DISTRIBUTED AND FLEXIBLE WORKFLOW COORDINATION USING DEFEASIBLE LOGIC PROGRAMMING

by

Sakeer P V

A thesis submitted in partial fulfillment

of the requirements for the degree of

Masters in Informatics

Supervised By : Dr. Iyad Rahwan

Co-Supervisor: Dr. Khaled Shaalan

The British University in Dubai

Dec 2007

Abstract

This is an exploration to foretell a future in which traditional approaches for developing workflow management systems can be supplanted by new techniques and emerging technologies. This thesis recommends sets of methodologies for performing multiagent-based distributed and flexible workflow systems. Its objective is to deal with some of the present issues in the traditional workflow system from the business point of view. This thesis proposes that the traditional design of workflow management systems (client-server) could be replaced by a defeasible logic programming (DeLP) engine–based multiagent platform that is more flexible and can better replicate workflow's distributed characteristics in the open environment. This thesis presents sets of technologies for enacting multiagent-based distributed and flexible workflow systems. Its purpose is to tackle some of the existing problems in the traditional workflow system from the business point of view. This thesis proposes that the conventional system architecture of workflow management systems (client-server) could be replaced by a defeasible logic programming (DeLP) engine–based multiagent platform that is more open and collaborative, and can better reflect workflow's distributed features in the open environment. This system also eliminates the requirement for centralized workflow coordination and proposes to build a flexible and distributed workflow management system using a multiagent system on a java agent

development (JADE) framework, where the workflow semantics and business logic are built using DeLP. The main outcome of this research is to provide approaches for utilizing defeasible logic programming methodologies in application development, especially in business applications, where DeLP can contribute much for the automation of business logic. Moreover, a multiagent system on a JADE framework helps to maximize the use of existing process models and tools for automation of business processes. The model implemented as part of this thesis confirms that a workflow framework using DeLP improves the adaptability and decentralization of workflow management.

Table of Contents

Acknowledgments

I sincerely express my deepest gratitude to my supervisor, Dr. Iyad Rahwan, and cosupervisor, Dr. Khaled Shaalan, who supervised, supported, and guided me to the completion of this project. Without their consistent support, I would not have been able to complete my thesis and this manuscript. I also express my sincere gratitude to my informatics colleagues and friends, whose timely and sincere guidance, during various phases, has indeed helped me to realize this project. I thank the British University in Dubai for offering me facilities during this endeavor. I am indebted to the Emirates Group for offering me a scholarship to fund my master's in informatics program.

Chapter 1: Introduction

In today's competitive environment, to achieve operational efficiency together with customer satisfaction, while improving consistency and quality, organizations are forced to implement novel systems that enable business processes that are responsive to the volatile nature of the business. Here comes into play the significance of a workflow management system for business process management. Nowadays, information technology is an essential part of business that reduces integration friction between applications and business functions. Workflow management systems (WMSs) can be widely used to manage business processes for their automation, coordination, and collaboration between entities. In other words, WMSs are systems that identify, control, and run tasks through the execution of specific software, the workflow of which is driven by a system representation of the workflow logic (Workflow Management Coalition [WfMC], 1999). While WMSs focus on managing process logic, they need to integrate other technologies to fully control business processes such as task assignments, resource allocation, and so on. In short, WMSs are developed to improve the efficiency of business procedures by executing tasks in an appropriate order, giving adequate access to the resources essential for performing the given task, and overseeing all aspects of the processes' execution. Broadly, WMS functionalities can be classified into the

design-time function, dealing with identifying and representing the workflow procedures and its tasks, and the executing-time function, related to generating and managing the workflow instances in an operational environment. To provide the preceding functionalities, the WMS has components to store and execute definitions, generate and administer workflow instances as they are executed, and manage their interfaces with workflow members and applications (WfMC, 1999). As a result of the advancements made in multiagent systems, workflow systems paved the way for the development of the agent-based workflow management system, which helps us in achieving the goal of integration of various technologies to improve functionality. "An *agent* is a computer system situated in some environments that is capable of autonomous action in this environment to meet its design objectives" (Woolridge, 2001).

Motivation

The traditional approach of the existing WMS systems and the unautomated agent coordination in the changing environment has always been a challenge in workflow systems. Industrial experts always dreamt of building a system that could respond to the changing environment, i.e., one that was adaptive to the changing environment/beliefs. This is the first and foremost motive behind this research.

It is found that the proper use of WMSs can make processes more cost-effective and in turn reduce turnaround times for business processes and improve productivity and service quality. Certainly WMSs have become an essential part of any organization as they are capable of incorporating different resources, such as various systems and applications, into the organization and its human elements (Mohan, 1998; Schmidt, 1999). A number of business workflow management systems are available such as FlowMark, InConcert, METEOR, Visual WorkFlow, ActionWorkflow and so on .

The main concerns of industrial experts concerning the current system are adaptability limitations of the system, centralization bottleneck, and runtime binding of the workflow semantics (P. George, Zurich Insurance Dubai, personal communication, June 20, 2006). The task management system (TMS) that is being developed and used by one of the international insurance organizations is analyzed as part of the problem analysis. This system is being used at the organization for distributing work among the departments and for the coordination of the tasks assigned. This system requires many manual interventions to reroute the task between the departments in case of change in the workflow semantics. At times, due to the lack of decentralization and context awareness, redundant tasks are handled by various departments, despite a case not being taken up. Most of

the time, even though the knowledge is available to a department, in the case of a re-query or an argument demand, manual intervention is required to handle such specific situations. Also, the centralized nature of the system demands that the various departments communicate back to the central desk with feedback.

This thesis reviews the above discussed shortfalls and limitations of conventional WMSs based on the client-server architecture and propose a novel idea to overcome these limitations. This innovative approach for building WMSs integrates multiagent architecture using JADE and defeasible logic programming (DeLP). The motivations of this research and an overview of the structure of this thesis are given in the following sections.

Objectives

After careful consideration of the present issues in traditional workflow systems and evaluation of some of the existing methods for workflow systems, it is found that resolving issues connected to flexible and distributed workflow design has become important for the design of forthcoming WMSs. In this research the main objective is to implement a WMS that can act based on changing beliefs or knowledge (i.e., adaptability using an argumentative approach). Also, it aims to form the semantics of workflow structure using DeLP and the distributed mechanism to overcome the other limitations of central coordination.

Methodology

Distributed and flexible workflow coordination using DeLP has been influenced by a model of the argumentative approach of DeLP theorists Garcia and Simari (2004). They have analyzed various aspects of the argumentative approach on DeLP in their paper. "DeLP is a paradigm that combines logic programming and defeasible argumentation. DeLP provides the possibility of representing information in the form of weak rules in a declarative manner and a defeasible argumentation inference mechanism for warranting the entailed conclusion" (Garcia & Simari, 2004). Defeasible reasoning adapts the rule-based reasoning methodology to arrive at a conclusion even when incomplete and inconsistent facts are available. This method is useful for system integration, where contradicting or incompatible information may arise in real time/runtime, and for the representation of business policies and rules where policies and rules with exceptions are often used. This enables a workflow system to use DeLP for coordination and forming well-defined workflow semantics. DeLP is simple to use, with strict and defeasible rules and priorities, based on translation of logic programming with declarative semantics that are flexible and adaptable to different intuitions within defeasible reasoning. Integrating DeLP with a multiagent system in a JADE environment brings a new approach toward the development of flexible and distributed workflow coordination. With this

unique approach, limitations of the traditional workflow systems can be overcome at a certain level. The main objective of this thesis is to develop a prototype implementation for the new approach, where agents can respond with common sense and react to changing beliefs.

Contributions

The significance of this research is that it introduces a new direction for tackling some of the unsolved problems in traditional workflow and also provides new methodologies for improving the adaptability of the workflow system. The major results achieved by this research are using a distributed, flexible, and multiagent design to deploy distributed workflow management systems. The major contributions of this thesis are the following:

- introduction of the argumentative approach for workflow management coordination
- identification of how adaptability of the workflow system can be best improved using a DeLP approach
- analysis of a new approach to formalize workflow semantics and runtime binding of workflow semantics
- implementation and demonstration of the new framework and analysis of how it contributes toward flexibility and distributed workflow coordination

Overview of This Thesis

In chapter 2, a background of the workflow system, limitations of the conventional workflow systems, and some related work is analyzed and discussed. In addition, problem space and methodology are also discussed at the end of the chapter. Chapter 3 describes the proposed framework on theoretical grounds and explores the main ideas behind the thesis. Chapter 4 focuses on implementation details specific to the insurance domain and demonstrates the working methodology using case studies. The last chapter, chapter 5, summarizes the ideas discussed in this thesis, the main contributions of this research, and future research directions.

Chapter 2: Background on Workflow, Distributed Workflow, and Defeasible Logic Programming (DeLP)

In today's business environment, WMSs have a significant role as an infrastructure for automating interorganizational interactions such as interdepartmental coordination in a financial organization. In such an environment, dynamic knowledge sharing and distributed workflow coordination are crucial to the smooth functioning of the business processes, whereas a traditional model may cause a performance bottleneck due to the centralized processing structure. The main concepts discussed in this context are WMS, multiagent-based WMS, adaptive workflow processes, and distributed workflow processes. In this chapter, a detailed discussion of the above concepts is handled. Additionally, a problem analysis and a proposed solution are also discussed.

Workflow Management System (WMS)

The Workflow Management Coalition (WfMC) is an international organization responsible for setting standards for workflow suppliers, user-groups, programmers/analysts, and university/research groups. The WfMC has been in charge of the formation of a workflow reference model and a glossary of standard workflow vocabulary and terminologies. Several fundamental terminologies are key to understanding the nature of workflow

and for a discussion of current trends in WMSs. The WfMC Terminology and Glossary document provides the following definitions (WfMC, 1999):

> "Workflow: the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules. "
>
> "Business Process: a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships."
>
> "Process Definition: the representation of a business process in a form which supports automated manipulation, such as modeling, or enactment by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc. In summary, a process definition is an abstract representation of a business process that can be consumed by a workflow management system in order to enact the workflow."
>
> "Workflow Management System: a system that defines, creates and manages the execution of workflows through the use of software,

> running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications."

