A COMPREHENSIVE SERVICE MANAGEMENT MIDDLEWARE FOR AUTONOMIC MANAGEMENT OF COMPOSITE WEB SERVICES-BASED PROCESSES

by

FARHANA H. ZULKERNINE

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
April 20, 2009

Abstract

Web services are autonomic software applications that provide specific services on the Web and are accessible through standards-based protocols and interfaces in order to ensure interoperability. Web services have gained immense popularity due to the potential of dynamically composing multiple Web services over the Internet into complex multi-organizational Business-to-Business and Business-to-Consumer processes. The management of such composite processes, however, poses a non-trivial problem in terms of cost and complexity due to technology growth, increasing consumer demands for service quality, and the varying Internet workload.

Based on a study of the state-of-the-art and a critical assessment of the limitations of the existing solutions, we present the Comprehensive Service Management Middleware (CSMM) framework to facilitate execution of the four major tasks of client-side process management namely, service selection, negotiation of Service Level Agreement (SLA), composition and execution of the process, and monitoring and validation of SLAs. We also propose the Negotiation Broker (NB) framework for automated intelligent agent-based negotiation of Service Level Agreements (SLAs), and the Performance Monitor (PM) framework for distributed client-side monitoring and verification of SLAs.

The NB expedites bilateral bargaining of SLAs in a trusted broker framework with enhanced decision algorithms to enable consumer feedback during negotiation. The PM presents a flexible and extensible trusted monitoring solution, which enables faster error detection and recovery and automatic creation of a reputation knowledge base.

We explain a scenario of autonomic process management using the CSMM. We describe experiments using agent simulations on a prototype of the NB to validate our proposed policy model for business level specification of negotiation preferences, the mathematical policy mapping model, and the decision algorithms for different consumer preferences. The optimality of the negotiation results are illustrated by combined utility value of the negotiation outcomes for both parties. The experiments conducted on the proof of concept prototype of the PM show its viability, efficiency, and accuracy in distributed SLA monitoring and verification because it does not include network performance. The CSMM enables partial or complete automation of all the client-side management tasks to leverage use of Web services in business processes.

Co-Authors

Zulkernine, F., Martin, P., Craddock[1], C., and Wilson[2], K., 2008. "A Policy-based Middleware for Web Services SLA Negotiation". In *Proceedings of the IEEE International Conference on Web Services (ICWS'08),* IEEE CS Press, Sep. 2008, Beijing, China, *is based on Chapter 4.*

Zulkernine, F., Martin, P., and Wilson, K., 2008. "A Middleware Solution to Monitoring Composite Web Services-based Processes". In *Proceedings of 2008 IEEE Congress on Services (SERVICES'08) Part II at the Workshop on Service Intelligence and Computing (SIC) of the IEEE International Conference on Web Services (ICWS'08),* IEEE CS Press, Sep. 2008, Beijing, China, *is based on Chapter 5.*

Zulkernine, F., and Martin, P., 2007. "Conceptual Framework for a Comprehensive Service Management Middleware" (Best Paper Award for SOCNE), In *Proceedings of the 2nd International IEEE Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE'07)* in conjunction

[1] Chris Craddock was the Senior Vice President research of Computer Associates (CA) Inc. and our industry collaborator.
[2] Kirk Wilson is a research staff member of (CA) Inc. and our industry partner.

with IEEE Advanced Information on Networking and Applications (AINA'07), vol. 1, pp. 995-1000, Niagara Falls, Canada, May 2007, *is based on Chapter 3.*

Zulkernine, F., and Martin, P., 2009. "Web Services Management: Towards Efficient Web Data Access", in *Selected Readings on Information Technology Management: Contemporary Issues*, (Eds.) Kelley G., Information Science Reference, IGI Global, PA, USA, and *Web Data Management Practices: Emerging Techniques and Technologies*, (Eds.) Vakali, A. and Pallis, G., pp. 266-288, Idea Group of Publishing, PA, USA, *is based on mainly the depth paper and partially on Chapter 2.*

Zulkernine, F., Martin, P., and Powley[3], W., 2009. "Autonomic Management of Networked Web Services-based Processes", (Eds.) Denko, M., Yang, L, and Zhang, Y., Accepted for inclusion in *Autonomic Computing and Networking*, to be published by Springer, USA, *is based on Chapter 1 and 2.*

[3] Wendy Powley is a research associate in the database laboratory at Queen's University.

Dedication

To my father, Chowdhury Jamal Hyder, whose dreams were my strength, and my mother, Chowdhury Nargis Hyder, whose love and selfless devotion paved the way to all my achievements.

Acknowledgement

With great respect, I acknowledge first, the guidance and support of my supervisor, Dr. Patrick Martin. I deeply appreciate the confidence he had in me, which enabled me to choose my objectives and empowered me to pursue my goals more efficiently. My sincere thanks to Dr. J. Cordy, Dr. J. Glasgow, Dr. H. Lutfiyya, Dr. H. Shatkay, and Dr. K. Brohman for their valuable comments. I would also like to convey my gratitude to Wendy Powley for her friendly support and encouragement along the way and thanks to the members of the database laboratory for being the greatest co-workers.

My sincere gratitude and thanks to Chris Craddock (previously of CA), Kirk Wilson, Serge Mankovski, and Gabby Silberman from CA Labs, for their expert advice and guidance throughout my research. I am grateful to the Ontario Centres of Excellence, Center for Communications and Information Technology, CA Labs and Queen's University for funding my research.

I am blessed to have the greatest parents and parents-in-laws whose prayers, affection, and support helped me to hold on to my aspiration at the toughest times, and whom I can never thank enough. My gratitude towards my husband, Md. Zulkernine, is more than words can say, and without his sacrifice, support, love, and inspiration I could never have reached any of my milestones. I thank my son, Wasif, and daughter, Meem, for being the most caring and understanding children in the world and I am truly proud of you.

I thank all our friends and extended family members for helping us out whenever we needed them. Finally, thanks to Allah, the all mighty, for everything.

Statement of Originality

I, Farhana H. Zulkernine, certify that the work presented in this dissertation is original unless otherwise noted. Any published (or unpublished) excerpts, ideas and/or techniques from the work of others are duly acknowledged in accordance with the standard referencing practices.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Acronyms

- AOP Aspect Oriented Programming
- API Application Programming Interface
- AWSE Autonomic Web Services Environment
- B2B Business-to-Business
- B2C Business-to-Consumers
- BPELWS Business Process Execution Language for Web Services
- CSMM Comprehensive Service Management Middleware
- DSS Decision Support System
- DF Desirability Factor
- HTTP Hyper Text Transport Protocol
- HTTPS Secure HTTP or HTTP over Secure Sockets Layer (SSL)
- NB Negotiation Broker
- NSS Negotiation Support System
- OASIS Organization for the Advancement of Structured Information Standards
- OWL Web Ontology Language
- PM Performance Monitor
- QoS Quality of Service
- SLA Service Level Agreement
- SLO Service Level Objectives
- SOA Service Oriented Architecture
- SOC Service Oriented Computing
- SOAP Simple Object Access Protocol
- UDDI Universal Description, Discovery and Integration
- XML eXtensible Markup Language
- W3C World Wide Web Consortium
- WSCI Web Services Choreography Interface
- WSDL Web Services Description Language
- WSDM Web Services Distributed Management
- WSML Web Service Management Layer
- WSMN Web Services Management Network
- WSOI Web Service Offering Infrastructure
- WSOL Web Service Offering Language

Chapter 1

Introduction

Information systems and services today proliferate around the Internet technology and the new generation of Web community. Web services offer yet another technological breakthrough in terms of inter-operability, Internet-based service provisioning, and ease of composing the fine granular autonomic services into large cross-organizational business applications [3][22]. Web services technology has evolved as a very important area of research because of its great potential for replacing Enterprise Application Integration (EAI) [106] [124] software with dynamic Business-to-Business (B2B) or Business-to-Consumer (B2C) [53] integration over the Internet. The EAI applications had large development and maintenance overhead because of the complexity in integrating software from different organizations and keeping the integration up-to-date with its component software. Web services are loosely-coupled autonomic software applications that are hosted and managed by the respective business organization and offer a standard-based interface and protocol, which allows the services to be consumed by other applications or services [99]. Thus

multiple Web services can be linked together to build business workflows[4] [51] [52] that can span multiple organizations all over the globe. Web services follow specific standards to ensure interoperability and are perfect examples of Service Oriented Computing (SOC) [31] where all applications are considered as services in a large distributed network.

The versatility of Web services comes at the cost of the complexity in managing Web services and service-based[5] composite processes [22] [24]. The unpredictable workload of the Internet, a wide variety of users, and the security and accessibility issues generally make management of Web-based systems a challenging task [49] [137] [141]. The proliferation of systems' complexity and the numerous configuration parameters make the problem worse. On top of that, the complexity in provisioning and composing Web services to maintain satisfactory quality in the performance of the business processes [10] makes the job of system administrators very difficult and challenging, if not impossible.

The term Quality of Service (QoS) [21] [30] [126] [131] is commonly used to express the non-functional service attributes that define the expected quality of a Web service such as reliability, response time, throughput and availability. The service provider and the service consumer can negotiate the expected QoS and lay out the terms of compensation when the required QoS is not provided in the form of a contract, which is called the Service Level Agreements (SLAs) [20] [34] [59] [127]. For increased consumer satisfaction in a progressively service-based world where service reputation plays an important role in businesses [3], it is imperative that the QoS is met both on the service provider's side and on the service consumer's side. A service consumer may use a single service or compose a chain of services to create a business process, commonly called a composite process [28] [32].

---

[4] We use the words "workflow" and "process" interchangeably throughout the thesis to avoid the phrasing of "*process* of building and managing service-based *processes*".

[5] The term "service-based" implicitly refers to "Web service-based" throughout this thesis unless otherwise specified explicitly.

Researchers are working on various aspects of Web service and service-based process management. A lot of research effort [1] [16] [35] [39] is going into better manage the server-side of the Web services so that the service providers can provide different classes of services to different group of users [34], manage the workload better [70], monitor and reconfigure the various components of the server [27], provide for better access protection, and error recovery features to ensure that the SLAs are met.

On the client[6]-side there are several areas that have drawn the attention of the research community. Different approaches to service selection have been proposed [126] based on either the QoS goal of the overall process or the QoS and reputation of the individual services. A large effort is going into enabling semantic matching of the functional and non-functional properties for service selection, and formal specification of the composition of services for monitoring and verification purposes. Negotiation of SLAs is another important area of research [30] [48], which is closely associated with the research on monitoring and verification of the SLAs both on the client and the server-side.

We present in this dissertation a novel middleware framework, the Comprehensive Service Management Middleware (CSMM) [17] [138] for client-side autonomic management of Web service-based processes, and thereby, leverage the use of Web services in building dynamic business processes. We also present the Performance Monitor (PM) [139] middleware and the Negotiation Broker (NB) [140] middleware, two of the four main modules of the CSMM. The PM provides broker services for automated client-side distributed monitoring and validation of the Service Level Agreements (SLAs) of the component Web services of a composite process. The NB provides services for automated localized negotiation of SLAs between the service consumer and the service provider.

[6] The words "client" and "consumer" hold the same meaning throughout this dissertation, which indicate the individual, party or application that invokes a service.

We provide a simple example of service composition in this chapter and a more elaborate example in the next chapter that demonstrate the usefulness and applications of service compositions. Our proposed frameworks can serve a wide range of consumers who prefer the agility to build their own service-based composite processes and execute them. However, third party service providers who provide services such as, making travel arrangements, online purchase of commodities, and customer relationship management (CRM) for specific organizations, can apply the framework more effectively to create and manage personalized composite processes to meet specific requirements of different category of customers based on their context and process requirements.

1.1 Motivation

Systems management is traditionally defined as the administration of distributed systems and involves functions such as fault management, configuration management, performance management, security and accounting. The move to Service Oriented Architectures (SOA), and specifically to Web service-based applications, is forcing a re-evaluation of this definition of management. Applications can now be defined at runtime through the composition of services, and this dynamic property of the workload means that services must be adaptable [39].

We can consider a simple example of a process to create a monthly sales report, which is illustrated in Figure 1.1. The process retrieves summary data from two departmental databases and then builds a report. Each database is accessible through a Web service that allows data retrieval. In order to implement the monthly sales report process the client must identify the Web services providing the data, negotiate Service Level Agreements (SLAs) with each service, and compose and execute a workflow to produce the report. At the same time, each Web service must verify the ability of the client to retrieve the

desired data, negotiate its SLA with the client, and then execute its part of the process while monitoring performance to ensure that its SLA is satisfied. SLAs are contractual agreements between the service provider and the service consumer, which outline the expected Quality of Service (QoS), and are important to guarantee consumer satisfaction in business transactions.

