets: How do They Integrate?” in Proceedings of the IFIP/IEEE International Conference on Distributed Platforms, Dresden, 1996

Merz, M., Liberman, B. Müller-Jones, K. (1996) “Interorganisational Workflow Management with Mobile Agents in COSM” in Proceedings of the PAAM 96, London April 1996

METEOR Project http://lsdis.cs.uga.edu/proj/meteor/meteor.html

Meyer, K., Erlinger, M., Betser, J., Sunshine, C., Goldszmidt, G., Yemini, Y. (1995) "Decentralising Control and Intelligence in Network Management." Integrated Network Management IV, Chapman & Hall, 1995. Ed. Sethi et al. pp. 4-15. ISBN 0412715708.

Milgrom, P. (1989) "Auctions and Bidding: A Primer" Journal of Economic Perspectives, Summer 1989, pp. 3-22

Milgrom, P. Weber, R (1982) „A Theory of Auctions and Competitive Bidding" Econometrica, Sept. 1982, pp.1089-1122

Miller M J (1996) "Your Own Private Internet" PC Magazine vol. 15, no. 5. p. 29.

Milner R.(1995) "Communication and Concurrency". Prentice Hall, 1995.

Milosevic, Z. (1995) "Supporting Business Contracts in Open Distributed Systems" 2nd International Workshop on Services in Distributed and Networked Environments (SDNE 95), Whistler, Canada, June 1995

MISSION Project Homepage, http://www.ims.org/project/projinfo/mission.htm

Mitrovic, D., Hunter, I. Male, S. (1999), "Characteristics of Networked Enterprise in Global Construction" in Proceedings of ICE'99, International Conference on Concurrent Enterprising, The Hague, The Netherlands, 15-17 March 1999, Nottingham: University of Nottingham, ISBN 0 9519759 86 pp. 447-454.

Mohan Kamath, Krithi Ramamritham "Pragmatic Issues in Coordination Execution and Failure Handling of Workflow in Distributed Workflow Control Architectures" Oracle Corp. Technical Report.

MQSeries Specification, Reference Architecture (1998) http://www-4.ibm.com/software/ts/mqseries/

Muller, H.J. (1996) „Negotiation Principles", chapter 7, Foundations of Distributed Artificial Intelligence, John Wiley and Sons, 1996

NIIP (1996) "The NIIP Reference Architecture", available at http://www.niip.org

Nwana H, Ndumu D, Lee L, Collis J.(1999) "ZEUS: A Toolkit for Building Distributed Multi-Agent Systems". In: Applied Artifical Intelligence Journal, Vol 13, Number 1, 1999. http://www.labs.bt.com/projects/agents/index.htm

Nwana HS.(1996) "Software agents: an overview". The Knowledge Engineering Review, 11(3), 1996, pp 205-244.

OASIS, Resources http://www.oasis-open.org/html/goldfarb.htm

Object Space (1997) "Voyager Core Technology User Guide – Version 1.0.0" Object Space Inc. 1997

Oliver, J. (1996) "On Artificial Agents for Negotiation in Electronic Commerce", Ph.D. thesis, Wharton, 1996

Oliver, J.R. (1997) "A Machine Learning Approach to Automated Negotiation and Prospects for Electronic Commerce", in http://opim.wharton.upenn.edu/~oliver27/papers/jmis.ps

OMG (1994-2000), Object Management Group, Common Object Request Broker Architecture Specification, Version 2.0

OMG (1994-2000), Object Management Group, Common Object Request Broker Architecture Specification, Version 2.0

OMG CORBA (1994-2000) ”Common Object Request Broker Architecture and Specification”, Revision 2, August 1995

OMG MASIF (1997), “Mobile Agent System Interoperability Facility (MASIF) specification” November 1997, ftp://ftp.omg.org/pub/docs/orbos/97-10-05.pdf

OMG, CORBA “Asynchronous Messaging and QoS Service Control” (1999), www.omg.org/docs/orbos/98-05-05.pdf.

OMG, CORBA Component (1999), www.omg.org/docs/orbos/99-02-05.pdf.

OMG, CORBA Facilities Architecture Specification (1998), www.omg.org/corba/cf2.html.