Agent-Based WMSs

To introduce the agent-based WMS, it is worthwhile to start with an overview of the agent and multiagent systems. Reactivity, adaptability, autonomy, mobility, and collaborative behavior are some of the standard characteristics of an agent explained by Bradshaw (1997). The most significant characteristic is autonomy. Wooldridge (1999; see also Shoham, 1997) explained the differentiation between agents and objects. According to these authors, the main three differences are all connected to the autonomy of agents: Agents have control over their behavior; agents can behave reactively or proactively, that is, flexibly; and agents mostly have their own threads of control. According to Sycara (1998), there are lot of advantages to adopting a multiagent methodology for developing software. For instance, a multiagent methodology can provide encapsulation and abstraction. Additionally, as the agents are autonomous, each agent can determine by itself the appropriate approach for resolving its specific challenge. These agents can be developed by various programmers, as long as they can communicate with each other. In addition, the multiagent system offers distributed, flexible, and open

design. Thus agents can interface and interact with the dynamic system without understanding all the elements in advance.

WfMSs using software agents have been developed with various aspects. In some instances, the agents execute specific functions that are required by other tasks in the workflow. In other situations, the present workflow is used to align the coordination of these agents (Jennings, Faratin, Norman, O'Brien, & Odgers, 2000; Joeris, 2000; Nissen, 2000). For example, Nissen (2000) described an approach to developing a set of agents to execute activities connected with the supply chain process in e-commerce. In other cases, the agents have been used as part of the infrastructure linked with the WfMS itself to create an agent-enhanced WfMS (Stormer, 2001; Wang & Wang, 2002). These agents offer an open system with loosely coupled elements that are more flexible compared to conventional methods. Some researchers have come up with a combination of these methods (Chen, Hsu, Dayal, & Griss, 2000), where an agent-based WfMS is used in combination with specific and dedicated agents that offer apt system-related services. In most cases, agents are working toward their own objectives, and either whole knowledge is available to everyone or no knowledge is shared.

Adaptive Workflow Systems

Adaptive workflows have been explored by a number of researchers for many years, and most have described what should be done. Only a few

researchers have presented methodologies to manage adaptability, and only a few real implementation attempts have been made to resolve some aspects of adaptability. Reassigning and handing over a task under process to a new model is yet an issue to deal with. This is indicated in a study of different WfMSs that was conducted by van der Aalst, Hofstede, Kiepuszewski, and Barros (2002). The method explained in most of the studies is to predefine a limited set of potential reassignments and relocations, such as adding an additional task in the series, by passing a task without execution or interchanging a task with a new one. In this approach, the semantics of the workflow remain well defined. The drawback of this method is that the set of potential reassignments and relocations is very limited and/or the expressive power of the specification language is too restrictive, limiting the possibilities for specifying a process. For instance, the method of van der Aalst, Basten, Verbeck, Verkoulen, and Voorhoeve (1999) is based on inheritance, but it requires that all workflow should be derived according to a limited set of transitions from some basic workflow definition.

Adaptability in overseeing the tasks and responding to feedback mechanisms of workflow systems has been discussed by many people for many years. Few papers have discussed the problems connected with monitoring and feedback (Cui, Odgers, & Schroeder, 1998; Muehlen & Rosemann, 2000). When it comes to agent-based oversight of tasks, there has

been one presented solution (Wang & Wang, 2002), but this does not offer response to feedback and also lacks distributed monitoring, which is vital to any workflow system, as described by van der Aalst et al. (2002).

Distributed Workflow

The distributed workflow system is a hot topic in research related to WMSs. Tremendous research efforts are carried out in this area. The significance of integrating workflow management with distribution has been dealt with by Weissenfels, Dittrich, Muth, Wodtke, and Weikum (1998), Eder and Panagos (1999), and Jablonski et al. (1999). A number of basic methods and prototypes have been presented and designed that attempt to resolve these issues, which makes traditional distributed WMSs more complicated.

One of the main research directions in distributed workflow is Application Development Based on Encapsulated Premodeled Process Templates (ADEPT). This venture was initiated at the University of Ulm in 1996, with the objective of developing a new workflow methodology for commercial workflow management (Rinderle, Reichert, & Dadam, 2003). The key purpose of the ADEPT program is to achieve distributed workflow control to overcome the bottleneck of the workflow servers because of overloading and the communication network. To resolve these issues, ADEPT adapts the methodology of transferring the control of workflow

tasks from one server to another during execution time and reduces the load by partitioning workflow definitions; that is, a workflow task may be controlled by more than one server. When carrying out such a transfer and migration, a report of the states of instance is communicated among various servers. This report contains detailed data about activity stages and workflow. To prevent unwanted communication among servers, the execution of simultaneous workflow instances is controlled. ADEPT's message transfer functionalities can be further improved with changes in the expressions used for the allocation of servers. These expressions could be determined at the developmental stage, choosing an appropriate workflow server to store most of its communication, and involve less extra effort at the time of execution (Li, 2006). Also, ADEPT allows fixed and dynamic server allocation (Bauer & Dadam, 1999). The fixed assignment allows suitable workflow servers to be selected for different partitions of a workflow definition. In dynamic allocation, different servers are allocated at runtime, which makes the system perform better. However, this approach made the system sophisticated and complex to use.

Another system in this area is METUFlow (Cingil et al., 1997), which is a distributed WMS designed by researchers at Middle East Technical University. METUFlow uses a methodology for distributed workflow that requires a number of schedulers on various network nodes.

Every scheduler deals with different tasks in the process instances. Hence such a workflow system is best suited for distributed environments to make performance of systems better. METUFlow is founded on the study that controlling event triggering enables better task coordination in the workflow; that is, relationships among the tasks are presented by dependencies of events. In METUFlow, every single event is in control of its execution time and chooses the right time to occur to help the workflow's distributed execution. A temporal expression is set on an event as a guard, and event occurrence is only allowed if this guard is set to true. In addition, a Common Object Request Broker Architecture (CORBA) object is used to implement an interface with a guard handler that sends and receives messages (Yang, 2000).

As Web services turn out to be well accepted and extensively utilized as tools for enterprises, many methods have been presented to implement distributed workflow systems using Web services, in which the client's part is handled by Web services and process flows are controlled by a centralized engine that has interfaces among different Web services. Some examples of systems using such a methodology are Business Process Execution Languages for Web Services (BPEL4WS; Technical Report, 2003) and OWL-S (OWL; Technical Report, 2001).

BPEL4WS (BPEL4WS; Technical Report, 2003) formalized a method to properly specify business processes and interface protocols. This developed the Web services interaction architecture and helps it to support business transactions. BPEL4WS defines an interoperable integration model that should allow the widening integration of the automated process in business-to-business cases and intraenterprise situations. OWL-S is an ontology Web language (OWL)–based Web service ontology that provides a set of markup language constructs for describing the properties and capabilities of their Web services in a clear, computer-understandable format. OWL-S markup of Web services facilitates the computerization tasks in Web services, which comprise discovery of Web services and their integration, execution, and composition. After development of the layered approach for markup languages, OWL-S built on the OWL (OWL; Technical Report, 2001) recommendation created by the Web Ontology Working Group at the World Wide Web Consortium.

The methodologies described previously helped the workflow system to improve its performance and flexibility together with distributed features. The challenge is that all these approaches were still based on client-server architecture. Hence these methods either resolved the issues partially or required the definition of complex algorithms and programming methods to handle the situation. Additionally, the centralized services approach, such as

centralized assignment of tasks and process instantiation, made them useless in some application domains. As a result, the problems that are associated with centralized system architecture cannot be fully resolved if the workflow management system is built on client-server architecture.

Agents and Distributed Workflow Coordination

The development of new technological strategies, such as multiagent systems, has made available novel approaches to support process management, while traditional distributed workflow methods fail to rightly resolve some of the issues such as adaptability to changing environments, flexible workflow semantics, and so on, and some limited research efforts have explored using these collaborative and decentralized platforms to support WMSs.

A language for programming the coordination of agents with a graphical syntax and clearly defined with operational semantics is defined by Little -JIL. This is founded on two key ideas: Coordination control structures can be separated from other process programming language matters, and the processes execution can be handled by agents who know to handle their process but can benefit from support coordination. Hence, every step is allocated to an execution agent, and agents are accountable for commencing each steps and executing the task available with them. This method is more

efficient in defining and expressing the agent coordination feature of workflow.

The preceding approach gives up traditional client-server architecture and adopts a distributed methodology to develop a workflow management system. These few strategies, which utilize a multiagent computing model with current workflow management strategies, have introduced a novel approach to workflow management and the process support area. The flexibility of the multiagent computing method makes it appropriate to handle issues associated with client-server architecture. However, from the literature review of existing multiagent-based workflow models, it is obvious that the multiagent method is still immature, with many issues addressed inefficiently.

Additionally, the preceding approaches concentrate on decentralizing the workflow management process at execution time to address performance bottlenecks and offer more flexibility and openness to the system. On the other hand, many features that are key to distributed workflow management have not been tackled efficiently by these methods. For example, it is difficult to understand how the data of process definition are controlled and executed for distributed workflow agents to access task information at execution time. Also, process initiation is not clearly defined and addressed

by these methods. In addition to the preceding problems, dynamic agent selection is also not efficiently addressed by this method.

Problem Space and Solutions

After detailed examination and study of the current issues in the traditional workflow management systems and evaluation of some of the present methods for workflow processes, it is found that tackling the issues related to flexible and distributed workflow management are key for the improvement of current WMSs. In the agent-based WMSs, the unchanging agent beliefs make them work based on rigid workflow semantics or a workflow model. This limits the agents to thinking independently, and the centralization approach causes system bottlenecks.

Also, according to Yang (Yang, 2002 & Li, 2006),

> "industry trends such as virtual enterprises and flattening of organizational structures indicate that the future image of business will include distributed groups of collaborating teams that combine talents and skill sets to create new methodologies and processes. Therefore, there is growing need for the next generation of workflow systems to be built in a truly distributed manner"

The emergence of multiagent architecture gives a good opportunity for the distributed workflow management systems.