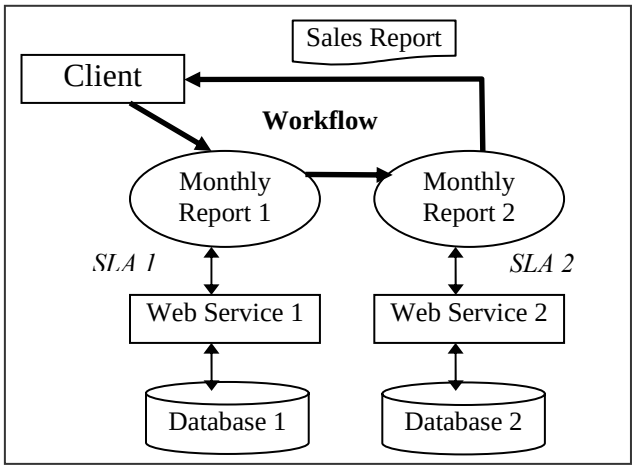

Figure 1.1 Monthly Sales Report Process [141]

On demand compositions of Web services to build business processes greatly reduces the development and maintenance cost of traditional Enterprise Application Integration (EAI) [106] software while allowing task outsourcing, and using the most up-to-date service available at the time of service invocation. The example shown in Figure 1.1 represents a composite Web service-based process, where the process of generating a sales report comprises two Web services that are invoked sequentially to provide a convenient and coherent on-demand Web-accessible software solution. This agility however, comes at the added cost and complexity of systems management both on the service consumer and the service provider's sides.

Management of Web services systems on the client-side and on the server-side pose different challenges and need to provide different kinds of support to the administrators. Therefore, we consider management from two perspectives. *Server-side management*, on the one hand, focuses on ensuring proper execution

and QoS by the service provider based on a set of pre-negotiated SLAs. *Client-side management,* on the other hand, focuses on ensuring QoS of the Web services-based process by supporting proper selection of services for the process [60], negotiation of SLAs [140] with each of the component services to meet the overall process QoS [127], definition and execution of a workflow composed of the selected services [129], and finally monitoring the workflow to verify that the SLAs are satisfied [110].

1.1.1. Research Trail

In an effort to address the increasing problem of systems management, we first looked at the server-side management of Web services with a goal to meet the agreed SLAs. Server-side management primarily focuses on service provisioning, resource distribution, workload management, and monitoring the QoS of the Web service on the provider's side. Due to the numerous configuration parameters of large systems, particularly the Web-based systems that require more frequent tuning to adapt to the varying workload, manual administration has become inefficient and more error-prone. Autonomic Computing [39] [46] [65] has emerged as a solution for dealing with the increasing complexity of managing and tuning computing environments. Computing systems that feature the following four characteristics are referred to as Autonomic Systems:

- **Self-configuring** - Define themselves on-the fly to adapt to a dynamically changing environment.
- **Self-healing** - Identify and fix the failed components without introducing apparent disruption.
- **Self-optimizing** - Achieve optimal performance by self-monitoring and self-tuning resources.

- **Self-protecting** - Protect themselves from attacks by managing user access, detecting intrusions and providing recovery capabilities.

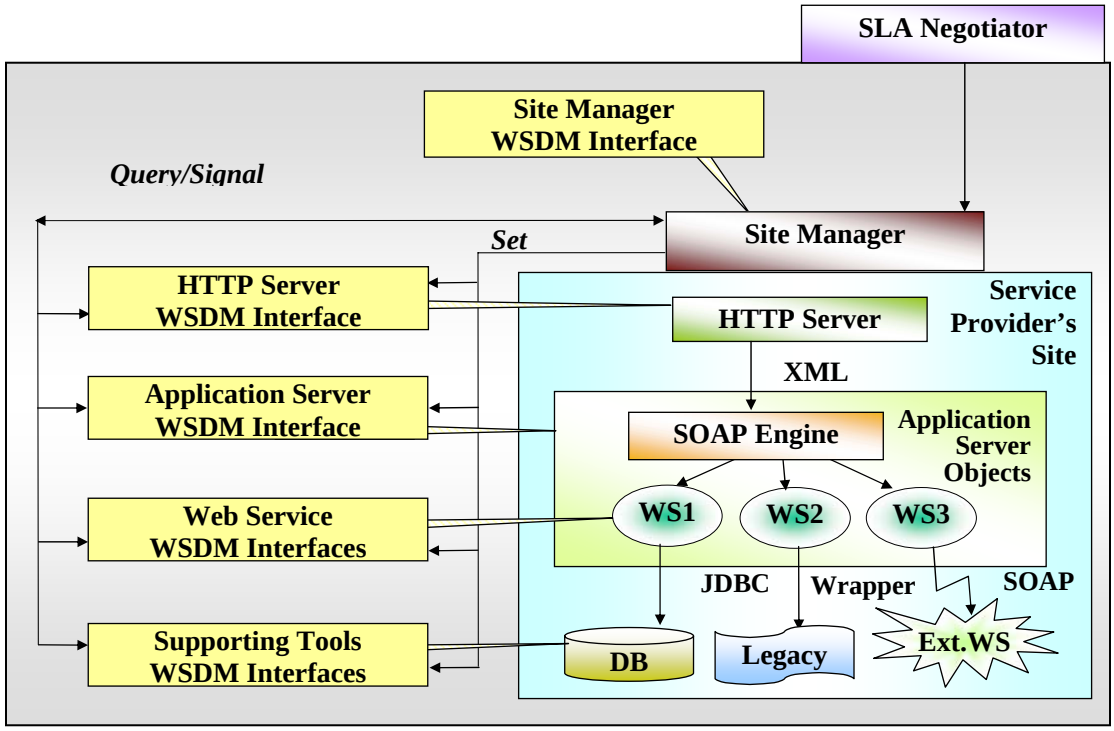

Figure 1.2 Autonomic Web Services Environment Framework [142]

We deem autonomic computing to be an efficient and viable approach to systems management. As a result of our initial group research effort, we proposed an Autonomic Web Service Environment (AWSE) [112] as shown in Figure 1.2. We also developed a prototype to illustrate the design of an autonomic element, the core building block of the AWSE [142]. The framework consists of a hierarchy of autonomic elements, each of which tunes itself at the lower level and communicates with a higher level management element to deliver performance reports and get updated performance goals. The topmost management element in the AWSE framework, the Site Manager, gets the overall performance goals for the Web service from the SLAs. A SLA negotiator component negotiates the SLAs with the service consumer. Our research on

client-side process management initiated as we tried to define the negotiation counterpart for the service consumer.

1.1.2. Client-side Management Problems

The primary motivation behind the research on client-side process management is founded on several observations. First, most of the current research efforts are directed towards one of the four major aspects of the client-side process management namely: service selection, SLA negotiation, workflow composition and execution, and monitoring and management of business processes.

Second, often the solutions are dependent on a specific system setup and architectural backbone, and therefore, are not flexible enough to be used with other approaches that address a different client-side management aspect.

Third, some of the approaches address more than one of the management aspects but do not provide the flexibility to choose a specific one. This incurs additional overhead for the other management aspects when a consumer needs management support for a single aspect, such as SLA negotiation.

Fourth, many of the proposed client-side management solutions require installation of tools on the consumer's end, which may not be cost-effective for small or one-time service consumers.

Fifth, there is no automated solution which will take consumer requirements and automatically create, execute and manage the workflow. Although the complexity of all management tasks vary depending on the granularity of the Web services and the orchestration of the workflow, the above observations inspired us to look into a more practical, comprehensive yet flexible approach to client-side Web service-based process management, which is the focus of our current research as presented in the dissertation.

1.1.3. Research Objectives

The research aims at achieving the following objectives.

- Categorize the various tasks that the service consumers need to carry out for building and executing Web services-based processes.
- Provide a general framework for complete autonomic client-side service-based process management, i.e., for facilitating all of the above tasks.
- Propose a solution to automate some of the tasks, and at the same time, reduce footprints[7] on the consumer side using outsourcing with Service Oriented Computing (SOC) approaches.
- The approach should maintain transparency and user control by allowing consumers to select specific tasks to be automated or to request a completely automated solution to their given problems.
- The solutions should be based on open standards to be flexible and adaptable to the existing solutions.
- The proposed solutions should address the drawbacks and weaknesses of the existing solutions and contribute to the state-of-the-art research in the corresponding areas pertaining to the problems of Web service-based process management.

1.2 Thesis Statement

We address the critical problem of the rising complexity in managing Web service-based processes. Based on our study of the four major tasks of process management namely, service selection; SLA negotiation; workflow composition and execution management, we propose a novel framework called the Comprehensive Service Management Middleware (CSMM), to enable autonomic process management. Each of the tasks mentioned above covers a critical

[7] Software installation on consumer machine

research area of Web services that presents challenging research problems and is carried out by a module within the CSMM. The implementation of the complete framework, therefore, will have to follow the implementation of the individual modules within the framework.

The CSMM is modular and based on open standards, and therefore, allows more flexibility and adaptability than the existing approaches to process management. The main concept behind the design of the CSMM is that each of the management tasks can be outsourced and made available as a service from a trusted service provider. As such the individual modules within the CSMM can be invoked as a service independent of the other modules. Together the modules within the CSMM allow the consumer to specify process requirements, and in return provide an autonomic process management service, which includes selection, negotiation, composition, execution, monitoring and error handling services to deliver a seamless execution of the desired process.

In the scope of this dissertation, we research and implement two of the modules of the CSMM, the Performance Monitor (PM) and the Negotiation Broker (NB). The PM provides trusted distributed process monitoring services, that is, given the process information and the SLAs of the component Web services, it monitors the performance of each component Web service on the server-side using message interception technique, verifies the SLAs, and reports errors to an Error Tracking and Recovery (ETR) module to initiate necessary recovery measures. It enables building of a reputation knowledge base. The open standards-based architecture, the concept of distributed monitoring, and the provisioning of third party services make the framework novel compared to the existing solutions, and enable multi-organizational service monitoring.

The NB provides trusted broker services for localized bilateral bargain of SLAs within the framework. The novelty of the research lies in the following: definition of a policy model for business level specification of consumer preferences for negotiation; definition of a mapping model that translates the

high level preferences to low level negotiation decision models; analysis of three time-based negotiation decision functions; decision algorithms including an adaptive algorithm that allows consumer feedback during the negotiation process, and finally, the design of the NB framework that enables automated intelligent agent-based negotiation.

We assume that the CSMM and its modules are trusted service providers. We conduct our experiments with example Web services in our laboratory setup. The research proposes various techniques that focus on specific management aspects and are independent of the type of the example Web services.

1.3 Scope of the Dissertation

The dissertation presents our research on client-side Web services-based process management and proposes the CSMM framework as a viable approach to autonomic process management. We extend two of the four main modules of the CSMM, the Performance Monitor and the Negotiation Broker, within the scope of this thesis. We provide a literature study, propose the two broker middleware frameworks, and illustrate the viability of our approach through experimental study.

1.4 Contributions

We address the client-side Web services-based process management problems in our research. Our main research contributions are described below.

- We define the CSMM framework for comprehensive autonomic management of Web services-based processes. The framework applies principles from autonomic computing, distributed management, and Web services technology paradigms to enable outsourcing of the management tasks to the different modules of the CSMM framework. The modules can

either be invoked independently or as a composite CSMM service by the service consumer.

- We define a trusted broker framework for automated SLA negotiation of Web services where consumer preferences for negotiation are specified as business level policies. The framework applies theories and techniques from negotiation for the specification of the negotiation protocols [12] [44] [102], decision support strategies based on time-based decision functions [41], and intelligent agents. The NB offers a powerful feature of allowing consumers to update their policies or preferences during an ongoing negotiation based on the changed status or availability of resources.
  - We propose a policy model that shows the different entities in the policy model and their relationships. We use the WS-Policy standard for policy specification. It is very important to properly describe negotiation preferences for automated negotiation and our policy model allows consumers to specify their policies at the business level and update them as necessary. The framework includes a negotiation knowledge base. If the policy specifications are incomplete, the knowledge base can be consulted or other learning techniques can be applied to initialize the missing parameters, which is part of our future work plan.
  - We define a mapping model to translate the high level policy specification to low level decision model parameters and rules that are applied to conduct the negotiation. We propose mathematical equations to derive the numerical parameters of time-based negotiation decision functions from the information given in the policy specification. The mapping rules also enforce execution of specific decision algorithms as part of the negotiation decision model.
  - We propose an adaptive algorithm to enable dynamic update of the boundary values of the negotiable issues through redefinition of the

negotiation decision function. The algorithm has three features; first, it enables adaptive negotiation using the simple time-based decision functions; second, it allows consumers' feedback into the process before the final agreement is made based on the changing status of the consumers or their resources; third, it shows a considerable improvement in the negotiation outcome in terms of utility value, which is a measure of consumer satisfaction, when used with the time-based negotiation decision function.