OMG, CORBA Interoperable Naming Service (1998), www.omg.org/docs/orbos/98-10-11.pdf.

OMG, CORBA Objects-by-Value (1998), www.omg.org/docs/orbos/98-10-06.pdf.

OMG, CORBA Scripting (1998), www.omg.org/docs/orbos/98-12-08.pdf.

OMG, CORBA Services Specification (1998), www.omg.org/corba/sectran1.html.

OMG, CORBA/IIOP 2.3 Specification (1998), www.omg.org/docs/ptc/98-12-04.pdf.

OMG, Interworking between CORBA and TC Systems (1997), www.omg.org/docs/telecom/97-12-06.pdf.

OMG, Java Language to IDL Mapping (1998), www.omg.org/docs/orbos/98-04-04.pdf.

OMG, Minimum CORBA (1998), www.omg.org/docs/orbos/98-08-04.pdf.

OnSale homepage (1998) http://www.onsale.com

Orfali, R. Harkey, D. and Edwards J. (1996) “The Essential Distributed Object Survival Guide” John Wiley & Sons.

Ouzounis, V (1998a) “Electronic Commerce Commercial Scenarios, Business Models and Technologies for SME’s”, invited paper, European Multimedia, Microprocessor Systems and Electronic Commerce Conference and Exposition (EMMSEC 98), Bordeaux, France 28-30 September

Ouzounis, V. (1998b) “An Overview of Advanced Electronic Commerce Platforms and Architectures”, invited paper, Information Society Technologies (IST) 98, Vienna, Austria, 2-4 December

Ouzounis, V. (1998c) “Electronic Commerce and New Ways of Work – An R&D RoadMap”, European Commission –Directoral General III

Ouzounis, V. (2000b) “Managing Dynamic Virtual Enterprises using FIPA Agents“ submitted as a chapter in the book „Managing the Virtual Web Organization in the 21st Century: Issues and Challenges" authored by Ulrich Franke, Cranfield University

Ouzounis, V. Tschammer V. (1998d) “Plattformdienste zur Unterstützung virtueller Unternehmen im elektronischen Handel”, Praxis Informationverarbeitung und Kommunikation (PIK) Jul-Sep 1998, Volume 3

Ouzounis, V. Tschammer V. (1998e) ”Integration of Electronic Commerce Business Processes in Virtual Enterprises” European Multimedia, Microprocessor Systems and Electronic Commerce Conference and Exposition, (EMMSEC 98) Bordeaux, France, 28-30 September

Ouzounis, V. Tschammer V. (1999a) “A Framework for Virtual Enterprise Support Services”, 32nd International Conference on Systems and Sciences (HICSS32) Maoui Hawaii, 3-5 January

1999

Ouzounis, V. Tschammer V. (1999b) “Iuk –Dienste zur Unterstützung virtueller Organisationen”, Handbuch Electronic Commerce, Springer Verlag, ISBN 3-540-660008-9

Ouzounis, V. Tschammer, V. (2000a) “Einsatz mobiler intelligenter Agenten zur Unterstützung virtueller Organisationen” to appear in “Virtuelle Organisationen in der Praxis”, Springer Verlag,

Parsons, S. Sierra, C., Jennings, N. (1999) “Agents that Reason and Negotiate by Arguing” Journal of Logic and Computation, Vol. 2. pp. 34-43, 1999

Perdikeas, M.K., Chatzipapadopoulos, F.G., Venieris, I:S., Marino, G. (1999) "Mobile Agent Standards and Available Platforms", Computer Networks Journal, Special Issue on “Mobile Agents in Intelligent Networks and Mobile Communication Systems“, ELSEVIER Publisher, Netherlands, vol. 31, issue 10

Pereira Klen, A.A. Rabelo, R.J. Spinosa, L.M. (1999) “Distributed Business Process Management” in Infrastructures for Virtual Enterrises. Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999, Boston: pp. 241-258

Petrie, Charles J.,(1996) “Agent-Based Engineering the Web and Intelligence”, IEEE Expert, December 1996, also http://cdr.stanford.edu/NextLink/Expert.html

Pozzi, G. Ceri, S. (1998) “Exception Management in WIDE” International Symposium on Advanced Database Support for Workflow Management, Enschede, The Netherlands, May 19th, 1998

R. Hull, F. Llirbat, E. Simon, J. Su, G. Dong, B. Kumar, G. Zhou. (1999) "Declarative Workflows that Support Easy Modification and Dynamic Browsing” Proc. Int. Joint Conf. on Work Activities Coordination and Collaboration, 1999.