The above analysis points out that limited adaptability is the main challenge in every workflow process, from semantics of workflow to feedback and monitoring of the tasks. To overcome this limitation, the way forward is to enable the agent to think independently based on changing beliefs and to influence the knowledge of other dependent agents, making them coordinate and work based on the new beliefs or knowledge. In this project, the main objective is to implement a WMS that can act based on changing beliefs or knowledge. Also, this thesis aims, to form the semantics of workflow structure using DeLP and the distributed mechanism to overcome the other limitations of central coordination.

For achieving the above, this thesis proposes a Flexible and Distributed WMS using DeLP. This system enables the protection of the individual agents' private knowledge using distributed workflow coordination and flexible and open architecture for integrating workflow management process applications and communicating directly with other participants to reduce the network traffic.

This thesis explores how to utilize the potential benefits of DeLP in distributed agent–based WMSs. Generally, it discusses mechanisms that allow software agents to loosely couple to behaviors they process. This loose coupling takes the form of runtime binding to the task structure or workflow, which defines individual behaviors. The long-term benefits of this approach

will be that an agent can use its autonomy to realign its behaviors by binding to alternative workflow structures in response to environmental dynamics.

Chapter 3: DeLP Framework for Flexible and Distributed Workflow Coordination

The proposed framework is a flexible and distributed workflow coordination using DeLP, where each agent works in its own capacity using available knowledge to achieve the predefined objective in a distributed manner. Even though the agents are distributed in their work, they are built in such a way that they can share knowledge and influence others to avoid the execution of redundant tasks and to achieve the common goal in minimum time. The system is built using the DeLP logical framework for coordinating tasks, for forming workflow semantics, and for knowledge updating and sharing.

This chapter explores the concept of DeLP and the proposed framework for DeLP-based flexible and distributed workflows. To understand the proposed framework, it is important to understand the background of DeLP. The following section explains the DeLP framework and methodology and how it is utilized for workflow management.

An Overview of DeLP

Current research in agent communication, argumentation, logic programming, and nonmonotonic reasoning has offered challenging developments in knowledge representation and common sense reasoning. Evolution in these areas is leading to significant and useful results for other

technological advancements such as the development of intelligent agents and multiagent system applications. DeLP is one of the up-and-coming research areas in common sense reasoning. DeLP provides the possibility of representing information in the form of weak rules in a declarative manner and a defeasible argumentation inference mechanism for warranting the entailed conclusion. In DeLP argumentation, formalism will be used for deciding between contradictory goals. Queries will be supported by arguments that could be defeated by other arguments. The DeLP approach allows us to deal with incomplete and contradictory information in dynamic domains. This approach is suitable for representing agents' knowledge and for providing an argumentation-based reasoning mechanism to agents.

Defeasible reasoning is a rule-based approach for efficient reasoning with incomplete and inconsistent information. This kind of reasoning is useful for system integration, where conflicting information arises naturally, and for the modeling of business rules and policies, where rules with exceptions are often used. This enables a workflow system to use DeLP for the coordination and forming of well-defined workflow semantics. DeLP is simple to use, with strict and defeasible rules and priorities, based on the translation of logic programming with declarative semantics, and is flexible and adaptable to different intuitions within defeasible reasoning.

Defeasible reasoning is a nonmonotonic reasoning approach in which the gaps due to incomplete information are closed through the use of defeasible rules that are usually appropriate. This logic performs defeasible reasoning, where a conclusion supported by a rule might be overturned by the effect of another rule.

Language of Defeasible Reasoning

A defeasible theory (a knowledge based in defeasible logic) consists of five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation (Antoniou, Billington, Governatori, & Maher, 2005; Garcia & Simari, 2004). For the purpose of discussion, these concepts are explained in the following ,the concepts explained below are take from "Embedding Defeasible Logic into Logic Programming" (Antoniou, Billington, Governatori, & Maher, 2005)

- Facts are literals that are treated as known knowledge (given or observed facts of a case).
- Strict rules are rules in the classical sense: Whenever the premises are indisputable (e.g., facts), then so is the conclusion. An example of a strict rule is "emus are birds," written formally, emu(X)-> bird(X).
- Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is "birds typically fly,"

written formally, bird(X) => flies(X). The idea is that if we know that something is a bird, then we may conclude that it flies, unless there is other, not inferior, evidence suggesting that it may not fly.

- Defeaters are rules that cannot be used to draw any conclusions. Their only use is to prevent some conclusions. In other words, they are used to defeat some defeasible rules by producing evidence to the contrary. An example is "if an animal is heavy, then it might not be able to fly," written formally, heavy(X) ~> ¬flies(X). The main point is that the information that an animal is heavy is not sufficient evidence to conclude that it does not fly. It is only evidence against the conclusion that a heavy animal flies. In other words, we do not wish to conclude ¬flies if heavy, we simply want to prevent a conclusion flies.
- The superiority relation among rules is used to define priorities among rules, i.e., where one rule may override the conclusion of another rule. For example, given the defeasible rules

  r : bird(X)) flies(X)

  r0 : brokenWing(X)) ¬flies(X)

  which contradict one another, no conclusive decision can be made about whether a bird with broken wings can fly. But if we introduce a superiority relation > with r0 > r, with the intended

meaning that r0 is strictly stronger than r, then we can indeed conclude that the bird cannot fly.

The above discussed concepts in this thesis mainly deal with facts, strict rules, and defeasible rules. For workflow coordination, the above concepts are important from an argumentative approach. The paper "Defeasible Logic Programming: An Argumentative Approach" by Garcia and Simari (2004) developed a methodology to simplify programming of the defeasible language using DeLP. According to their terminology, the programming is simplified with more notation. Some of this terminology, which is used for building the proposed framework, is explained in the following sections.

Notations -< and <-. The symbol -< distinguishes a defeasible rule from a strict one. A strict rule is represented using the symbol <-.

Defeasible logic program (d.e.l.p.). A defeasible logic program *P*, abbreviated d.e.l.p., is a possibly infinite set of facts, strict rules, and defeasible rules. In a program *P*, we will distinguish the subset $\prod$ of fact and strict rules and the subset Δ of defeasible rules. When required, we will denote *P* as $(\prod, \Delta)$.

Example 3.1

The following example is from the underwriting department of an insurance company. In this case, if the medical results are satisfactory, then the underwriting of the case becomes successful.

However, if there is a bad medical history, the underwriting fails. But there are some instances in which the medical officer can request a new medical report, and in the light of new evidence, underwriting can proceed with the case. Irrespective of all the above, the management can decide on a rejection for a case, whatever the result may be. Here the management's decision gets priority and will be treated as a strict rule. This example is given below as a DeLP program:

$\prod$ = ~uw_checkok(X) <-mangmentdecisionno(X) – rule 1
medical(1) <-true.
medicalbadhistory(1) <-tue.
medical(2) <-true.
medicalbadhistory(2) <-tue.
dmo_new_medicalrpt(2)<-true.
medical(3) <-true.
dmo_new_medicalrpt(3)<-true.
mangmentdecisionno(3)<-true.

Δ= uw_checkok(X) -< medical(X). – rule 2
~uw_checkok(X) -< medicalbadhistory(X). rule 3
uw_checkok(X) -< medicalbadhistory(X),
dmo_new_medicalrpt(X). - rule 4

According to the above defeasible rules, the answer for the query uw_check(1) is no (rule 3) and for uw_check(2) (rule 4) is yes. However, the answer for uw_check(3) is no due to its strict rule (rule 1).

Defeasible derivation. In DeLP, a set of rules is contradictory if and only if there exists a defeasible derivation for a pair of complementary literals from this set. A derivation is called defeasible if there exists information in contradiction with a literal L that will prevent acceptance of L as a valid conclusion. From Example 3.1, uw_checkok(2) has defeasible derivation. When contradictory goals can be defeasibly derived, a formalism for deciding between them is needed. DeLP uses defeasible argumentation formalism to perform such tasks. In DeLP, no priority relation is needed for deciding between contradictory goals. This characteristic maintains the declarative nature of the knowledge represented in DeLP; that is, the interaction among the pieces of knowledge is expressed as a result of the influence of the whole corpus of the agents' knowledge and not because of the language alone.

Dialectical tree. A dialectical tree is a pictorial representation of the argumentation line in a DeLP program. In a dialectical tree, every node (except the root) represents a defeater of its parent, and leaves correspond to nondefeated arguments. Each path from the root to a leaf corresponds to a different acceptable argumentation line. A dialectical tree provides a structure for considering all the possible acceptable argumentation lines that can be generated for deciding whether an argument is defeated.

Defeasible argumentation. The main idea explored and utilized in this thesis is defeasible argumentation. Informally, an argument is a minimal

and noncontradictory set of rules used to derive a conclusion. In DeLP, answers to queries will be supported by an argument. Thus, although a d.e.l.p. could be contradictory, answers to queries will be supported by a noncontradictory set of rules. In DeLP, strict rules are not part of an argument structure. Consider the defeasible logic program in Example 3.1: The literal uwcheck(2) is supported by the argument structure shown in Figure 3.1.

It is important to understand that in DeLP, the construction of an argument structure is nonmonotonic; that is, adding facts or strict rules to the program may cause some argument structure to be invalidated because it becomes contradictory.

Rebuttal or counterarguments. In DeLP, an argument may be defeated by other arguments. Usually, a query will succeed if the supporting argument for it is not defeated. To establish whether an argument is a nondefeated argument, argument rebuttals or counterargument defeaters for the argument are considered. Since counterarguments are arguments, there may exist defeaters for them, and so on.

Disagreement: Let $P = (\prod \cup \Delta)$ be a d.e.l.p. We say that two literals h and h1 disagree if and only if the set $\prod \cup \{h, h1\}$ is contradictory.

Figure 3.1. A dialectical tree.

Counterarguments: It is said that <A1,h1> counterargues, rebuts, or attacks <A2,h2> at literal h if and only if there exist subarguments <a,h> of <A2,h2> such that h and h1 disagree.