- We demonstrate the strategy of conducting impartial negotiation locally within a broker middleware using intelligent agents and selection of the appropriate time-based decision function for an issue based on its consumer preference values. The NB implements three different time-based negotiation decision functions with the provision to add more negotiation strategy models in future. The performance of these functions for different preference specifications are observed and stored in the negotiation knowledge base, so that given a specific preference, the proper function can be chosen to improve the overall negotiation strategy.
- We present extensive experimental results in support of the practicality and effectiveness of our approach.

- We propose the Performance Monitor middleware framework for distributed monitoring of the SLAs of the component Web services of a composite service-based process that span multiple organizations. We use open standards to enable inter-organizational process monitoring and assume the PM to be a trusted broker. The PM allows the monitoring task to be outsourced and reduces consumers' overhead from having to setup and maintain a monitoring framework. The PM framework has two sub-systems. The secondary sub-system of the framework does the monitoring of the service performance and reports the data to the Web service end-

point[8] of the primary sub-system. This architecture allows the secondary sub-system to be replaced by any existing monitoring system on the service provider's side that is capable of reporting performance data to a Web service endpoint. Thus the PM can be used to monitor more general Web-based processes.

- The experimental results show that the data collected at the primary sub-system is reliable and more accurate than the measurements taken at the point of execution of the process. The reason is that the data collected at the PM excludes the network performance, and therefore, is better for SLA verification.

1.5 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 presents the background on Web services technology and the literature study on Web services-based process management, SLA negotiation and process monitoring. In Chapter 3, we present our CSMM framework with a case study to demonstrate the functionality of the different modules and the overall autonomic management of a Web services-based process. The NB module is described in Chapter 4, which includes a discussion of the evolution of negotiation theory; a description of the various negotiation decision functions, algorithms and our NB framework, and the experimental validation of our negotiation approach. The PM module is illustrated in Chapter 5. It includes a discussion of the various monitoring approaches, our PM framework and its experimental validation. Finally, we summarize the contributions of our research with a critical assessment in Chapter 6, discuss some of the future work directions and conclude the thesis.

[8] End-points are basically addresses that are accessible to consumers and expose certain functionality.

1.6 Summary

The complexity of composite Web services-based business processes creates many new challenges for the researchers in the area of Web services management. We address the problem of client-side composite process management, which includes the challenges in service discovery, SLA negotiation, service composition and process execution, and process monitoring. The dissertation presents three different aspects or areas of client-side Web services-based process management; first, the overall autonomic management of the process; second, the negotiation of SLAs, and third, distributed monitoring of the workflow. We start with laying out the motivation behind this research dissertation, which evolved from our group research on the AWSE framework [112][142]. Then we state our research hypotheses, the scope of our work, and the contributions of this research in the area of Web services management. In the following chapters we provide the background study, detail presentations of our CSMM, NB and PM frameworks with experimental evaluations of their methodologies and prototypes, and finally summarize and conclude the thesis outlining some of the future work directions.

Chapter 2

Background and Literature Study

Our research on client-side Web services-based process management applies theories and techniques from several areas, such as distributed systems management, autonomic computing, Web services technology and standards, and negotiation decision support systems. This chapter consists of two main sections. In the background section, we discuss some of the basic concepts of Web services, service-based processes, and commonly used standards for Web services. Our research contributes to three different aspects: first, the complete Web services-based process management; second, the negotiation of Service Level Agreements (SLAs), and third, distributed monitoring of composite processes. In the literature study section of this chapter, we discuss related work that addresses the above three aspects under three separate sub-sections. Comparison and contrasts of the closely related works with our approaches are presented later in the corresponding chapters.

2.1 Web Service

Web service is the pioneer of the current drive of converting existing software frameworks to Service Oriented Architectures (SOA) [46] where all software functionality is provided as services. The differences between traditional software and the current concept of service are prominently with respect to accessibility, proprietary rights, installation cost and complexity, granularity and interoperability. SOA lays out a framework to develop software as services and provide the services over a network that is often supported by an Enterprise Service Bus (ESB) [52]. The ESB acts as a backbone to connect services in an enterprise environment to leverage service management. Due to the existing Web infrastructure and the growing number of Internet users, services are primarily made accessible on the Web through standards-based communication protocols and interfaces, which are called Web services. The use of standards is crucial in creating Web services in order to achieve the goal of interoperability. Web services are, therefore, software applications that offer specific granular services through standard Web-based communication protocols and interfaces.

Advantages of Web services are manifold. By building Web service interfaces to existing legacy applications, these systems can now be made accessible on the Web and existing software systems can be effectively moved to SOA. The different levels of granularity in service provisioning and the interoperability achieved through the implementation of standards allow multiple Web services to be composed to create complex workflows that span multiple organizations.

2.1.1. Web Service Composition

Figure 2.1 shows an example where a consumer uses a chain of three Web services from three different organizations to execute the process of purchasing a CD on the Internet. First, the consumer places the order for a CD using the CD store's Web service. After the order is processed, the Banking Web service is

invoked to collect the payment for the order. When the payment is made, the Shipping Web service is invoked to deliver the CD to the consumer. A process or workflow that is created using a chain of Web services as the one described above is called a Composite Process [139], and the process of building such a workflow is commonly called Service Composition [38].

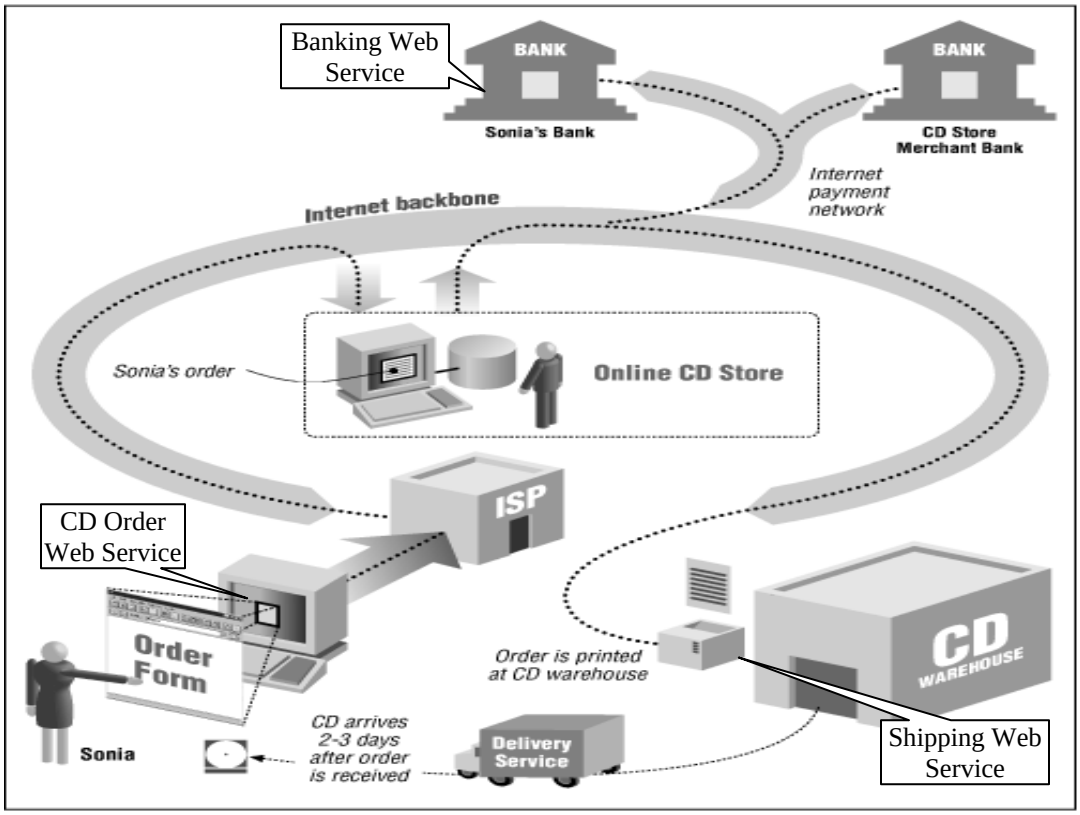

Figure 2.1 Buying a CD with credit card over the Internet [47]

The consumer invoking a service may be a user using a Web browser, a client application, or another service. Thus a composite process can contain one or more sub-processes. The information flow in such processes becomes harder to track when one service A calls another service B, which in turn calls another service C and the results do not follow the same return path, i.e., the reply from service C directly returns to service A without going through B. Depending on the composition structure, therefore, the process can grow to become very large and complex, and very hard to manage. Moreover, with respect to the binding

time of the services, i.e., when a service address is hooked to the service call structure, there are two types of service composition. If the services in a composite system are selected and bound before execution, it is called static composition [137]. On the other hand, if a process is composed of placeholders that are filled by services selected and bound during execution time, it is called a dynamic composition [137], and in that case the system is built gradually with each service call.

2.1.2. Web Services Life Cycle

The key to the success of Web service technology is its interoperability, which is achieved through numerous standards that ought to be followed to build, publicize, and use Web services. While it has become a difficult challenge to maintain the huge number of standards that exist today, many more are on the way to becoming new standards and are currently being reviewed by the two main standards committees, the Organization for the Advancement of Structured Information Standards (OASIS) [87] and the World Wide Web Consortium (W3C) [120]. The standards that are referred to later in this dissertation are compiled in the list of acronyms. We will describe some of the common standards in this section as we describe the life cycle and usage of Web services.

After developing a Web service, it is advertised on the Web in a structured directory or Yellow Pages for Web services called the Universal Description, Discovery, and Integration (UDDI) [87]. The UDDI contains information about accessibility, communication protocols, and functionality of the Web services so that consumers searching for specific services can look up in this directory, match necessary functional criteria to find the desired service, and then follow the given information to invoke the Web service. Web Services Description Language (WSDL) [120] is used to describe information about the Web services in the UDDI. The service providers can optionally publish additional non-functional information such as response time, reliability, and availability in an

extended part of the UDDI [15] [126]. The extension mechanism of the standard UDDI, proposed by researchers as the requirements for including non-functional information, is getting important for QoS-based service discovery. Some of the researchers propose storage external to the UDDI for the non-functional information [30] [80] for complexity and accessibility reasons. The UDDIs are usually hosted by large reputed organizations.

The life cycle of a Web service consists of three main stages: Service Publication, Discovery, and Invocation as shown in Figure 2.2. Service providers publish or advertise services in the UDDI where service consumers try to find or discover services that match their selection criteria. If a service is found, the corresponding WSDL specification is retrieved from the UDDI by the service consumer, which is necessary for using or invoking that service. Finally, the consumer typically uses Simple Object Access Protocol (SOAP) to send messages over the Hyper Text Transport Protocol (HTTP) [119] to invoke the selected service. The service is used either individually or as part of a process from the site where it is hosted by the service provider.

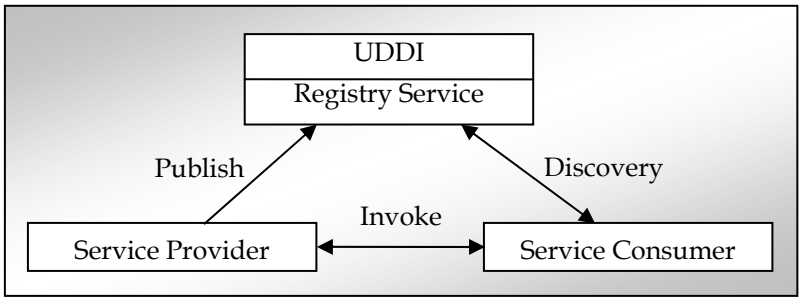

Figure 2.2 Web Service Life Cycle

2.1.3. Web Services Standards

There are many other standards for Web services. Some of the primary standards are shown in Figure 2.3 organized in a stack based on their use. Extensible Markup Language (XML) [122] is the basis of all languages that are used for communication or specifications of Web services. XML Schema

describes data types used in an XML representation. SOAP is the most common protocol used for communication and WSDL is used for the specification of Web services in the UDDI.