R. Hull, F. Llirbat, J. Su, G. Dong, B. Kumar, G. Zhou. (1998) "Adaptive Execution of Workflow: Analysis and Optimization". Bell Labs Technical Report, November, 1998.

Raiffa, R. (1982) “The art and science of negotiation” Harvard University Press, Cambridge, USA

Redlich, J.-P. Suzuki, M. and Weinstein, S. (1998) “Distributed object technology for networking” IEEE Communications Magazine, Vol. 36, No. 10, October, pp. 100 –111.

Reichert, Dadam (1998) “ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control” Journal of Intelligent Information Systems (JIIS), Special Issue on Workflow Management Systems, Vol. 10, No. 2, pp.

Reinhardt, A. (1994) "The Network with Smarts", BYTE Magazine, pp. 51-64, October 1994

Rosenschein, J. Zlotkin, G. (1994) “Rules of Encounter: Designing Conventions for Automated Negotiations among computers”, MIT Press, 1994

Rubin, H., Natarajan, N. (1994), ”A Distributed Software Architecture for Telecommunication Networks”, IEEE Network, Vol.8, No.1, January/February 1994

Russell, S. J., Norvig, P.(1995) “Artificial Intelligence: A Modern Approach. Englewood Cliffs, NJ: Prentice Hall, 1995.

Sandholm, T. (1995) “Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework” First International Conference on Multi-Agent Systems

(ICMAS-95), San Francisco, CA. USA. 1995

Searle, J. (1969) “Speech Acts - An essay in the Philosophy of Language” Cambridge University Press, U.K.

Sheth A.(1998) "Changing focus on interoperability in information systems: from system, syntax, structure to semantics" In Interoperating Geographic Information Systems goodchild, M., Egenhofer M., Fegeas R., Kottman C. (eds.), Kluwer.

Sheth, A. (1997) "From Contemporary Workflow Process Automation to Adaptive and Dynamic Work Activity Co-ordination and Collaboration". Proceedings of the Workshop on Workflow Management in Scientific and Engineering Applications, Toulouse, France, September 1997.

Sierra, C. Faratin, P. Jennings, N. (1997) “A Service-oriented Negotiation Model between Autonomous Agents” Proc. 8th European Workshop on Modeling Autonomous Agents in a multi-agent world (MAAWAW 97), Ronneby, Sweden, pp 17-35

Smith R G and Davis R (1988) “Frameworks for Cooperation in Distributed Problem Solving”. In Readings in Distributed Artificial Intelligence, Ed. Alan H. Bond and Les Gasser, Morgan Kaufmann Publishers, Inc.: San Mateo, CA.

Smith, R.G. (1980) “The Contract Net Protocol: High Level communication and control in a distributed problem solver” In Reading in Distributed Artificial Intelligence, Morgan Kaufmann, San Mateo, California

Snapp C D (1990) “EDI Aims High for Global Growth” Datamation, March 1, 1990. pp. 77-80.

Spinosa, L.M., Rabelo, R., Klen, A.P. (1998), “High-level Co-ordination of Business Processes in a Virtual Enterprise”, in Innovation, Agility, and the Virtual Enterprise, Proceedings of the 10th International IFIP WG 5.2/5.3 Conference, PROLAMAT 98, Trento, Italy, 9-12 September 1998, pp.725-736.

Srinivasan K, Kekre S and M and Mukhopadhyay T (1993) “Impact of Electronic Data Interchange Technology on JIT Shipments” Graduate School of Industrial Administration, Carnegie Mellon University.