In Example 3.1 of the insurance case,{~uw_checkok(X)-< medicalbadhistory(X)} is a counterargument for {uw_checkok(X)-< medicalbadhistory(X), dmo_new_medicalrpt(X)} when an additional knowledge of dmo_new_medicalrpt(X) is available.

Dialectical analysis of defeasible argumentation. In Example 3.1, there are three defeasible rules representing tentative information about *the* underwriting *medical* check of a case. It also has a strict rule expressing that if management decides to reject the case, that case will never succeed *i*n underwriting. From the above example, it is possible to derive uw_check(2) and ~uw_check(2). For the treatment of contradictory knowledge, DeLP incorporates a defeasible argumentation formalism. This formalism allows the identification of the pieces of knowledge that are in contradiction, *a*nd a dialectical process is used for deciding which information prevails as warranted. Figure 3.*1* illustrates how the dialatical process works. This dialectical process involves the construction and evaluation of arguments that either support or infer with the query under *analysis*. Once the analysis i*s* done, the generated arguments will represent an explanation for the query. Arguments that explain an answer for a given query will be shown in a particular way using dialectic*a*l trees.

Argumentation dialogue in agent communication. One approach to agent communication is to insist that an agent not only send messages, but support them with reasons why those messages are appropriate. This is argumentation-based communication. Apart from its naturalness, there are two major advantages of this approach to agent communication. One is that it ensures agents are rational to a certain degree. In other words, agents will

only accept things if they do not have a reason not to. The second advantage of an argumentation-based dialogue system is that the reason supporting the argument can be sought. Moreover, the reason may be accepted or rejected and possibly challenged and argued against by other agents.

JADE: The tool used for realizing the proposed framework. JADE (java agents development) is a popular Foundation for Intelligent Physical Agents (FIPA)-compliant, Java-based agent development platform. FIPA is a consortium that produces standards to enhance the interoperability of heterogeneous agents (Foundation for Intelligent Physical Agents, 2001). The target agents in the demonstration system were also constructed with JADE; however, any FIPA-compliant agent tool kit would work. JADE implements the FIPA reference model for agent platforms. JADE's Remote Agent Management utility provides facilities for interacting with agents and managing the agent platform.

Figure 3.2 shows that one agent platform with two containers is executing. The main container supplies basic services to the agent platform. The directory facilitator (DF) provides yellow page services to the agents running on the platform. The DF also provides the mechanism for agents to advertise their services in the agent directory. The agent management system (AMS) provides services to the agent platform that allow the creation, deletion, and migration of agents.

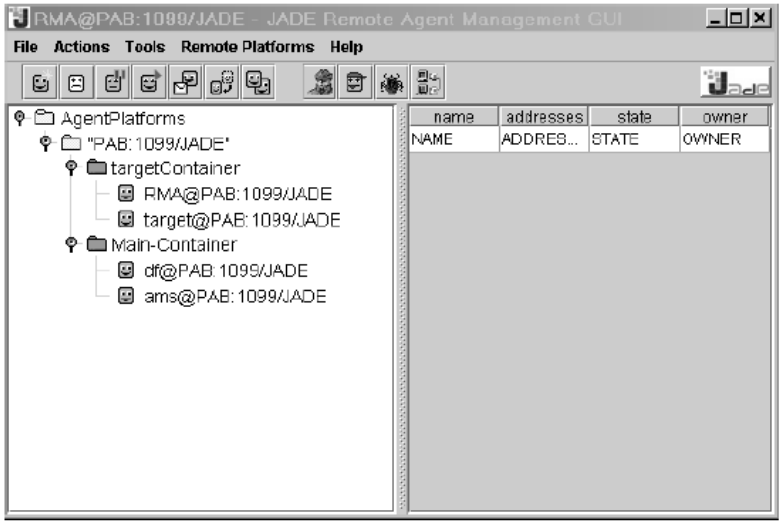

Figure 3.2. A JADE platform (Buhler, 2004).

General Architecture of the Proposed Framework

Figure 3.3 provides an architectural block diagram of the proposed framework.

Having explored the DeLP and its argumentative approach, we now need to analyze how it can be utilized in the proposed framework of a distributed WMS. The following sections illustrate how the integration is achieved using DeLP.

Recall that DeLP considers two kinds of logical rules: (a) defeasible rules used for representing weak or tentative information, like all birds can fly, and (b) strict rules used for representing strict (sound) knowledge such as that a penguin is a bird. So a defeasible rule is used to represent defeasible

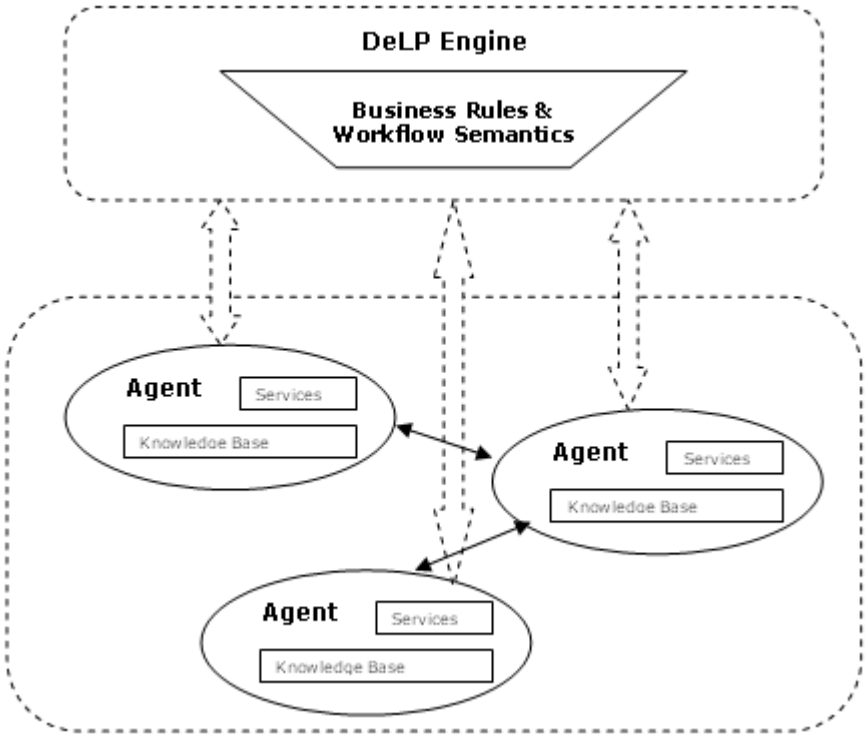

Figure 3.3. General architecture of the framework.

knowledge, i.e., tentative information that may be used if nothing could be posed against it.

DeLP and WMS. this proposal models workflow semantics and business rules in terms of a DeLP program built on top of a traditional JADE platform. Each agent is built on JADE and can be plugged into the JADE framework. Additionally, the DeLP engine is built as a stand-alone server, and agents can initiate DeLP queries on argumentation on the DeLP engine whenever required.

JADE and distributed agents. In this proposed system, all the agents' functionalities are distributed, and they are plugged into the JADE framework. Depending on the tasks, specific logic is built into the agents. However, additional agents can be plugged into the system, if required. This architecture eliminates the requirement of central coordination and thus helps to remove the bottleneck of the traditional systems due to its centralized nature.

DeLP for controlling workflow semantics. In this proposed framework, DeLP is used for controlling workflow semantics. With the available knowledge and predefined business rules, each agent can query its own workflow routing and route the task according to the business logic. For example, it considers a vetting agent task-routing scenario. Vetting agents receive and vet the case in an insurance company. On the basis of the available information, a vetting agent can query the DeLP engine for the workflow semantics of a case and should be in a position to route the case to the participating agents. In case additional knowledge is available at a later stage, other agents should be able to reroute the case directly to additional participating agents that were not involved in the earlier stage. This will eliminate the requirement for central coordination and make the workflow semantics more adaptive to the business rules. This in turn helps for runtime binding of the workflow semantics.

Example 3.3

The workflow modeling of a vetting agent in an insurance workflow structure is explored below. A vetting agent validates each application it receives and decides which department the case should go through to. The following examples show how the workflow semantics of the above case is represented in DeLP and how it is queried by the vetting agent to decide which department the case should be transferred to:

compliance_check(X)-<vetting(X).

uw_check(X) -< vetting(X).

system_input(X) -< vetting(X).

governance_check(X) -< vetting(X).

~uw_check(X) <- vetting(X), nilbenefit(X). % strict rules

Vetting(5) <-true.

Vetting(6) <-true.

Nilbenefit(6) <tue.

According to the above workflow semantics, a case will go through all the agents in a normal case, but if the case is a nil benefit case, the case does not require underwriting and has no need to go through this agent. There may be other exceptions from time to time in the business, and the system can change the workflow semantics simply by changing the rules. Here the

vetting agent decides that Case 6 does not require to be routed to an underwriting agent.

DeLP for decision making and knowledge sharing. As explained earlier, in the proposed framework, business rules are formulated as DeLP programs for the purpose of making decision and workflow coordination. Using the existing facts and available knowledge, each agent makes its own decision and communicates to peer agents using an argumentative approach. However, the peer agents have the option to accept or reject agents' decisions based on its knowledge base. In any case, it provides feedback to the originating agent. In the case of rejection, a reason is also communicated for the awareness of the originating agent, and this will in turn initiate an argumentation, if necessary. An agent can counterargue the situation based on its knowledge. This process continues until both parties reach an agreement.

Example 3.4

Consider the following business rules and facts available with two different agents.

uw_checkok(X) -< medical(X):

~uw_checkok(X) -< medical(X), medicalbadhistory(X).

~uw_checkok(X) -< medical(X),invaliddocuments(X).

~uw_checkok(X) -<

medical(X),invaliddocuments(X),medicalbadhistory(X).

uw_checkok(X) -< medical(X), dmo_new_medicalrpt(X).

~uw_checkok(X) <-

medical(X),dmo_new_medicalrpt(X),mngmtdecisionno(X)

Facts with underwriting agent	Facts with governance agent
medical(2) <- true. . medicalbadhistory(2) <-true. dmo_new_medicalrpt(2) <- true.	medical(2) <- true medicalbadhistory(2) <- true. Mngmtdecsionno(2) <-true.