The layer that sits on top of the Interface layer contains multiple standards that are designed to address different aspects and are not dependent on each other but can leverage the implementation of Web services. These standards provide different levels of expressiveness and support for automated service discovery, composition, and management. We show three main categories of standards with names of only a few of the commonly used ones at this level from the many that currently exist.

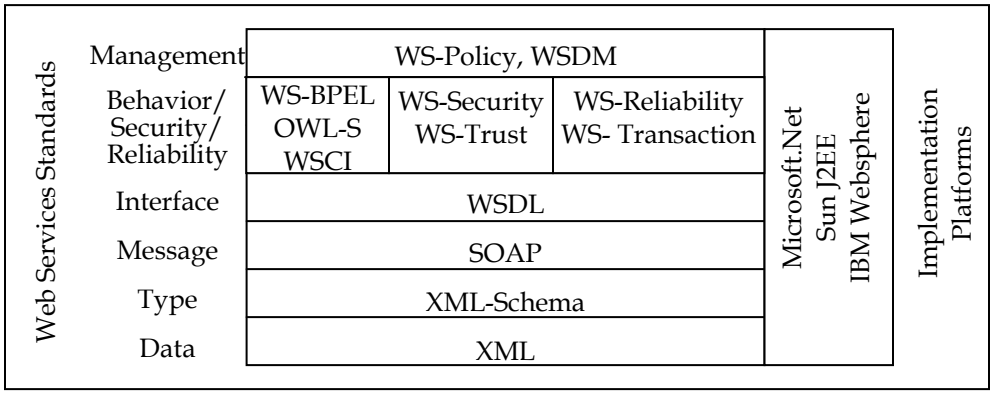

Figure 2.3 Web service standards

WS-BPEL, OWL-S and WSCI are listed in the Figure in the behavioral category. Business Process Execution Language for Web Services (WS-BPEL) [88] is used to specify the composition or orchestration of a Web services-based process, which includes the list of the interacting component Web services and a view of the control flow of the process and message interchanges among component services from a central execution point. WS-BPEL specifications can be verified using the BPEL execution engine. The Web Service Choreography Interface (WSCI) is an XML-based interface description language that describes the flow of messages exchanged by a Web Service participating in choreographed interactions with other services. Choreography provides a more

detailed view of the interactions between Web services at the message level for multiple processes, and includes the business logic and execution order of the interactions. Orchestration differs from choreography in that it describes message flows between services in a specific business process controlled by a single party.

Ontology is a hierarchical categorization and representation of concepts from a domain. OWL-S (Ontology Web Language for Services) is an OWL-based Web service ontology, which supplies Web service providers with a core set of markup-language constructs for describing the properties and capabilities of their Web services in unambiguous and computer-interpretable form [98]. The semantic annotations facilitate automated Web service discovery, execution, composition and management [67] [115], particularly in the pervasive computing [29] [77] paradigm, where semantic processing of context information and other metadata is important. A good survey on semantic Web services is provided by Zhou *et al.* [135].

The next category at this level addresses Web services security and the two listed standards are WS-Security [92] and WS-Trust [94]. WS-Security describes enhancements to SOAP messaging, such as use of XML encryptions and signatures to secure message exchanges as an alternative or extension to using HTTPS to secure the channel. It enables message integrity and confidentiality and accommodates a wide variety of security models and encryption technologies. WS-Trust defines extensions that build on WS-Security to provide a framework for requesting and issuing security tokens, and to broker trust relationships.

In the reliability category we list two standards WS-Reliability and WS-Transaction. WS-Reliability [90] is a SOAP-based protocol for exchanging SOAP messages with guaranteed delivery, no duplicates, and guaranteed message ordering. WS-Reliability is defined as SOAP header extensions and is independent of the underlying protocol. WS-Transaction [93] describes an

extensible framework for providing protocols that coordinate the actions of distributed applications. Such coordination protocols can be used to support a wide variety of applications that require consistent agreement on the outcome of distributed transactions.

The topmost layer in Figure 2.3 contains the standards for the management of Web services. The Web Services Policy Framework (WS-Policy) [121] provides a general purpose model and corresponding XML-based syntax to describe the policies for a Web Service. It defines a base set of constructs that can be used and extended by other Web services specifications to describe a broad range of service requirements and capabilities. WSDM (Web Services Distributed Management) [89] defines two sets of specifications: Management Using Web Services (MUWS) and Management Of Web Services (MOWS). The WSDM standard specifies how the manageability of a resource is made available to manageability consumers or administrators via Web services. It requires all manageable resources and their manageable properties to be accessible through a Web service endpoint called a manageability endpoint. The implementation behind manageability endpoints must be capable of retrieving and manipulating the information related to a manageable resource. MUWS defines how an IT resource is connected to a network and provides manageability interfaces to support local and remote control. MOWS builds on MUWS to address management of the Web services endpoints using WS protocols.

This section presents only a subset of the existing standards of Web services. Some of these standards consist of other standards. We will refer to some of the above standards as we describe our research in later chapters.

2.2 Management of Web Services Systems

Web services have largely evolved during the last few years from a mere standards-based Web interface to a popular tool that has a wide range of

applicability such as constructing multi-vendor workflow systems, remote resource management, and providing ubiquitous accessibility to legacy systems. It is of utmost importance for the continuing growth in a competitive market and acceptance of this technology that the service quality meets the consumer's expectations. Establishing a formal Service Level Agreement (SLA) is, therefore, crucial for guaranteeing the Quality of Service (QoS) through efficient management frameworks. In this section we give an overview of Web service systems, some of the management issues and objectives, and common management approaches.

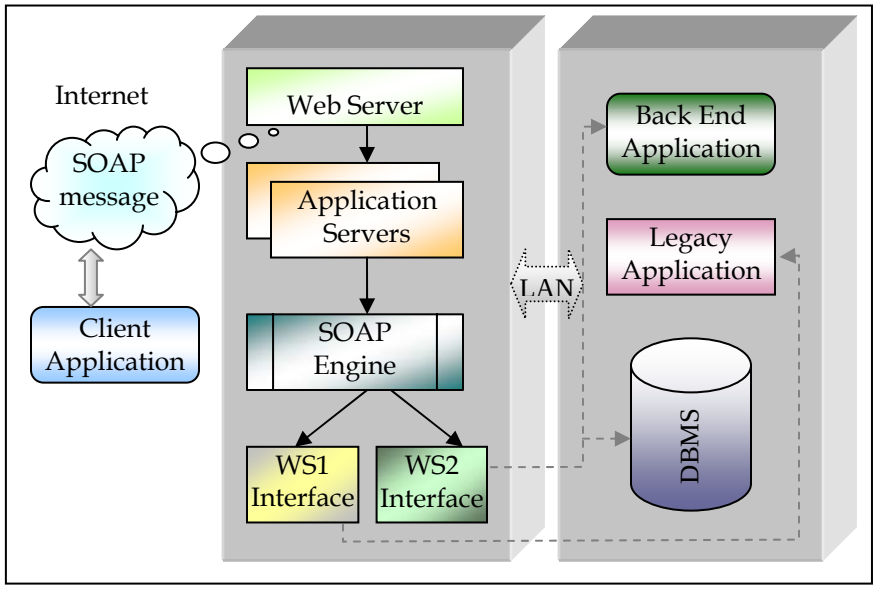

Figure 2.4 Components of a Web service hosting system

2.2.1. Components of a Web Service System

A Web service requires several essential components to be hosted on the Internet as shown in Figure 2.4. These components can reside on the same server machine or be distributed among multiple interconnected servers. Consumers typically communicate with Web services using SOAP messages, which first reach the Web servers at the site that hosts the Web services. There can be one or more instances of the application server serving as containers to host single or

multiple Web services. Messages received by the Web server are forwarded to the application servers. SOAP messages are basically XML data bounded by headers and footers containing the messaging protocol. An interpreter called the SOAP engine translates the SOAP envelope of the message, and after necessary preprocessing, passes the message to the appropriate Web service.

Web services process the messages and compose replies to send back to the client. A more complex service can require connections to backend applications, legacy or database systems. These backend applications may reside on separate servers connected to the HTTP server via a Local Area Network (LAN). Due to the dependability on other components, management of the system implies managing all the related components to ensure satisfactory performance on the server-side.

From the consumer's perspective, a Web service system is a composite process, which is composed of the component services and sub-processes. Management of Web service systems on the client-side, therefore, comprises the tasks of building and executing Web service-based processes, which includes service selection, SLA negotiation, workflow composition and execution, and monitoring and recovery.

2.2.2. Management Tasks and Complexity

On the service provider's side, management of Web service systems implies managing each component that contributes to the provisioning of the service to guarantee the SLAs. The SLAs are primarily mapped to Service Level Objectives (SLOs) at the system administration level. For example, to meet a certain response time goal the parameters of the database component need to be properly configured for a specific workload. Also the various components of the system must be managed harmoniously to meet the overall SLOs. Some of the tasks for server-side management include: breakdown of the overall SLAs to each component level SLOs, configuring the system components, tuning the

parameters as the workload changes, manage the variable Internet workload as much as possible to maintain a consistent system performance, apply techniques for speedy recovery and increased reliability, and coherent management of interdependent system components to achieve satisfactory performance. The diversity of the Internet workload, heterogeneity and interdependency of the system components, numerous configurable system parameters, and necessity to provide differential services i.e., different levels of services to different groups of consumers, make the management tasks more complex.

The client-side process management includes the tasks of building, execution and management for seamless execution of the process. The functional and non-functional service attributes advertised by the service provider are commonly known as *service offerings*. Based on the service offerings, service consumers first select the services that match the requirements. Then the service providers and the service consumers negotiate to establish a formal contract called the SLA that outlines consumers' expectations of the QoS and providers' commitments to meet those expectations. SLAs must be monitored to guarantee the QoS. Often the SLAs of the component Web services are compiled to compute a process SLA. If the parties fail to reach an agreement then consumers look for alternative services and the steps are repeated. Consumers can invoke services without establishing an SLA but in that case providers do not have any accountability.

The next step for the consumers is to compose and execute the workflow by invoking the services in certain orders with proper checks for the validity of the expected results after every invocation so that recovery measures can be taken if a failure occurs. In addition, the performance of the component Web services (availability, response time, and reliability) has to be monitored for the verification of the SLAs. The network and the various systems components between a service provider and a service consumer affects the actual service performance and make verification of SLAs on the consumer side a difficult and challenging task. If any anomaly in the SLA is detected, consumers need to take

proper measures to recover from failure in order to complete the execution of the workflow.

The complexity of the above management tasks depends on the complexity of the service orchestration. Since a Web service can be a service requester and a service provider at the same time, there can be a chain of service calls in a composite process, which can be very difficult to track. In a *complex composite process*, one process can contain a sub-process, which can again contain another process, and so on. In such cases, generally the top-most sub-process would provide the SLA, which is monitored against the overall performance of the sub-process instead of the individual component services. We discuss the challenges further in the next chapter, and present our CSMM framework for client-side process management.

2.2.3. Management Goals and Aspects

The general objective of systems management is to ensure seamless execution of its software applications and meet certain standard performance goals. Due to the increasing systems complexity, it is hard to find a universal approach from the existing research work that tries to achieve all the goals of Web service-based system and process management. Most of the researchers focus on achieving only one or more specific goals by addressing certain aspects of systems performance. We describe some of the common management goals such as, QoS, security and resource optimization, and the management aspects for each of these goals below.

QoS

The goals for QoS are usually determined from the SLAs between the service providers and the service consumers. The typical QoS goals include response time, throughput and availability. Some of the factors that make these goals hard to reach are:

- necessity to provide differential services to different categories of service consumers based on their corresponding SLAs
- heterogeneous environment of Web services
- dependability on other components, services, and the network
- versatile and variable Internet workload and
- numerous system configuration parameters

Given the above range of challenges, researchers have proposed various approaches to meeting the desired QoS goals by addressing specific management aspects as described below.

Workload management: One way to achieve a desired performance goal is to efficiently handle the unpredictable Internet workload. A sudden rise in the service request rate can degrade the system performance due to unavailability of sufficient resources. Researchers propose various workload characterization, prediction and adaptation techniques [13] [34] [70] to manage the workload. Workload characterization techniques attempt to categorize the different types of workloads to be able to provide differential SLAs. Prediction techniques typically implement a feed-forward control loop that *monitors* and *analyzes* previous workloads to predict future workload, and thereby, *plans* to *execute* necessary changes in the system beforehand to be prepared for the changing workload [13]. A feedback loop composed of similar monitor, analyze, plan and execution stages is used for reactive adaptation of the system [1]. Workload distribution applies various queuing models, priority control and capacity planning techniques [16] [27] [101] to distribute the incoming workload to the existing resources based on their capacities.