Stamos, J.W., Grifford, D.K. (1990): "Implementing Remote Evaluation", IEEE Transactions on Software Engineering, Vol.16, No.7, pp. 710-22

Stricker, C. Riboni, S. Kradolfer, M. and Taylor. J. (2000) “Market-based Workflow Management for Supply Chains of Services.” In Proc. 33rd Hawaii Int'l Conference on System Sciences (HICSS-33), Maui, Hawaii, January 2000.

Stricker, C. Riboni, S. Kradolfer, M. Taylor J. (1999) “Market-Based Workflow Management for Supply Chains of Services” In Proc. 33rd Hawaii Int'l Conference on System Sciences (HICSS-33), Maui, Hawaii, January 1999

Swaminathan J M, Smith S F and Sadeh N M (1994) “Modeling the Dynamics of Supply Chains” The Robotics Institute, Carnegie Mellon University.

Tambe, M. (1997) “Towards Flexible Teamwork” Journal of Artificial Intelligence Research, 7, 83-124

Taubes, G. (1998) “Taming Virtual TaskMasters” Network Series, IBM Research Vol. 36, No3, 1998 pp. 7-9

TEAMS Project Homepage, http://cewww.eng.ornl.gov/team/int.html

The ACE-FLOW Project (1999) http://www.ifi.unizh.ch/dbtg/Projects/ACEFLOW/index.html

The ProcessLink Project http://www-cdr.stanford.edu/ProcessLink/

The SEAMAN Project http://www.ifi.unizh.ch/dbtg/Projects/SEAMAN/seaman.html

Thompson, S.G. Odgers, B.R. Shepherdson, J.W. (1999) "Cross Organisational Workflow should be Co-ordinated by Software Agents" Workshop on Cross-Organisational Workflow Management and Co-ordination February 22nd 1999, San Francisco

Tigue, J. Lavinder, J. (1998) "WebBroker: Distributed Object Communication on the Web" W3C Note, World Wide Web Consortium 1998, available at: http://www.w3c.org/TR/1998/NOTE-webbroker

Timmers, P., Stanford-Smith, B., Kidd, P.T., (eds.) (1998), "Electronic Commerce: Opening Up New Opportunities for Business" Macclesfield: Cheshire Henbury.

Tombros, D. (1999) "An Event- and Repository-Based Component Framework for Workflow System Architecture" University of Zurich, 1999

Tombros, D. Geppert, A.(2000) "Building Extensible Workflow Systems using an Event-Based Infrastructure", Proc. 12th Conf. on Advanced Information Systems Engineering, Stockholm, Sweden, June 2000

TRAMS Project http://www.ifi.unizh.ch/dbtg/Projects/TRAMs/trams.html

Tschammer, V, Mendes, Ouzounis V. (1997) 'SECCO – Support Environment for Electronic Commerce", author, Cost 237 Workshop, Lisboa, Portugal, December 15-19, 1997

Tschammer, V. Ouzounis, V. (2000) "Dynamic Virtual Enterprises– a Prospect of Collaborative Commerce Between SMEs", 2nd South Eastern European Conference on e-Commerce, 24 – 26 Oct., 2000, Sofia, Bulgaria

UMBC Knowledge Sharing Effort, http://www.cs.umbc.edu/kse or http://www.cs.umbc.edu/agents/

Unified Modelling Language (UML) Specification, http://www.rational.com/uml/index.jtmpl

VEGA Project Homepage and Papers, http://cic.cstb.fr/ilc/publicat/list.htm

Veloso, M. Pollack, M. Cox, M. (1998) "Rationale-based monitoring for planning in dynamic environments". In proceedings of Artificial Intelligence Planning Systems (AIPS-98) Pittsburg,

VENTO Project Homepage, http://www.cas-software.de/CAS/EU-Vento-Kurz-Architecture.htm

Vickerey, W.(1961) "Counterspeculation, Auctions, and Bidding" Journal of Finance, March 1961, pp 8-37

Violino B (1996) "Your Worst Nightmare" Information Week, Feb. 19, 1996. pp. 34-36.