In Example 3.4, the governance agent gives the final decision for a case on underwriting. Initially, the underwriting agent starts communication with the governance agent, saying that this case is ready to get approved, but when the governance agent checks it out, it has a fact that defeats the argument of the underwriting agent and goes back with a reason, informing of a bad medical history of the case. The underwriting agent replies back that it knows this client has a bad medical history; however, the medical officer has issued a clearance certificate for this case. But when the governance agent checks its knowledge base, there is a further rule that defeats the argument of the underwriting agent, saying that management has made a decision that this client cannot go with the insurance policy. The governance

agent reveals to underwriting the new status of the case, and they both reach an agreement.

Agent communication in DeLP workflow. JADE makes use of FIPA ACL(Agent Communication Language) for communication purposes. It is important to insist that such a system not only send messages, but support them with reasons why those messages are appropriate. This is argumentation-based communication. For the purpose of building this system, the following types of dialogues are used:

- *information-seeking dialogues.* One participant seeks the answer to some question from another participant who is believed by the first to know the answer.
- *inquiry dialogues.* The participants collaborate to answer some questions for which answers are not known to any single participant.
- *persuasion dialogues.* One party seeks to persuade another party to adopt a belief or point of view the second party does not currently hold.
- *negotiation dialogue.* The participant bargains over the division of some scarce resources in a way acceptable to all, with each individual party aiming to maximize its share.

The main issue in a workflow scenario is that an agent needs to be communicated in a way so as to reach an agreement quickly to benefit both

agents; to achieve this, agents do not just exchange facts, but also exchange additional information, if required. In persuasion dialogues, which are by far the most studied type of argumentation-based dialogues (Parsons & McBurney, 2003), these reasons are typically the reasons why the facts are thought to be true. Thus if Agent A wants to persuade Agent B that *P* is true, it does not just state the fact that *P* is true, but also gives the reason for it. Using DeLP in this situation, Agent B can justify itself how it is true. If Agent B is not justified on the reason, it can continue the argumentation with reasons until they reach an agreement. To further elaborate on the communication protocol, the communication strategy between the underwriting agent and the governance agent is illustrated in Figure 3.4.

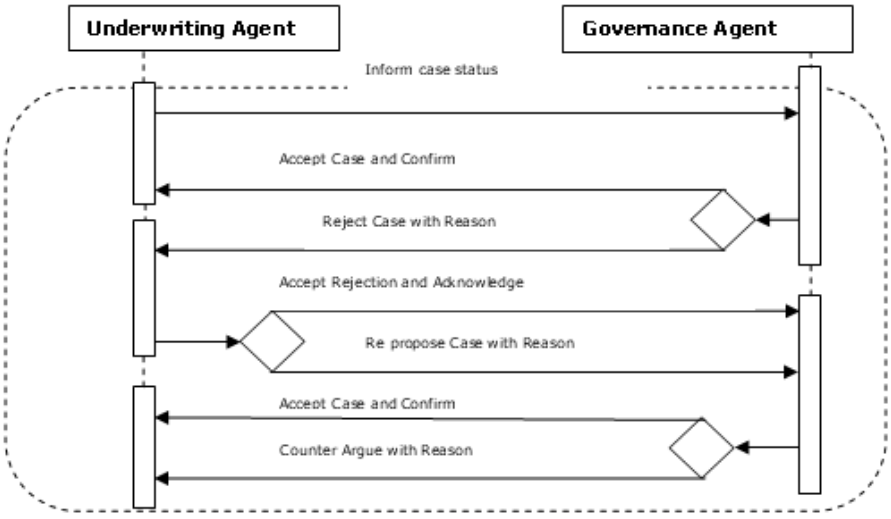

Figure 3.4. Communication protocol.

Table 3.1

Explanation of communication predicates and their natural language equivalence

Predicate	Natural language
Propose	Whenever an agent is required to propose something to another agent, e.g., when an agent wants to propose checks for a case
Inform	When an agent informs another agent about the status of the case
Request	If one agent requires additional information
Agree	When an agent agrees with another agent's argument
Refuse	When an agent disagrees with another agent's argument
Failure	When an agent wants to inform others about a failure

Agents use the predicates given in Table 3.1 to achieve the communication among them. The natural language equivalences of these predicates are also given.

This system proposes dialogues and persuasion dialogues between two agents using defeasible logic programs as a knowledge base, together with an algorithm defining how this dialogue is managed (Figure 3.5).

Agent A, involved in a dialogue with another Agent B, needs to produce an argument to refute the last argument of the opponent. To do this, Agent A must use its knowledge base KBa and be able to use some rules shown in the dialogue by Agent B. The general outline of any dialoguing by DeLP-based agents is shown in Figure 3.5. An agent may have an internal

representation of the dialogue that is being carried out. During this dialogue, agents share their beliefs to convince each other.

Summary

This new approach gives significant importance to managing adaptability in making decisions during workflow processes and to managing workflow semantics. Adaptability is maintained by updating the knowledge base and sharing the knowledge between agents. Its also has the provision to broadcast important knowledge to all participants in case of a breakthrough. This affects the workflow scenario at a greater level to improve efficiency. In addition, the runtime binding of the workflow semantics also helps the system to eliminate unwanted communication. The distributed nature of the framework does not limit the communication path for any agents. Ultimately, this approach should be the way forward for the new WMSs.

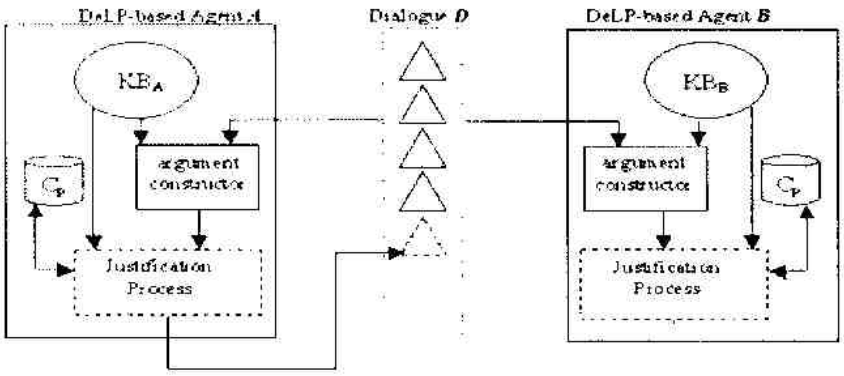

Figure 3.5. Dialoguing agents (Martinez & Garcia, 2002).

Chapter 4: Implementation

Nowadays, financial industries are one of the dominant fields where workflow systems are being used, and the availability of industrial experts gives us confidence to build the skeleton application for this domain.

General Architecture of the System

Figure 4.1 depicts the architecture of the implemented system. The architecture of the system can be divided into three core elements.

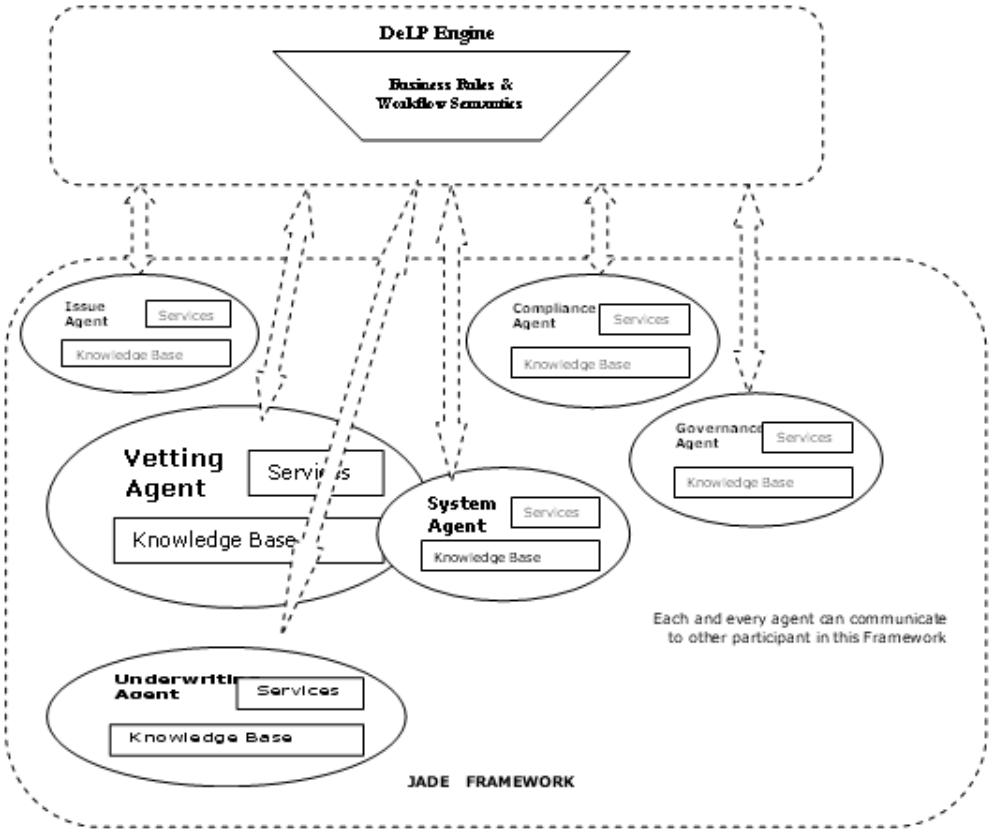

Figure 4.1. Workflow coordination for insurance agent society.

DeLP engine. This is a critical component of the system that helps the agent to query about knowledge and infer various beliefs about the environment. This engine helps the agents process DeLP queries and provides answers to the agents. The DeLP engine receives a set of rules and facts as its input. Using the given facts and rules, the DeLP engine infers the queries posted by the agent and updates the agents with the current status at any given point in time.

Agents can issue two types of queries to the DeLP engine, as explained subsequently.