System management: This management aspect addresses collaborative management of all the components within the system to meet the overall performance objectives of the system. We categorize system management into the following three smaller management aspects.

- Resource management – Allocation of redundant resources to guard against sudden peak workloads increases installation and management cost. Therefore, techniques such as dynamic resource allocation, resource redistribution and sharing are often proposed by the researchers to make efficient use of the existing resources.

  Other popular resource management approaches are dynamic reconfiguration and parameter tuning [27]. However, practically only a few parameters can be reset dynamically. Many of the reconfiguration and tuning actions require a system reboot, which is very inconvenient for online systems. The large number of parameters in the systems today also poses a challenge to computing the right configuration settings to suit the changing workload. Statistical models with a feedback control loop are typically used to monitor and tune system parameters. The loop monitors the changing performance, analyzes the data to diagnose potential problems or necessary changes, plans what parameters need to be tuned and then executes the plan to materialize the changes in the system [132] [142].

- Distributed component management – To achieve the system level performance objectives, all the components within the system need to meet their individual performance goals. This coordination is achieved through management frameworks that apply centralized or distributed management approaches. Centralized management approaches typically have a central manager or control point, which oversees the performance of the components and sets new performance goals for individual components [28]. Distributed management approaches have multiple managers, which manage their immediate neighborhood and coordinate to control the overall system [75]. Most management approaches currently apply hybrid approaches using a hierarchy of managers where a higher level central manager collaborates with lower level regional managers [24]

[112]. Researchers in this area address the issues of mapping system level SLOs to component level SLOs, definition of management frameworks, design of the managers at different levels, and coordination protocols among the managers in the framework [34] [24] [75].

- *Recovery* – Availability and reliability are getting increasingly more important in the SOA due to dependence on third party service provisioning for critical business processes [32] [49] [129]. Effective monitoring is essential for all aspects of system management, and specially, for error detection and recovery because recovery measures cannot be taken if the error is not detected at the right time. Too much monitoring is costly and has an adverse effect on the system. Therefore, monitoring and analysis of the performance data for fault and failure detection has always been an important research topic. In distributed SOA researchers propose various approaches to address the reliability and recovery aspect that use replication; resource replacement; reliable messaging protocols; rule-based root-cause analysis for error detection; autonomic service deployment, and policy-based recovery using various monitoring frameworks [24] [35].

Process management: The four main tasks to build and execute a composite service-based process are: service selection, SLA negotiation, workflow composition and execution, and monitoring and recovery. Researchers often integrate one or more of these tasks in their proposed solutions. Management aspects for a composite service-based process, therefore, revolve around these tasks, which are described below along with the corresponding common management approaches proposed in the literature.

- *Service selection* – The commonly seen service selection approaches are: QoS-based service discovery [15] [80] with a goal to meet end-to-end process QoS [131], use of reputation systems [79], semantic matching of

the selection criteria [2] [28] [108], dynamic service selection using workflow models [21], and broker-based selection [60] [126].

- *SLA negotiation* – Negotiation is an interesting research area that has been examined by researchers from many different areas [12]. Existing negotiation theories are now being combined with Web technologies to design efficient tools for Web services' SLA negotiation [127]. Some of the current research in this area include Web-based Negotiation Support Systems (NSS) [36] [71]; semi-automated and automated negotiation tools; application of various negotiation theories (e.g. game theory [43], business models [41], genetic algorithms [12], and artificial intelligence learning techniques [66] [85]); adaptation of bilateral bargaining or multi-party bidding for SLA negotiation [30] [130]; decomposition of process SLA to service SLA for service selection and negotiation to meet a specific process SLA [25]; negotiation languages [26] [125], and broker-based negotiation [109].
- *Workflow composition and execution* – In the literature, composition often represents service selection and composition [2] [38] [103] and focuses on topics such as analysis of process execution engines (ex. WS-BPEL and BPEL4People) [88] [101]; expression of message interchanges among the component services in the process for monitoring and fault and failure detection [129]; redundancy free service composition [3]; semantic service composition [28]; formal specification of composition for model checking [5], and composition models for dynamic service selection [116] [133].
- *Monitoring and recovery* – Monitoring has been extensively explored by the software industry for their proprietary intra-organizational management application suites [19] [23] [55]. Researchers have also proposed different instrumentation techniques for collecting performance monitoring data. However, inter-organizational process monitoring and recovery still

require further research. Current research in monitoring is mostly geared towards the following: message tracking in inter-organizational processes [104]; message interception and in-code instrumentation techniques [57] [105]; network of intermediaries and other middleware frameworks for distributed service monitoring [75]; policy or rule-based error detection [114] [132]; SLA validation [59]; fault and root-cause analysis [115], and dynamic service replacement for automated recovery [136].

Security

Security is an important issue for the Web users to guard against anomalous intrusions [47]. Some of the approaches proposed by the researchers for security enforcement are: use of various access control techniques and encryption techniques [29] [33]; security policy specification and matching [100]; negotiation and enforcement with different hand-shaking protocols [96]; authorization and access control frameworks [50]; trust frameworks [81]; federated systems [79], and use of reputation systems and secured message passing protocols [92].

Resource Optimization

Due to increasing consumer demand for higher availability and reliability, and price reduction of some of the resources like storage and memory, this particular aspect has become less important than the others. However, there is a big drive towards grid computing that tries to achieve maximum utilization of resources. Some of the resource optimization techniques include scheduling algorithms for resource sharing and workload distribution techniques [13] [27] [70]. For Web service-based processes, resources are considered as the granular services and optimization implies selection of services to avoid redundancy or minimize service calls [2] [38]. Web services facilitate outsourcing of jobs to third party service providers, which is another popular approach to resource optimization in the current trend towards SOA [24] [60] [61].

2.3 Literature Study

The three main contributions of our research are the Comprehensive Service Management Middleware (CSMM), the Negotiation Broker (NB) and the Performance Monitor (PM). The CSMM addresses the overall management of the composite process, which includes service selection, SLA negotiation, workflow composition and execution, and monitoring and recovery. The NB and the PM are two of the four modules of the CSMM that provide automated broker-based approaches to SLA negotiation and distributed SLA monitoring, respectively. We present the literature review for the dissertation in three sections for the CSMM, the NB, and the PM as described below. In each section, we discuss the strength and weaknesses of the research works with a view to highlight the motivation behind our research directions. In later chapters as we present our frameworks, we briefly refer to the work presented in this section with respect to comparisons and contributions.

2.3.1. Related Work for the CSMM

We propose CSMM as a complete framework for autonomic Web services-based process management. In this section we present different process management approaches proposed by other researchers, along with the approaches to service selection and composition. The negotiation and monitoring approaches for process management are discussed later as related work for the NB and the PM correspondingly.

Pautasso *et al.* [101] describe a framework for the autonomic execution of Web service compositions using dynamic system configuration, resource allocation and scheduling. As shown in Figure 2.5, a process control manager queues the requests for process execution, a navigator schedules the execution of processes from the queue, and a dispatcher makes the service calls for each process in the system. Based on the resource utilization and thresholds, the optimization policy

determines the allocation of additional navigator and dispatcher threads dynamically during process execution.

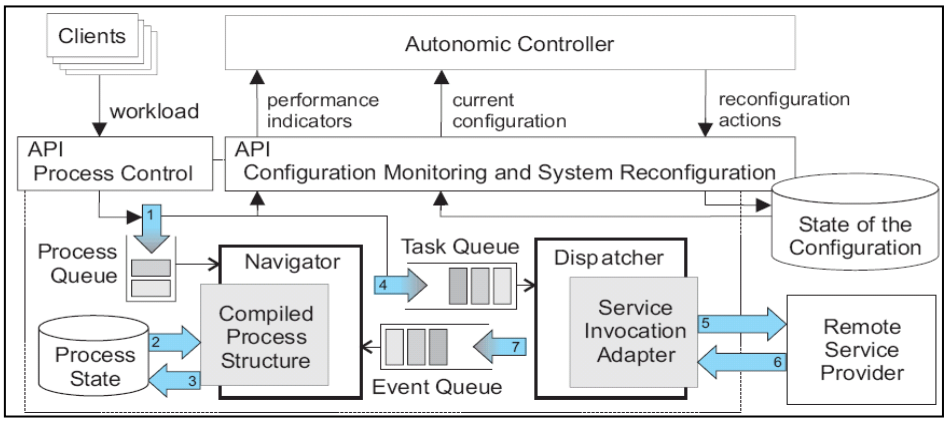

Figure 2.5 JOpera autonomic service composition platform [101]

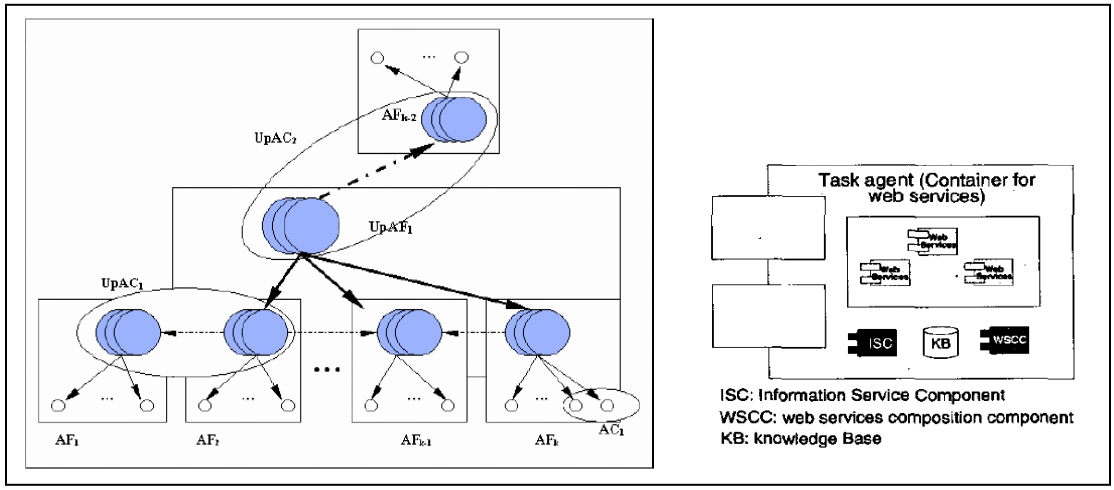

Figure 2.6 Federated Multi-agent System for Autonomic Web Services Management [72]

Liao *et al.* [72] [134] present a hierarchical agent infrastructure containing two types of agents, Task agents and Management agents as shown in Figure 2.6, where federated coordination of the agents execute and manage Web service transactions. Task Agents encapsulate and control one or more Web services; multiple task agents form an Agent Federation, and multiple agent federations can be grouped to form an Upper Agent Federation. In each federation, multiple Management Agents perform service registration, negotiation, and transactions. A container of task agents implements the control logic to select and compose

services from within a federation to provide a service required by the management agent.

An Autonomic Web Services Net Traveler system is proposed by Monge *et al.* [84] where Peer-to-Peer (P2P) Web service brokers (WSB) coordinate, plan and perform Web service choreographies to create and execute business processes. Web services need to be first registered with the framework. WSBs accept client requests through front-end GUIs (Graphical User Interface). Additional software plug-in is used with the WSB to enable fault-tolerance and autonomic management features. Multiple brokers are organized in a structured layout to add scalability and fault-tolerance to the framework.

Yan *et al.* [128] propose a distributed composite process management framework using a network of collaborative agents namely, initializing, monitoring and peer agents. The agents execute relevant sub-processes with the help of a distributed BPEL engine and coordinates with other agents to complete the execution of composite processes. The initializing agent distributes the task among participating agents given a BPEL process specification and returns the final result to the consumer. The monitoring agents monitor service status by subscribing to status reports from the peer agents. The peer agents execute the delegated sub-processes. If necessary, one physical agent can take up the role of another agent, for example, of the monitoring agent or of the initialization agent to further decompose a sub-process.