VIVE Reference Model (1999) http://www.ceconsulting.it/VIVE/Results/default.html

Walles, M.G. (1999) "WIDL: Interface Definition for the Web" IEEE Internet Computing Jan/Feb 1999

Walton, J. Whicker, L. (1996) "Virtual Enterprises: Myth and Reality", J. Control, Oct. 96

Weitzel, T. Buxmann P. Westarp F. (1999) "A Communication Architecture for the Digital Economy - 21st Century EDI" In Proc. 33rd Hawaii Int'l Conference on System Sciences (HICSS-33), Maui, Hawaii, January 1999

WfMC Terminology and Glossary (WfMC-TC-1011, Feb 1999, 3.0) available at http://www.wfmc.org/

White J.E. (1994) "Telescript Technology: The Foundation for the Electronic Marketplace", General Magic White Paper

White, J. E. (1997) “Mobile Agents”. In Bradshaw, J. M., (Editor) (1997) ‘Software Agents’. MIT Press, ISBN 0-262-52234-9. Chapter 20. pp437-472.

WIDE Project http://dis.sema.es/projects/WIDE/

WISE Project http://www.inf.ethz.ch/department/IS/iks/research/wise.html

Wognum, N., Thoben, K.-D., Pawar, K. S. (1999a), Proceedings of ICE’99, International Conference on Concurrent Enterprising, The Hague, The Netherlands, 15-17 March 1999, Nottingham: University of Nottingham, ISBN 0 9519759 86.

Wognum, P.M., Faber, EC.C. (1999b), “A Framework for Improving the Quality of Operation in a Virtual Enterprise” in Infrastructures for Virtual Enterrises. Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999, p. 365-376.

Wood L. (1999) “Programming the Web: The W3C DOM Specification” IEEE Internet Computing Jan/Feb 1999

Wooldridge, M and Jennings, N. R. (1995a) “Intelligent Agents: Theory and Practice” The Knowledge Engineering Review. Vol. 10. No. 2. 1995. pp.115-152.

Wooldridge, M., Jennings, N (1995b) Intelligent Agents: Theory and Practice. The Knowledge Engineering Review 10(2). pp.115-152. 1995.

Workflow Management Coalition (1993-2000) Interoperability Abstract Specification, October 1996, available at: http://www.wfmc.org

Wurman, P. Wellman, et.al. (1999) “A Control Architecture for Flexible Internet Auction Servers” First IAC Workshop on Internet-based Negotiation Technologies, New-York, USA, 1999

XML Resources and Papers, http://www.w3c.org/xml/ or http://www.xml.org/xmlorg_resources/index.shtml

Yemini Y., Goldszmidt G., Yemini S., (1991) “Network Management by Delegation”, Proceedings of the 2nd International Symposium on Integrated Network Management, pp. 95-107.

Zapf, M., Herrmann, K., Geihs, K., (1999) „Decentralised SNMP Management with Mobile Agents“, in Integrated Network Management VI, Sloman, Mazumdar, Lupu, eds., pp. 623-635, IEEE.

Zarli, A. Poyet, P. et. al. (1997) “Integrating Emerging IT paradigms for the Virtual Enterprise: the VEGA Project” Proceedings of ICE 97, 4th International Conf. On Concurrent Engineering, Nottingham, UK, Oct 97,

Zarli, A., Poyet, P. (1999), “A Framework for Distributed Information Management in the Virtual Enterprise” The VEGA Project, in Infrastructures for Virtual Enterrises. Networking Industrial Enterprises. IFIP TC5 WG5.3 / PRODNET Working Conference for Virtual Enterprises (PRO-VE'99), Porto, Portugal, 27-28 October 1999, pp. 293-306.

Zeng, D. Sycara, K. (1996) “How Can agents learn to negotiate?” Intelligent Agents III: Agent Theories, Architectures, and Languages (Proceedings of ECAI 96), LNCS. Springer, August 1996

Zeng, Dajun, Sycara (1996) „Bayesian Learning in Negotiation“ Working Notes of the AAAI 1996 Stanford Spring Symposium Series on Adaptation, Coevolution, and Learning in Multi-agent systems, also at http://www.cs.cmu.edu/~zeng

Zhang, T., Magedanz, T., Covaci, S. (1998) ”Mobile Agents vs. Intelligent Agents - Interoperability and Integration Issues”, 4th International Symposium on Interworking, Ottawa, Canada