Workflow semantic queries. In this schema, the agents can query about the environment and decide with which agent to communicate next. This makes the system flexible enough to communicate with different agents based on an agent's beliefs at that point in time. If it is not required that a specific task be communicated to one agent, this can be avoided. This enables the system not to follow a rigid structure of the workflow and in turn helps the system to adapt to a flexible workflow structure. There are predefined defeasible rules that enable the system to infer a flexible workflow structure at runtime based on the facts available.

Decision-making queries. Knowledge and facts available to different agents about a case in a workflow scenario are different. In all situations,

these facts and beliefs are not necessarily required to be shared. However, in decision making, these facts play a critical role. In a distributed workflow system, based on knowledge, agents can communicate, share, and convince other agents of critical decision-making facts. One unique feature of this processing model is that all the agents are not required to share all information all the time; sharing is required only when there is a conflict of interest while making a decision. There is predefined defeasible logic available to each agent based on the business rules. In the light of the facts available, an agent can infer the mismatching facts and communicate to another agent regarding decision making.

Workflow agents. The core difference of this system is its distributed processing model. In a WMS, each agent plays an individual role to complete a common goal. In the traditional model, all the agents' tasks are coordinated and monitored by a central unit. This is a time-consuming and inefficient way of getting things done as each agent is required to communicate with the central unit, and this unit is responsible for rerouting the task and getting any extra information required. This causes a bottleneck in the centralized methodology and in turn makes the process slow and time consuming. In a distributed model, each agent is responsible for its own work, and it is the individual agent's responsibility to get the information required, reroute, and communicate with other agents involved in completing

the task. In this particular domain, the following agents are there for demonstrating the capability of the system:

- *vetting agent.* This is the first agent to receive the case. The vetting agent is responsible for checking and vetting the case. If the case is qualified for acceptance, the vetting agent queries the DeLP engine with the facts available to formulate the workflow semantics and structure. On the basis of the workflow structure, the vetting agent then communicates to other agents involved with the case. Once the case is acknowledged by other agents, the vetting agent removes the case from its queue.
- *underwriting agent.* This agent is responsible for underwriting benefits for the client. On the basis of the available facts and its understanding of the previous history of the client, the underwriting agent accepts or rejects the case. Once the underwriting agent rejects the case initially, it informs the governing agent so as to confirm the process, and subsequently, after confirming the rejection status, the underwriting agent communicates to the other agents involved about the current status in order for them to be aware about the situation, which in turn makes other agents stop working on a case that is rejected by the underwriting agent. If the case is accepted, the underwriting agent also communicates to the governing agent for

confirmation and then informs the issuing agent about the status. This completes the underwriting process.

- *system agent.* This agent is responsible for inputting the details of the system. On the basis of the availability of resources, this agent inputs the details of the system and acknowledges to the issue agent the completion status.
- *compliance agent.* This agent checks a case for compliance: whether the case agrees with the terms of an insurance policy. This checking involves anti–money laundering, proof of address, origin of wealth, and so on. If a finding is doubtful, this agent refers the case to the governance agent for a detailed analysis of the case. If accepted, the agent acknowledges the issuing agent, and if rejected, it acknowledges other agents.
- *governance agent.* This agent is the main monitoring agent. It monitors all the cases referred to underwriting and compliance for governance clearance. Governance has the authority to forfeit the decision of any other agent and reject the proposal. This agent acknowledges the status of the checks to all agents involved and completes the process.
- *issuance agent.* This agent gets updated information from all the agents involved, and once it receives an accepted status from all the

agents , this agent issues a policy document to the client; if any one of the agents informs of a rejected status, it issues a reject letter to the client.

JADE environment. In this system, all the above agents are running in a JADE environment. JADE is the framework responsible for agent communication. Java-based agents are plugged into the JADE environment to get the required system agents up and running.

Communication Protocol

The communication protocol is considered as the core part of a multiagent system. This models the interaction among the agents, i.e., what to say to whom in a particular situation. Communication protocol defines how the dialogue between the agents is controlled and managed. Especially in a dialogue-based or argumentation-based agent communication strategy, it is crucial that agents are capable of checking and confirming a particular format for communication strategy with their peer agents. To comply with an agent communication strategy, this system uses the FIPA communication protocol, which is embedded in a JADE environment. To communicate with other agents, this system uses the following strategies:

1. For controlling the workflow semantics, the system initially queries the workflow semantics from the DeLP engine and informs the participants involved.

2. For decision making and knowledge sharing, the new framework uses the strategy shown in Figure 4.2.

Case Study and Discussion

For the purpose of demonstrating the implemented model, an underwriting scenario is considered with the following rules:

uw_checkok(X) -< medical(X).
~uw_checkok(X) -< medical(X), medicalbadhistory(X).
~uw_checkok(X) -< medical(X),invaliddocuments(X).
~uw_checkok(X) -<
medical(X),invaliddocuments(X),medicalbadhistory(X).

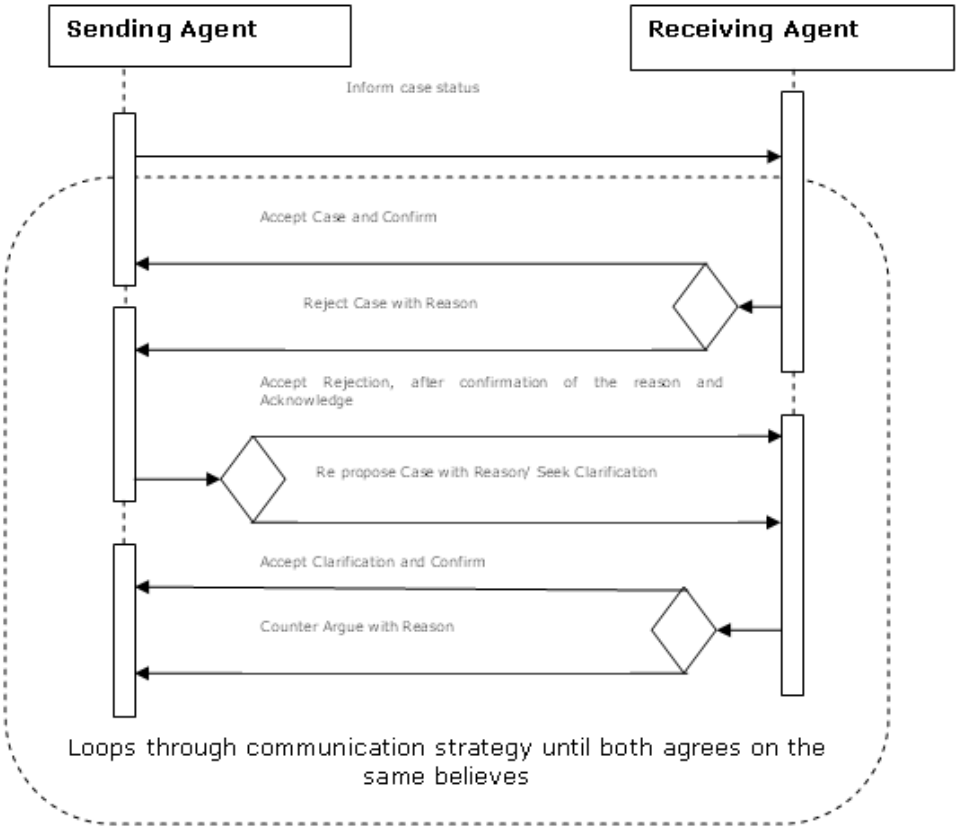

Figure 4.2. Agent communication protocol.

```
uw_checkok(X) -< medical(X), dmo_new_medicalrpt(X).
~uw_checkok(X) <-
        medical(X),dmo_new_medicalrpt(X),mngmtdecisionno(X).
cw_checkok(X)-< compliance(X).
uw_check(X) -<vetting(X).
~uw_check(X) <-vetting(X),nilbenefit(X).
medicalbadhistory(X)-<pastmedicalfailure(X).
~medicalbadhistory(X)-

        <pastmedicalfailure(X),dmo_new_medicalrpt(X).
```

Consider the scenario of a policyholder whose case was rejected in the past due to a bad medical history. This case was referred to the underwriting department for verification, which produced a new medical report from the medical officer.

However, the governance agent only knows of the past medical failure and is not aware of the new developments. At this stage, the agents are communicated the sequence in Figure 4.3 to keep the knowledge of each agent updated.

First of all, the underwriting agent initiates the dialogue with the governing agent to confirm that Case 20 has been completed successfully in underwriting, but according to the governance agent's beliefs, the case was supposed to fail its underwriting. This causes initiation of an argumentation dialogue between agents. The underwriting agent receives the reasons for the governance agent's finding, i.e., that this case failed its underwriting due to