Momotko *et al.* [83] propose a functional model for adaptive management of QoS-aware service composition. The model supports multiple execution strategies based on dynamic service selection, negotiation, and conditional re-negotiation, flexible support for exception handling, monitoring of SLOs and profiling of execution data. Different execution strategies are described by the authors. According to the contract-all-then-enact strategy, first all services are selected and SLAs are negotiated and then the composition is executed. In the step-by-step-contract-and-enact strategy, the selection and negotiation are done

for part of the composition, which is then executed as the selection and negotiation is done for the next part of the composition. A few other strategies are also described in the paper. The preliminary implementation of the proposed model addresses an adaptive service grid project.

Zeid *et al.* [132] propose an autonomic Web services framework where each resource is encapsulated within an autonomic resource shell that is managed by an autonomic manager. The shell includes a collaboration manager, which manages access to multiple services and invokes them through a reputation manager. The collaboration manager composes the local services to publish a composite service and responds when consumers call to negotiate SLAs of its local services. It collaborates with other collaboration managers of external resource shells. The framework does not provide details regarding the implementation of the autonomic behavior, negotiation mechanism, and process management and mainly focuses on service management rather than process management.

Cibrán *et al.* [28] propose Web Service Management Layer (WSML), a middleware to facilitate development and management of integrated service applications as shown in Figure 2.7. JAsCo [28], an Aspect Oriented Programming (AOP) language, is used to modularize implementation of the management functionality within WSML. A new JAsCo *Aspect Bean* is defined dynamically as required with specific policies for each management aspect such as dynamic service selection and binding, service swapping, automated billing, caching, and monitoring. JAsCo *Connectors* are also created dynamically to bind the client application with a specific Web service based on the policies defined in the aspect bean. Client-side management is implemented using AOP in client applications. By isolating the management tasks from the service and application codes and placing the same in the WSML, repetition and maintenance of similar management related code is avoided.

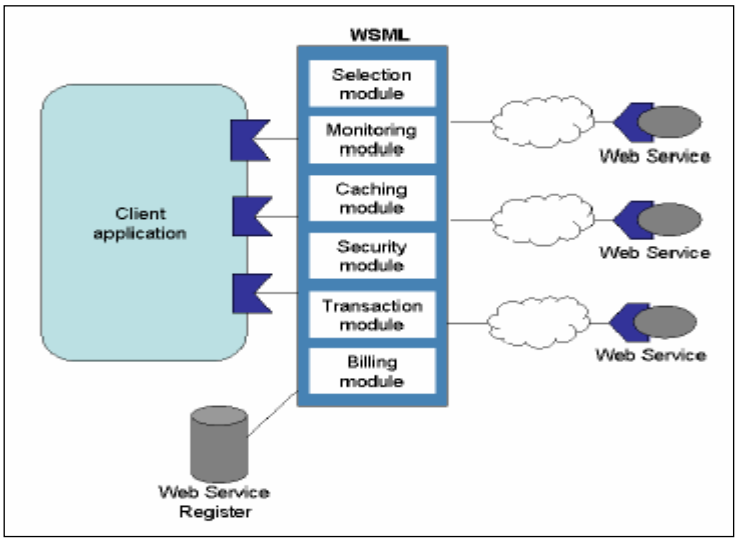

Figure 2.7 General architecture of the WSML [28]

Dustdar [39] describes the necessity of autonomic process and service management and his team's research initiatives and contributions in this area. The author specifically addresses the following aspects and cites corresponding research works in the paper: model driven framework design and verification, building active service registries and search engines for efficient service discovery, and context-based and relevance-based service composition and enactment.

We summarize the process management approaches described above in Table 2.1.

Table 2.1 Process management approaches

Work	Objective	Approach	Comments
Pautasso [101]	Workflow execution and management	Autonomic system configuration, resource allocation and scheduling	Service selection and composition not discussed
Liao [72]	Workflow composition, execution and management	Hierarchy of federations of task and management agents	Management of federation is necessary, which is not discussed in detail.

Work	Objective	Approach	Comments
Monge [84]	Scalability, fault-tolerance and availability in process management	P2P WS brokers coordinate to orchestrate and execute business processes in an Autonomic Web Services Net Traveler system	Presents a preliminary representation of the framework with no details about the broker co-ordinations for process management or fault-tolerance.
Yan [128]	Distributed composite process management	Distributed P2P agent framework decomposes BPEL process for distributed execution and monitoring	The framework is not described in detail, specifically the functionality and multiple roles that the peer agent can take.
Zeid [132]	Autonomic Web service management	Collaborative framework for autonomic Web services using multiple managers for service management, composition and invocation	Main focus is on service management, details about implementation of the various managers are not given.
Cibrán [28]	Client-side process management	Applies aspect-oriented programming in client code and middleware to dynamically create and manage composite process	Proposes a semantic match-making algorithm for automatic service selection. However, the policy and process management are not described in detail and services need to be registered with WSML.
Dustdar [39]	Autonomic process and service management	Model driven approach, service registries and search engines for discovery, and context-based and relevance-based composition and enactment.	Describes disjoint works on different aspects but a process management framework is not proposed that combines the different tasks

Service Selection and Composition

The service selection and composition modules are not implemented in the scope of the dissertation. However, the state-of-the-art research on these two areas was studied to design the CSMM framework. More detailed surveys can be found in Rao *et al.* [103] and Dustdar *et al*. [38].

The two main approaches to service selection are: using the UDDI for service discovery and using the semantic languages for matching service selection criteria. Research in this area is targeted towards expression of service requirements and offerings, efficient matching of the selection criteria, storing and managing the service quality and reputation information for discovery, and

selection of appropriate services based on non-functional requirements and reputation given a process QoS.

Service selection is often combined with service composition as the selected services are orchestrated to create a composite process. Research on service composition includes approaches and frameworks for static and dynamic service composition, formal specification and execution of composite processes, computing and meeting the goal of a composite process QoS, verification of data flow or workflow in composite processes, optimized selection of services, and using logic-based approaches for semantic service requirements matching. We describe briefly some of the common approaches below.

Computation and management of process QoS has been addressed by several researchers. Yu *et al.* [131] present a broker-based architecture for QoS-based service selection using two different models with an objective of maximizing composite process specific utility function under end-to-end QoS constraints. The authors compared the performances of the two models for service selection.

Cardoso *et al.* [21] propose a mathematical model to automatically compute the QoS of a composite Web service workflow process from the QoS metrics of the component Web services. However, service selection methods are not included in the literature.

Aggarwal *et al.* [2] present a Web service composition framework METEOR-S (Managing End-To-End OpeRations for Semantic Web services) to create and manage dynamic service compositions. Users define an abstract process with placeholders for services, which are automatically selected at execution time based on particular business and process constraints and bound to generate an executable process. METEOR-S uses semantic Web technology to represent the requirements for each service in the process and multi-phase constraint analysis to satisfy the constraints for service selection. The automatic service selection feature facilitates automatic recovery of composite processes.

Maximilien and Singh [80] propose an ontology and a Web Services Agent Framework (WSAF) to disseminate reputation and endorsement information for dynamic Web services selection. Agencies manage services, their registries and reputations, while agents communicate with the agencies to find appropriate services that meet the functional and non-functional requirements expressed in an XML policy language. Agents execute and monitor service call, and report the results to the agencies. Reputation is built from consumer ratings of a service. New services with no reputation are endorsed by trustworthy service providers or consumers. Details about the computation of the reputation score are not provided in the paper.

Blum [15] proposes to extend the use of categorization technical models, called *tModels* [87], within the UDDI to represent different categories of information such as version and QoS information. A Web service entry in the UDDI can refer to multiple tModels that are registered with the UDDI, which in turn can contain multiple property information. Each property is represented by a name-value pair in the tModel. Xu *et al.* [126] propose a service discovery approach that use tModels to include QoS information in the UDDI.

Other comparable work in this area includes study of the requirements for representing [77] and processing heterogeneous context information [68] for Web services to enable context-based service selection. Research on semantic Web languages is geared towards representing information in a machine understandable format to leverage automated service selection, composition, and management. Bansal *et al.* [11] defined the OWL-S based semantic Universal Service Description Language (USDL) for automated service discovery. Tosic *et al.* [113] proposed Web Service Offering Language (WSOL) to allow formal specification of important management information such as classes of service, functional and accessibility constraints, price, penalties and other management responsibilities. The authors also proposed Web Service Offering Infrastructure

(WSOI) to demonstrate the usability of WSOL in management and composition of Web services.

Issa *et al.* [58] propose WS-Notification be used as a base medium to enable sensing and routing information change at the level of Web services using a publish-subscribe mechanism. They describe an algorithm where based on the notifications, component services of a composite process update their information about the current state of other services in the process. Then the component services re-compute the pre-established global execution plan to reflect the updated status and continue process execution from the new state.

Security and Reliability Aspect

Autonomic security mechanisms are usually addressed discretely from the other autonomic aspects. Some of the security systems can be added as a layer on top of an autonomic Web services system to enable its self-protection aspect. Gutiérrez *et al.* [50] describe the state-of-the-art of the current security and reliability standardization efforts and highlight the importance of a universal standard for addressing the problem of Web service security.

Zhu *et al.* [136] illustrate Reliable Web Services Bus (RBUS), a QoS-aware middleware, for ensuring Web services reliability. The RBUS demonstrates three features to achieve higher reliability in services systems: reliable messaging; service fault tolerance that applies Virtual Service (VS) concept; and service priority where services with higher priorities are more reliable than those with lower priorities. Yang *et al.* [129] propose use of colored Petri-nets to model service compositions for higher reliability.

Dai *et al.* [33] propose an approach to detecting security problems using the feature recognition technique by virtual neurons, which are distributed in a compound P2P and hierarchical structure in the network. Park *et al.* [99] propose a policy based Autonomic Protection System (APS) that applies Role Based Access Control with an Intrusion Detection System, and allows self-adaptation of the security policies to suit various computing environments.

Trust models are increasingly getting popular to establish a federated user group or an on-demand trust relationship between service providers and service consumers for increased security and reliability. Olson *et al.* [96] propose an approach to negotiate trust relationships as an authorization procedure for Web services. The authors present a third party negotiation system for trust negotiation to gain access to a Web service. Maximilien *et al.* [79] present a self-adjusting trust model to establish trust relationship between coordinating agents that assist in QoS-based service selection in a multi-agent framework.

Coetzee *et al.* [29] and Mecella *et al.* [81] propose access control frameworks for Web services conversations pointing out the necessity to address the nature of repeated communication with Web services where one time access control may not be enough. The model demonstrated in Coetzee *et al.* [29] takes in account both trust and context awareness. They propose a logic-based access control framework, which defines access control policies for decision making on authorization. Mecella *et al.* [81] focuses on the importance of a trade-off between the protection of the access control policies and the necessity to disclose partial policy information to the clients.

Birman *et al.* [14] extend the general architecture of Web service systems to add high availability, fault tolerance, and autonomous behavior. The architecture includes server and client side monitoring, a consistent and reliable messaging system using information replication, data dissemination mechanism using multicasting, and an event notification system. The reliable messaging and fault-tolerance techniques can also be applied to process management.

Discussion of Related Work on Process Management

The literature study gives an overview of the extent of research that has been carried out in the different areas of composite Web services-based process management. We discuss the existing solutions summarized in Table 2.1 with a

view to outline their limitations and portray our research objectives in the areas of complete composite process management.

Liao *et al.* [72], Monge *et al.* [84], Yan *et al.* [128] and Zeid *et al*. [132], all propose a high level overview of frameworks for process management. In most cases, one management aspect is tied to another, for example, service orchestration is often tied to either service selection or monitoring. Therefore, the consumer has to implement the framework for everything and does not have the flexibility to select one service without the other. Monge *et al.* and Yan *et al.* propose distributed P2P agent-based process execution and management, where service selection and composition is done at the framework to enable monitoring. P2P frameworks reduce the risk of having a central point of failure but make failure tracking, recovery and coordination of the workflow much more challenging. The papers do not describe those aspects of process management in detail. Monge *et al.* provide a very brief overview of the framework while Yan *et al.* do not describe how the different types of agents coordinate, particularly, how the peer agents change their roles.

Zeid *et al.* [132] mainly focus on the framework for autonomic Web services, which also enables process management. The services can be composed with other services in other shells by the collaboration manager and thus initiate a composite process. However, the control and execution of the process and the internal negotiation mechanism between the resource services and the collaboration manager are not explained by the authors.

Liao *et al.* [72] propose a very interesting hierarchical multi-level multi-agent federated process management framework. The approach requires services to be grouped into federated agent framework to deliver process management services. However, the formation and management of agent federations is not described in detail. Moreover, this approach also combines the different tasks of process management, and services have to be selected, composed and executed by this framework to be monitored during execution.