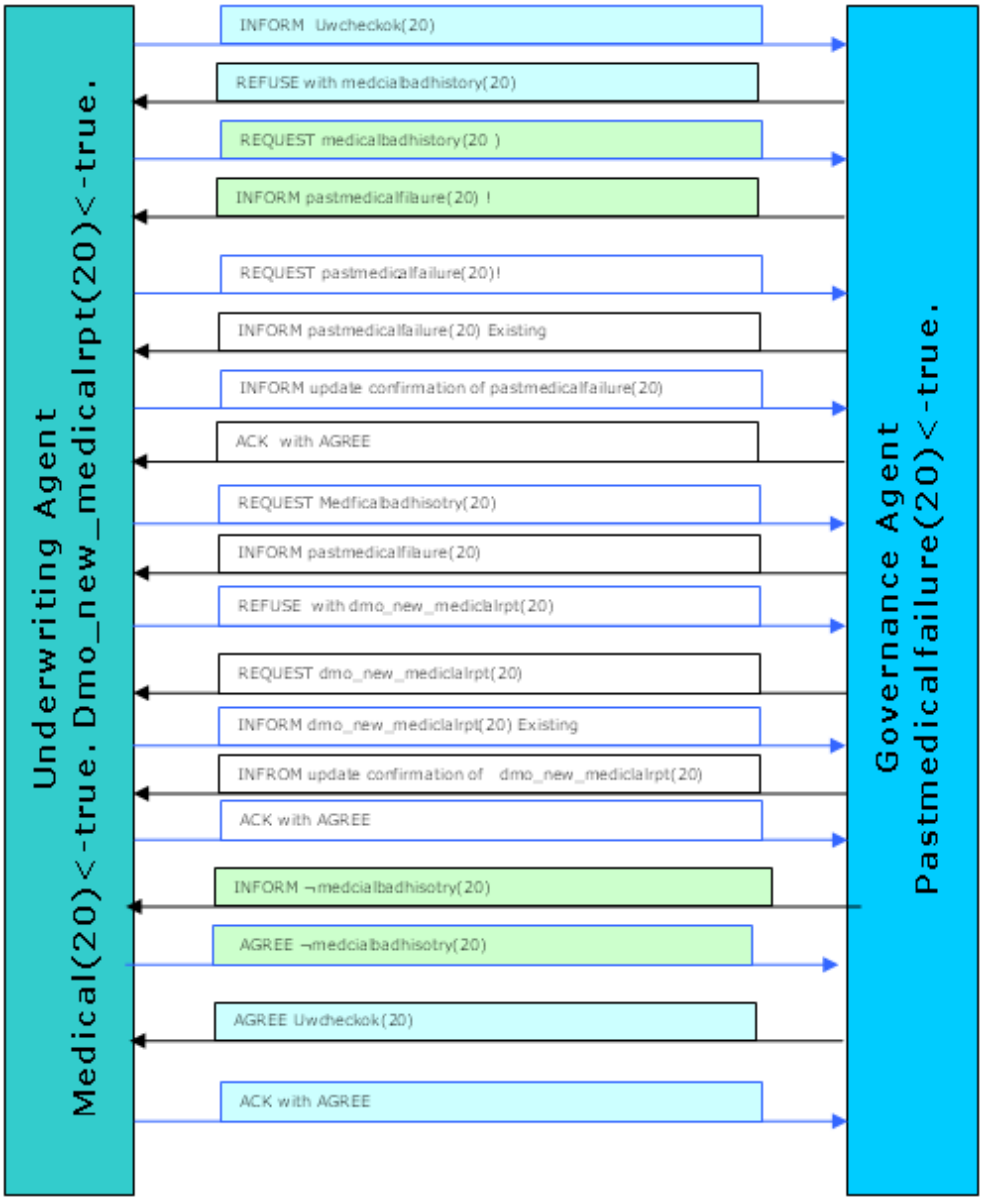

Figure 4.3. Communication sequence.

the facts of a bad medical history. However, the underwriting agent questions the governing agent about the given fact, as it is not aware of the situation.

The governance agent again updates the underwriting agent as to why this fact is true; that is, medicalbadhistory is true because there is a given fact of past medical failure. In this situation, the underwriting agent acknowledges the given fact and updates its belief. It checks with this given belief and, according to the business rule, whether medicalbadhistory is true or false. But the underwriting agent realizes still that medicalbadhistory is false and informs the governance agent. This initiates the second argumentation dialogue between the agents. At this stage, the underwriting agent provides its beliefs. Once the governance agent receives the given dmo_new_medicalrpt fact, it updates its knowledge base and requeries the case. Finally, both agents reach an agreement, and governance approves the underwriting.

In the above communication, agents share their knowledge to ensure that they meet their design objective. Finally, both agents will have a common knowledge base on Policy 20. On the basis of this knowledge, they agree on the proceedings of this case.

Using the same strategy, the compliance agent also communicates with the governance agent and completes the process. Once all agents give an OK to the issuance agent, this agent issues the case, and the case gets closed. Figures 4.4 and 4.5 illustrate the actual communications between the underwriting and governance agents.

Discussion

It has been well recognized that business process systems are important and crucial in all organizations. Workflow management is an important part of today's business process systems, but many of the current systems experience problems of poor performance, single-point failure, limited openness, and lack of adaptiveness, in addition to other problems. Furthermore, in a growing organization, interorganizational rules and regulations change very frequently, and interdepartmental communication becomes more and more complex; in such situations, it is important to have systems that can respond to the changes and adapt by changing the rules of the system. With traditional workflow management systems, it is difficult to implement such features.

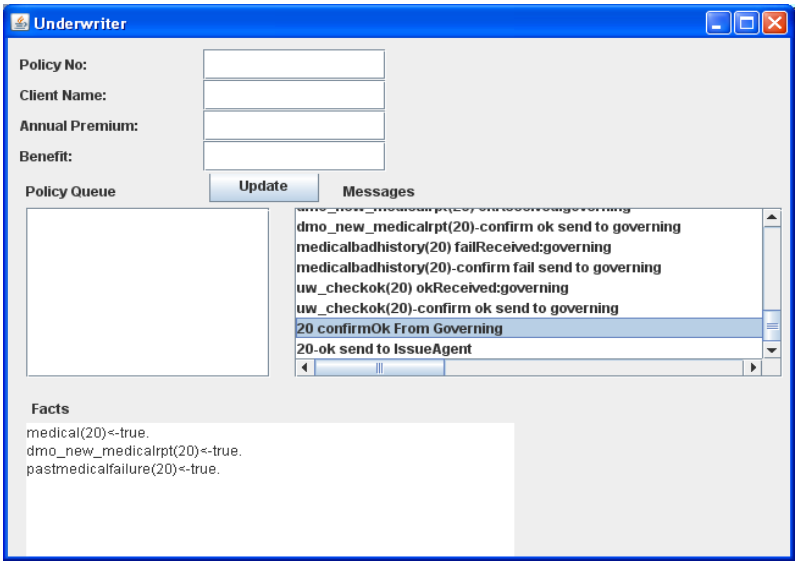

*Figure 4.4.*Underwriting agent.

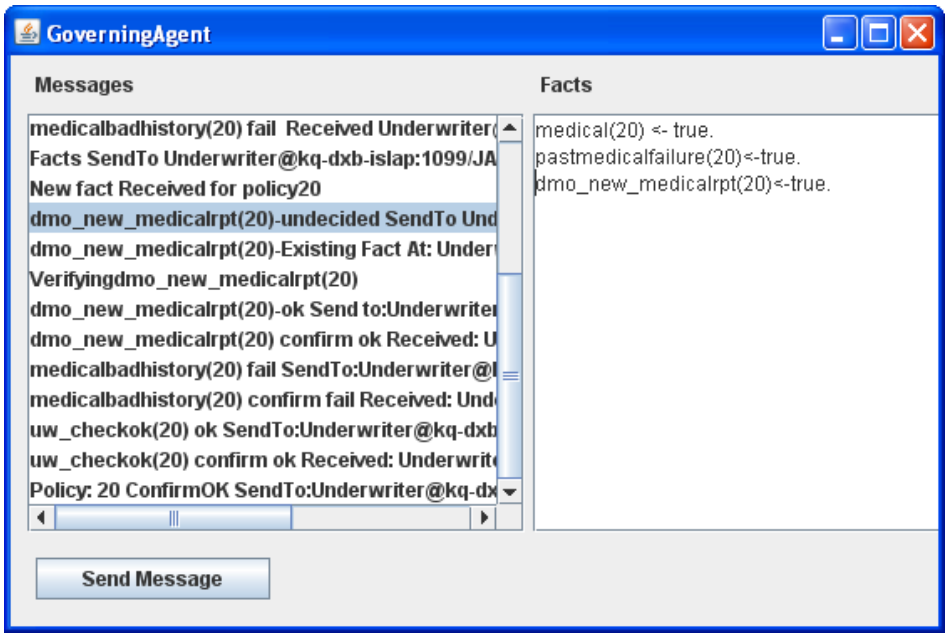

*Figure 4.5.*Governing agent.

The methodologies discussed in this dissertation address all the above described issues with a new approach. Many of the preceding issues are the result of a disparity between features required by the system or business and system implementation. Hence the centralized management method, which is not appropriate for distributed workflow management systems, must be replaced by a flexible and distributed system, while the existing business grounds and basis for developing WMSs should not be affected by the change in system architecture. On the basis of this study, the objective of this thesis is to tackle these unresolved issues by exploring new aspects of

multiagent technology enhanced with DeLP that gives a distributed design to support workflow management. Therefore a novel approach and corresponding process coordination and controlling methods are presented. The advantages of this approach are summarized in the following paragraphs.

Three different computing paradigms, namely, DeLP, WMSs, and multiagent systems, are linked. With this approach, existing multiagent-based workflow management frameworks can be constructed and used for achieving the novel framework, while increasing the efficiency of the systems.

A centralized workflow coordination server is eliminated. By eliminating the central server, single-point failures, lag time between processes, and so on are eliminated. It is more suitable in the situation where the whole system fails because some individual part of the system faulure.. The chances of single-point failure is reduced in this system since the computation and communication are better coordinated and balanced between all the agents. This approach also supports concurrent processing so that the lag time between the processes can be eliminated.

In this system, a loosely coupled computing paradigm is used for representing agents, and hence openness of the system is also improved. This method also provides flexibility in terms of workflow participants. In this

system, workflow agents are autonomous, and with the necessary information, agents are able to participate in workflow systems more actively, and the centralized workflow server does not require updates to the agents' behavior, as compared to the traditional workflow system.

This approach utilizes novel techniques involving DeLP for updating agents' knowledge and executing multiagent workflow processes. With this approach, agents are adaptive to the changing knowledge and rules. Rules can be changed dynamically, and on the basis of the new rules, the whole system process will be adaptive to the new processes. This further makes the systems more flexible and open and supports service-oriented workflow l.

Shifting from a traditional workflow framework to a multiagent-based system framework includes some trade-offs that show possible disadvantages. Some of the trade-offs of these methods are summarized in the following paragraphs.

Management and monitoring of workflow execution may become more difficult in a multiagent-based workflow management system. Additional agents (e.g., administration agents) are required for managing the administration of agents and for collecting real-time agent-related information (such as current agent state). In the case that an agent for administration has to be developed for this requirement, this will create new performance issues and introduce a new problem to tackle. However, if there

is a common workflow agent available for management and monitoring, the complexity of an individual agent can be reduced.