Pautasso *et al.* [101] addresses the specific aspect of scheduling and resource distribution for process management. Dustdar [39] discusses their various research efforts towards building an autonomic process management framework using a complex and versatile SOA-based process modeling approach, a dynamic service binding and invocation approach, semantic composite service search engines, and autonomic context-based service adaptation for service composition and enactment. However, a complete framework has not been proposed that combines all the different aspects to enable autonomic process management. Cibrán *et al.* [28] provides a middleware framework that consists of multiple layers in a centralized tightly bound structure. Although the approach presents an interesting technique of on-demand creation of necessary aspect beans to satisfy clients' requirements for different management services, a specific aspect oriented language-based implementation and client-side code level instrumentation is required for using the WSML framework.

Based on the above observations, we deem that a distributed management framework is essential to provide the consumers with the flexibility of getting separate services for service selection, SLA negotiation, composition and execution, and process monitoring. This way the consumers would have the flexibility to move to other options if other services become available that best meet the consumers' requirements. Second, common standards should be adopted to maintain the main goal of interoperability in SOA. Any custom setup, language, and instrumentation limit this feature. Third, we believe that a central managerial view of the process execution state is important for most critical business processes, and therefore, a centralized approach to workflow execution and management may be preferable in most cases. Finally, no matter how distributed the different modules are in the process management framework, a central service request or access point is essential for complete automation of client-side autonomic execution and management of composite processes. For this reason also, the central execution of workflow is important.

2.3.2. Related Work for the NB

Negotiation systems have evolved greatly with the advancement of computation and communication technologies. Beam *et al.* [12] provides a detailed survey of the state-of-the-art of various negotiation strategies. Different Negotiation Support Systems (NSSs) [36] [63] have been proposed to assist human negotiators in computations for decision making in negotiation processes. The Web media further contributed to negotiation systems by facilitating on-line negotiation over the Web [109] [86]. The current trend towards SOA has enabled automated broker-based negotiation services on the Web with a view to leverage service compositions for business processes [71] [74] [85]. To ensure QoS of business processes, establishing a SLA between the service provider and the service consumer is critical. Therefore, much research effort is currently driven towards building on-line automated broker-based efficient SLA negotiation systems for e-Services [30] [71] [130]. We describe below some of the recent related work on Web services and Web-based negotiations, decision models and SLA negotiation frameworks.

In contradicting to the requirement of a negotiation framework, Wilkes [123] argues that the combined notion of a consistent approach to utility and a flexible pricing scheme can reduce the burden of having a complex negotiation system. The author proposes the concept of a price function model that maps multiple Service Level Objectives (SLOs) to a price value and embeds the model in the SLA. The model can have limitations in terms of the number of SLOs it can model as a price value.

Faratin *et al.* [41] define time-based negotiation decision functions with a cost-benefit model for bilateral bargain where the goodness of an offer is measured by its utility value. The time-based functions are used to compute offers. The authors also show how the different parameters of the functions can be varied to

influence the concession nature of the negotiating parties and how the functions may be combined to model other behavioral negotiation strategies.

Hung *et al.* [55] describe some of the issues in this research area. The authors also propose WS-Negotiation language, which contains negotiation message, protocol, and strategy, and give an overview of a framework for negotiation between two Web services over the Web.

Ludwig *et al.* [74] present an approach where a Thomas-Kilmann questionnaire is used to measure consumers' conflict mode or nature of negotiation. The data from the questionnaire is then fed into an agent-based automated tool to extract consumers' time-based concession graphs for each offer and attribute. This approach can be used for designing electronic negotiation systems but it has some drawbacks. It requires a large sample size to better estimate the conflict mode and the questionnaire covers only a few specific user groups. It does not take the opponent's offers into account during negotiation, which influences the strategy. The approach is validated using an agent simulation.

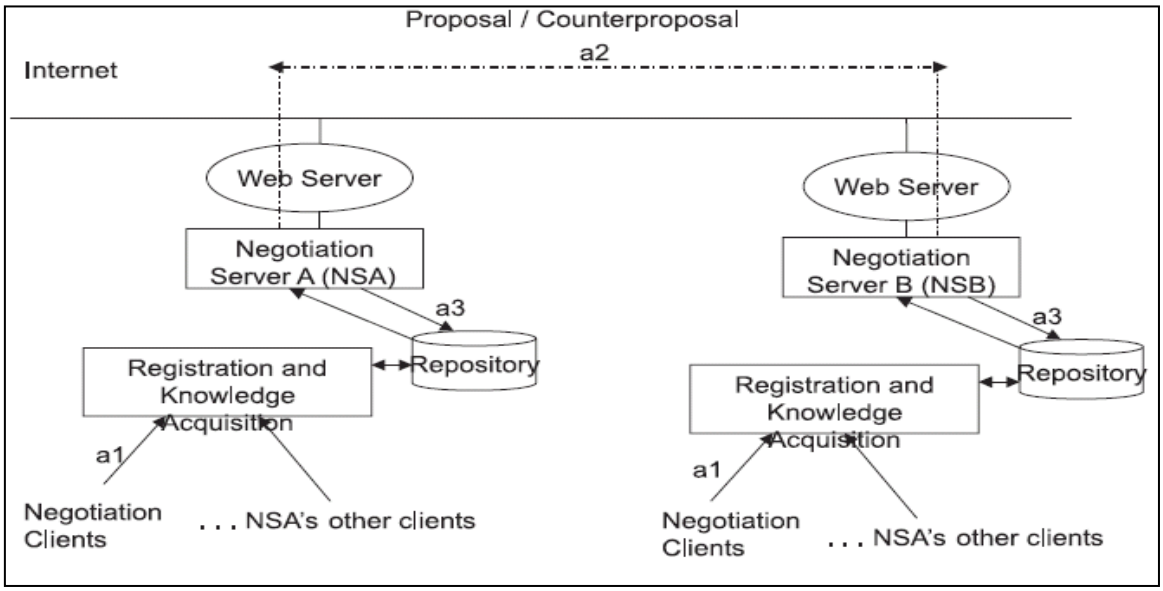

Figure 2.8 Automated context-based negotiation using negotiation server [109]

Su *et al.* [109] propose a negotiation server for e-commerce to perform bargaining-type negotiations automatically. Each negotiating party registers with a negotiation server and provides their goals, contexts, requirements and priorities. The servers then conduct negotiation automatically using constraint satisfaction, rule-based conflict resolution, and event systems.

Li *et al.* [71] propose an automated negotiation framework as shown in Figure 2.8 based on a finite state automata and a set of negotiation protocols. The framework maps negotiation context to negotiation goals using policies and the goals are mapped to negotiation rules and plans using negotiation strategy, to carry out bilateral bargaining. The framework uses the event and rule-based negotiation server proposed by Su *et al.* [109] in the back-end.

Narayanan *et al.* [85] propose a learning model to predict the opponent's strategy during negotiation, which is used as a basis for deriving own strategy in a non-stationary negotiation environment. The authors apply Bayesian learning to learn a mixed-strategy profile of the opponent to derive a strategy to generate counter-offer that produces maximum utility value payoff to reach optimal solution. A non-stationary Markov chain is used to model the negotiation process for a single issue.

Chhetri *et al.* [25] propose an agent-based negotiation framework as part of the Adaptive Service Agreement and Process Management (ASAPM) framework, which ensures service management by stateful coordination of complex services. Given an abstract composite process, the framework performs autonomous negotiation to find concrete services for the various tasks. For each task, the framework coordinates one-to-many negotiation with all candidate services for that task, and then selects one task based on the task level QoS constraint and agreed SLAs, otherwise, a re-negotiation is called with revised QoS constraints. The authors demonstrate the usability of the system in different application domains and on the service provider's side either as an Agent based Negotiation System or as a Web Service based Negotiation System.

Comuzzi and Pernici [30] propose a policy-based negotiation broker framework as shown in Figure 2.9 to perform partially or fully automated negotiation of QoS parameters for service selection. However, the negotiating parties need to have knowledge about the strategy model supported by the framework in order to specify their choices of parameters for the strategy model. The user preferences are communicated in the form of WS-Policy specification.

Yee and Korba [130] propose a scheme for negotiation of e-services under uncertainty that suggests what offer or count-offer to make using the existing records of similar negotiations of reputed participants who have negotiated the same issues. The paper presents initial state of the research with no validation results. The authors present a reputation model and manager to be used with software agent-based framework where each party has a separate store of its negotiation history.

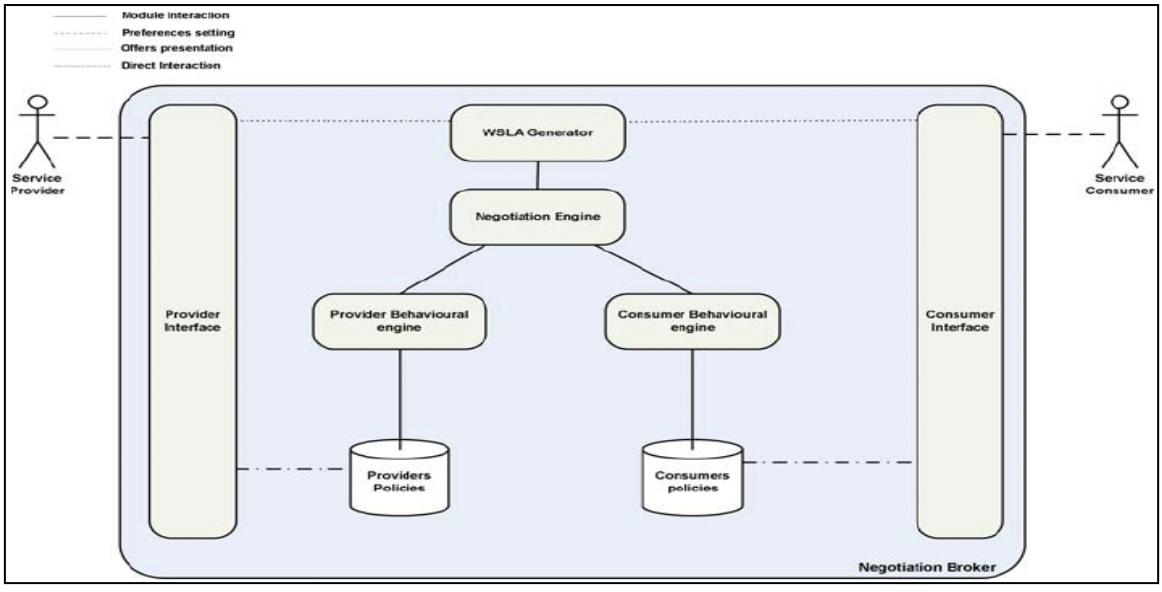

Figure 2.9 Policy-based negotiation broker framework [30]

Gimpel *et al.* [48] propose Policy-driven Automated Negotiation Decision-making Approach (PANDA) where a policy expresses a party's private negotiation strategy as a combination of rules and utility functions. In their approach, the decision making problem is decomposed into multiple aspects.

Each aspect is handled by a separate Decision Maker (DM) framework, which interact with each other to jointly provide a solution.

Brzostowski *et al.* [18] propose an approach to modeling behaviors of opponents in a negotiation for predictive decision-making. Only the history of the offers during the ongoing negotiation is considered. The mechanism estimates the influence of different factors that contribute to the opponent's behavior during negotiation and uses this information to predict the opponent's future behavior based on which strategies are chosen to generate counter-offer. The authors present comparative study of their approach against random strategy-based negotiation for simple test scenarios.

Hou *et al.* [54] propose a negotiation strategy which applies non-linear regression analysis for learning opponent's behavior in terms of decision functions and makes concessions accordingly to maximize own utility. The authors only consider the history of offers in the ongoing negotiation and use an agent simulation to validate their approach.

Chiu *et al.* [26] apply semantic Web technologies for streamlining the negotiation issues, alternatives, and trade-offs for automated negotiation. The authors also propose a methodology to elucidate the dependencies of the issues to facilitate trade-off in e-negotiation processes for service discovery. A NSS is extended for demonstrating the validity of the approach with regards to repeatable and semi-structured negotiations as in composite business processes.

Lau *et al.* [69] illustrate an intelligent agent-based negotiation approach for e-marketplace that applies a knowledge discovery method and a probabilistic negotiation decision making mechanism. The authors argue that their approach is more effective and efficient compared to an agent-based Pareto optimal negotiation approach for a simulated complex and dynamic market.