The capability to manage errors and exceptions is difficult within a multiagent-based workflow management system. Compared to traditional workflow management systems, in which, when erroneous situations arise, they can be proactively resolved by centralized servers, more sophisticated procedures, such as mechanisms to handle unexpected exceptions, are required. These will require future enhancement to the system.

In the future, more complex business scenarios should be taken into account, and more complex dynamic rules should be adapted; that is, agents that are adaptive and aware of context and environmental changes would be very useful for controlling, performing, and monitoring workflow management tasks.

Conclusion

In this chapter, we have explored the skeleton model developed for demonstrating the system capabilities. This is a basic working model and requires extensive modification to explore the other concept of the workflow model. However, the theoretical model has been implemented as explained and gives the expected output.

Chapter 5: Conclusion and Future Work

Summary of This Thesis

The main objective of this project is to develop an innovative distributed WMS based on DeLP using the multiagent architecture and process coordination methodologies. The thesis starts with an introduction to workflow concepts and basics workflow architectures. It also describes the motivation and outline of this thesis. Chapter 2 reviews the background of and analyzes some of the current issues in the area of traditional workflow systems in detail. On the basis of the problems analysis, it reviews a new approach to tackle the existing problems. Chapter 3 proposes a framework using DeLP and explains theoretical concepts of the proposed framework. Chapter 4 discusses the implementation aspect of the system particular to a specific domain and analyzes case studies to understand the working methodologies of the implemented system.

Thesis Contributions

The significance of this research is that it introduces a new direction for tackling some of the unsolved problems in traditional workflow and also provides new methodologies for improving the adaptability of the workflow system. On the basis of existing work from the multiagent world, this research integrates multiagent-based WMS process coordination technologies for deploying workflow systems with emerging DeLP

developments, which can be considered as an architectural change. This new architecture and its related technologies explore new aspects provided by multiagent technology to better reflect the distributed nature of current workflow. This dissertation contributes to the challenging research area of both DeLP and the multiagent system that explores a new methodology in workflow management system research. The main result of this research is using a distributed, open, and multiagent framework to implement distributed workflow management systems. Therefore the novel system framework proposed in this dissertation presents an alternate and efficient method from conventional systems for defining business rules and workflow semantics. The major contributions of this thesis follow:

- the introduction of an argumentative approach for workflow management coordination
- the identification of how the adaptability of the workflow system can best be improved using a DeLP approach
- an analysis of a new approach to formalize workflow semantics and runtime binding of workflow semantics
- the implementation and demonstration of the new framework and analysis of how it contributes to flexibility and distributed workflow coordination

Future Work

This project highlights a new direction for the future of WMS coordination. In the future, further investigation into DeLP-based multiagent distributed workflow should be carried out. Future research could include further exploration into the DeLP framework and adopting agents' intelligence into the whole workflow system. In the present system, the skeleton of the agents implemented and explained are single-task-oriented agents that can only perform required tasks following the instruction in the process model/interaction protocols. Even though these agents are adaptive , this requires the use of more functionalities of intelligent agents to become more adaptive and aware of the changes in context, which will in turn help in performing workflow management and monitoring tasks. The skeleton application explained here should be utilized for developing real-world applications by extending the features of the system. Thus a more sophisticated comparison of different workflow systems can be performed. Current systems do not concern task allocation, runtime verification of the workflow instance, and other existing features in the work coordination. To arrive at a full-fledged workflow solution, more research should be carried out on all of the above issues.

References

Antoniou, G., Billington, D., Governatori, G., & Maher, M. J. (2005, October 12). *Embedding defeasible logic into logic programming*.

Bauer, T., & Dadam, P. (1999). *Efficient distributed control of enterprise-wide and cross-enterprise work*. Paper presented at the Workshop Informatik99: Enterprisewide and Cross-enterprise Workflow Management: Concepts, Systems, Applications,

BPE14WS. (2003, May). *BPE14WS v1.1 specification*. Technical report.

Bradshaw, J. (1997). An introduction to software agents. In J. Bradshaw (Ed.), *Software agents* (pp. 3–46). Cambridge, MA: MIT Press.

Buhler, P. A. (2004). *A software architecture for distributed workflow: Enactment with agents and Web services*. Unpublished research paper.

Chen, Q., Hsu, M., Dayal, U., & Griss, M. L. (2000). *Multiagent cooperation, dynamic workflow and XML for ecommerce automation*. Paper presented at the 4th International Conference on Autonomous Agents, Barcelona, Spain.

Cingil, R., Tatbul, E. N., Koksal, P., Gokkoca, E., Altinel, M., & Dogac, A. (1997, June). *Design and implementation of a distributed workflow enactment service*. Paper presented at the 2nd IFCIS Conference on Cooperative Information Systems, .

Cui, B., Odgers, Z., & Schroeder, M. (1998). *An in-service agent monitoring and analysis system*. Paper presented at the 11th IEEE International Conference on Tools With Artificial Intelligence, Chicago, IL.

Eder, J., & Panagos, E. (1999, February). *Towards distributed workflow process management*. Paper presented at the Workshop on Cross-Organisational Workflow Management and Coordination.

Foundation for Intelligent Physical Agents. (2001, August 10). *Communicative act library specification* (Tech. Rep. XC00037H). . Retrieved May 5th, 2007, from http://www.fipa.org

Garcia, A. J., Rotstein, N. D., & Simari, G. R. (2007). Dialectical explanations in defeasible argumentation, *LNAI, 4724*, 295–307.

Garcia, A. J., & Simari, G. R. (2004). Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming, 4*, 95–138.

Jablonski, S., Neeb, J., Stein, K., Heinl, P., Horn, S., & Teschke, M. (1999, February). *A comprehensive approach to flexibility in workflow management systems*. Paper presented at the International Joint Conference on Work Activities Coordination and Collaboration.

Jennings, N. R., Faratin, P., Norman, T. J., O'Brien, P., & Odgers, B. (2000). Autonomous agents for business process management. *Applied Artificial Intelligence, 14*, 145–189.

Joeris, G. (2000). Decentralized and flexible workflow enactment based on task coordination agents. In *2nd int'l. bi-conference workshop on agent-oriented information systems* (pp. 41–62). Berlin: iCue.

Lerner, B. S., McCall, E. K., Osterweil, L. J., Wise, A., Cass, A. G., & Sutton, S. M. (2000, September). *Using little-jil to coordinate agents in software engineering*. Paper presented at the Automated Software Engineering Conference.

Li, G. (2006). *Enacting a decentralized workflow management system on a multi-agent platform*. Unpublished research paper.

Martinez, D. C., & Garcia, A. (2002). Dialoguing DeLP-based agents. 8th Argentina computation congress , 2002.

Mohan, C. (1998). *Workflow management in the internet age, advances in databases and information systems*. Paper presented at the 2nd East-European Symposium on Advances in Databases and Information Systems.

Muehlen, M. Z., & Rosemann, M. (2000). *Workflow-based process monitoring and controlling—Technical and organizational issues*. Paper presented at the 33rd Hawaii International Conference on System Sciences, Maui.

Nissen, M. E. (2000). *Supply chain process and agent design for e-commerce*. Paper presented at the 33rd Hawaii International Conference on System Sciences..

OWL. (2001). *OWL-S 1.0 release*. Technical report.

Parsons, S., & McBurney, P. (2003). Argumentation-based communication between agents. *Communication in Multiagent Systems, 2650*.

Parsons, S., McBurney, P., & Wooldridge, M. (2003). *The mechanics of some formal inter-agent dialogue*. .

Rinderle, S., Reichert, M., & Dadam, P. (2003). Adept workflow management system: Flexible support for enterprise-wide business processes (tool presentation). *International Conference on Business Process Management, 2678*, 371–379.

Schmidt, M. T. (1999). The evolution of workflow standards. *IEEE Concurrency,* 44–52.

Shoham, Y. (1997). An overview of agent-oriented programming. In J. Bradshaw (Ed.), *Software agents* (pp. 271–290). Cambridge, MA: MIT Press.

Stormer, H. (2001). *AWA—A flexible agent-workflow system*. Paper presented at the Workshop on Agent-Based Approaches to B2B at the 5th International Conference on Autonomous Agents, Montreal, Canada.

Sycara, K. P. (1998). Multiagent systems. *AI Magazine, 19*, 79–92.

van der Aalst, W. M. P., Basten, T., Verbeek, H. M. W., Verkoulen, P. A. C., & Voorhoeve, M. (1999). Adaptive work-flow: An approach based on inheritance. In M. Ibrahim and B. Drabble (Eds.), *IJCAI'99 workshop on intelligent workflow and process management: The new frontier for AI in business* (pp. 36–45)

van der Aalst, W. M. P., Hofstede, A. H. M. T., Kiepuszewski, B., & Barros, A. P. (2002). *Work-flow patterns* (technical report). Queensland University of Technology, Brisbane, Australia. Retrieved Dec 23rd 2006, from http://tmitwww.tm.tue.nl/research/patterns/

Wang, M., & Wang, H. (2002). *Intelligent agent supported flexible workflow monitoring system*. Paper presented at the Advanced Information Systems Engineering 14th International Conference, Toronto, Canada.

Weissenfels, J., Dittrich, A. K., Muth, P., Wodtke, D., & Weikum, G. (1998). From centralised workflow specification to distributed workflow execution. *Intelligent Information Systems,* 159–184.

Wooldridge, M. J. (1999). Intelligent agents. In G. Weiss (Ed.), *Multiagent systems* (pp. 27–77). Cambridge, MA: MIT Press.

Wooldridge, M. (2001). *An introduction to multiagent systems*. Hoboken, NJ: John Wiley.

Workflow Management Coalition. (1999). *Terminology and glossary* (Doc. no. WFMC-TC-1011). Retrieved May ,2007, from http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

Yang, Y. (2000). An architecture and the related mechanisms for webbased global cooperative teamwork support. *Journal of Computing and Informatics,* 13–19.

Yang, Y. (2002). Enabling cost-effective light-weight disconnected workflow for web-based teamwork support. *Journal of Applied Systems Studies, 3.*