The various negotiation approaches described above are summarized in Table 2.2 below.

Table 2.2 SLA negotiation approaches

Work	Framework	Automation	Decision Model	Organization	Short-comings
Faratin [41]	Various negotiation decision functions	Framework not defined, can be manual or automated	Time-based decision functions for different strategy models	Not defined	Only functions are defined, no framework for negotiation is proposed
Hung [55]	WS-Negotiation language to express message, protocol and strategy	Not defined and not implemented, two possible implementation scenarios stated	Cost-benefit, other models may be applied as specified in the language	Designed for distributed over the Web negotiation using NSS	Focus is on the language, other aspects are not described in detail
Su [109]	Event-based Negotiation server	Semi-automatic bilateral bargain	Rule-based reasoning	Internet-based distributed servers	Maintain server, define rules
Li [71]	Automated context mapping framework	Automated bilateral bargaining	Rule-based reasoning and server of Su [109]	Internet-based distributed servers	Define mapping rules by expert
Ludwig [74]	Questionnaire based user conflict model determination	Manual questionnaire step, then automated tool-based model extraction step	Time-based concession graphs for each offer and attribute	Two step model to build agent-based automated system	Large sample size required, doesn't consider opponents' offers, and only for specific user groups
Narayanan [85]	Learning model to predict opponent's strategy and derive own strategy	Automated negotiation for non-stationary environment	Bayesian learning based where Markov chain is used to model negotiation	Agent-based framework	Bilateral single issue negotiation is used with a limited number of hypothesis about opponent's strategies
Commuzi [30]	Broker-based negotiation framework	Automated or semi-automated	Cost-benefit model with time-based functions	Local	User preferences are taken as low level parameter values

Work	Framework	Automation	Decision Model	Organization	Short-comings
Chhetri [25]	Hierarchical agent and multi-agent based negotiation to ensure end-to-end process QoS	Automated	Uses time-based function and discusses the applicability of other adaptive strategies	Local agent-based system or Web service based system on provider's side, part of ASAPM	Decision model, decomposition of process SLA is not described in detail
Yee [130]	Design of a reputation system for consulting possible moves during uncertainty	Automated Negotiation framework not illustrated	Uses a reputation generation system to assist in decision making in uncertainty	Central reputation system while history of negotiation is distributed	A basic model and architecture; it is unclear how the reputation system is maintained
Gimpel [48]	Policy-driven Automated Negotiation Decision-making Approach (PANDA)	Automated	Decomposes the main problem into sub-problems; uses rules and policies in decision model	Distributed multiple Decision Maker (DM) modules	Maintain the multiple DMs; policies have to be specified for each DM
Brzostow-ski [18]	Learning to predict opponent's concession by difference method	Automated	Computes differences in opponent's offers to predict future sequence of offers	Agent simulation	User preference elicitation is not discussed, framework not provided
Hou [54]	Learning opponent's decision function, reservation values and deadlines	Automated	Uses linear regression analysis of opponent's offers to predict concessions	Agent simulation	Good for single negotiation tactic; framework not defined
Chiu [26]	Semantic web-based negotiation system	May be automated or NSS based	Semantic languages are used to express issues, alternatives and trade-offs and dependency among issues	Central Ontology definition and management and e-negotiation framework	Decision models and frameworks are not defined

Work	Framework	Automation	Decision Model	Organization	Short-comings
Lau [69]	Probabilistic negotiation agents empowered by knowledge discovery mechanism	Automated	Applies ranking algorithm to evaluate offers	Agent simulation	Only considers time-based strategy

Discussion of Related Work on SLA Negotiation

We summarized a number of negotiation approaches in Table 2.2 that are related to Web based negotiation and contracting. Many of these approaches only focus on the core Decision Support System (DSS) that comprises negotiation strategy and protocol, while others present the complete negotiation framework.

Faratin *et al.* [41], Narayanan *et al.* [85] Brzostowski *et al.* [18], Hou *et al.* [54], and Lau *et al.* [69] present core decision models based on independent or combinations of various negotiation theories, such as time-based decision functions, game theory, knowledge-based learning, mathematical regression analysis, and probability theories. These approaches can be adopted in other negotiation frameworks for similar types of negotiations. In most cases, researchers validate their approaches using software agent simulations.

Hung *et al.* [55] and Chiu *et al.* [26] emphasize the representational aspect of negotiation information. Ludwig *et al.* [74] propose a two step heterogeneous approach that combines a questionnaire-based modeling of user's conflict mode and then extraction of the user's concession graph from the data using automated tools to be able to select an appropriate negotiation strategy for the user. This approach, however, does not take into account the opponent's behavior, which is an important factor in human decision making approaches.

Policy is used in negotiation mainly in two different ways, one is to represent the decision mechanism and the other is to express user preferences. Li *et al.* [71] use policy to map negotiation goals to low level decision-action rules. Gimpel *et*

al. [48] use policy to express decision rules and actions including utility functions to evaluate goodness of offers. Commuzi *et al.* [30] use policy for both purposes. Policies can also specify rules for the management of the negotiation framework. Due to the flexibility of the use of policy for different purposes, we consider it as an effective means for representing negotiation information, particularly since WS-Policy [121] has already been accepted as a standard for Web services.

Commuzi *et al.* [30], Chhetri *et al.* [25], Gimpel *et al.* [48], Chiu *et al.* [26], and Su *et al.* [109] propose different negotiation frameworks. Gimpel *et al.* [48] propose a distributed negotiation approach where the problem is decomposed into multiple aspects to enable negotiation using multiple decision maker models. Although it may be good for complex problems, it introduces the complexity and overhead of problem decomposition and combination of the negotiation results. Therefore, this approach is not suitable for SLA negotiation that typically deals with a small number of negotiable issues.

Su *et al.* [109] propose remote Web-based negotiation using negotiation servers for each negotiating party. Although a party can register with a negotiation server hosted by another party, it involves security and privacy issues and depends on network performance. Chiu *et al.* [26] propose an extension of a NSS, however, an automated negotiation is possibly a more efficient approach considering the current trend of service composition in SOA.

Commuzi *et al.* [30] and Chhetri *et al.* [25] both propose agent-based automated negotiation for service composition. Chhetri *et al.* [25] propose a hierarchical multi-agent framework for conducting parallel negotiation with a number of candidate services to be able to select the best service to meet process QoS requirements. Although the concept is good, the overhead in terms of time and management of the agent framework may be considerable. Also given a basic service offering for all candidate services, a limited number of negotiable parameters, and reputation rankings (if available) of the services, multiple parallel negotiation may be unnecessary.

Based on the above discussion we deem that an automated negotiation framework is necessary for Web services SLA negotiation that can accommodate various decision models and dynamically select the most appropriate model given the clients' preferences. Clients should be able to express their preferences at the business level and a negotiation framework should ideally translate the information to negotiation strategy. Opponent's behavior plays an important role in decision making during negotiation and an ideal negotiation framework should be able to adapt to the dynamic status of the negotiation process and learn from the past negotiation data to address opponent's strategy efficiently.

2.3.3. Related Work for the PM

Extensive research has been done on server-side resource, network and inter-organizational process monitoring. Several software products are available in the market that can provide comprehensive monitoring data. However, monitoring intra-organizational Web services-based processes has not been addressed to the same extent.

Momm *et al*. [82] propose a conceptual manageability infrastructure based on the Web-Based Enterprise Management (WBEM) [37] standard to monitor Web service compositions for SLA-driven management. The Common Information Model (CIM) is part of the WBEM and is used to model management information, which can in turn be used with either WS-Management [37] or WSDM (Web Services Distributed Management) [89] to provide management services using Web services. Monitoring is done through instrumentation of the managed elements or services. Three different instrumentation techniques are described in the paper, which includes ORACLE BPEL Process Manager specific sensors, EJB bindings, and a management Web service.

Vaculín and Sycara [115] describe event-based monitoring of service interactions and error-handling mechanism for OWL-S [98] based semantic Web services using an OWL-S Virtual Machine (OVM), which lies in between the

service provider and the service consumer. The interaction trace allows analysis, replay and debugging of process execution by human or software agents. The OVM is a generic OWL-S processor that allows Web services and clients to interact on the basis of OWL-S description of the Web service and OWL ontology. It is also a generic execution engine, which can be used to develop applications that need to interact with OWL-S Web services. The authors present the taxonomy of an OWL-S based event model for Web services, which is implemented at the OVM. An event handler can be defined on the OVM to either log the event or inform a monitoring system. The client needs to be OWL-S aware because at least a basic OWL machinery is required to translate and process the events.

Tröger *et al.* [114] argue about the practicality of having stateful services rather than stateless services since many of the real world services are wrapping state information using application-specific concepts that enables monitoring through vendor-specific interfaces. The authors present Adaptive Services Grid (ASG) Services Infrastructure (SI) architecture that is based on established Web service standards, and that supports dynamic hosting and monitoring of heterogeneous and stateful service implementations. The SI is basically a thin and scalable abstraction layer that enforces the instantiation of services to be used in client applications through a factory operation. The resulting endpoint reference document is then used for service invocation and monitoring.

Sahai *et al.* [105] propose a Management Service Provider (MSP) model for remote or outsourced monitoring and controlling of E-Services on the Internet. The model requires E-Services to be instrumented with specific APIs to enable transaction monitoring using agent technology. An E-Service Manager is then deployed that manages the E-Services remotely with the help of several other components. The model does not address management of composite processes.

A distributed message tracking algorithm is proposed by Sahai *et al.* [104]. In a composite Web service process, each service provider executes the algorithm to

keep track of the current state of the process for monitoring purposes. A data structure is defined to contain all the data relating to the execution of the process. Each service provider adds its own data regarding its execution status to the existing data and analyzes the complete data to verify proper execution of the process or recover from failure. This approach to monitoring means potentially huge data messages must pass through the network, adds a processing overhead for execution of the algorithm to each Web service in the workflow, and makes it vulnerable to possible loss of data due to failure of a service in the process.

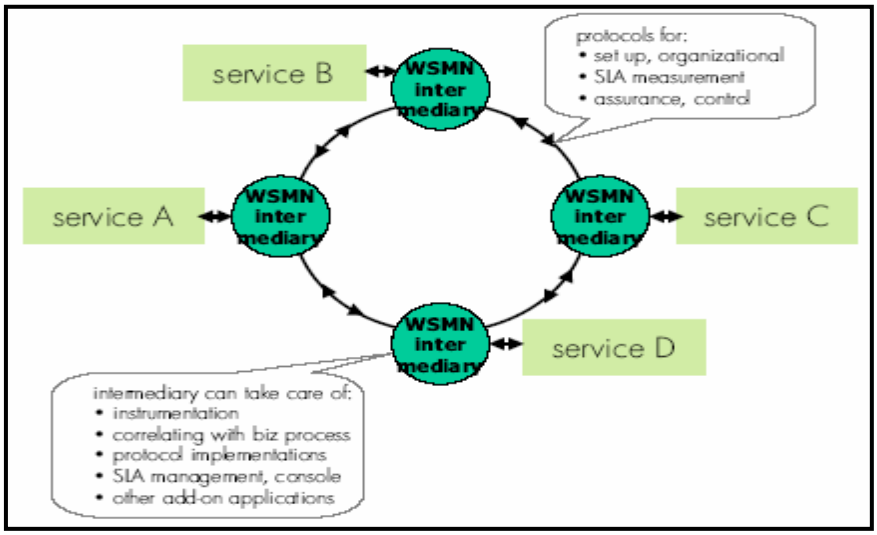

Figure 2.10 Web Services Management Network (WSMN) [75]

Sahai's message tracking algorithm is also used in the Web Service Management Network (WSMN) Agent framework [75], a logical overlay network proposed by HP Lab researchers, as shown in Figure 2.10. It introduces a specification language for the SLAs and uses the algorithm with a set of Service Level Objectives (SLOs). The WSMN implements an automated and distributed SLA monitoring engine for Web services using a network of cooperating intermediaries for federated service management. Each intermediary acts as a proxy sitting between the Web service and the outside world. The proxy components are attached to SOAP toolkits at each Web service site of a composite process, and execute the message tracking algorithm.

Monitoring is performed by a number of current enterprise process management products. CA's Unicenter [111] uses message interception at intermediaries. In-code instrumentation with Application Response Measurement (ARM) [64] APIs is used with reporting agents to collect monitoring data in the IBM Enterprise Workload Manager (EWLM) [23]. CA Wily SOA Manager [19] is another very recent product from CA that uses agents both at the service provider and the service consumer's ends to monitor and manage Web service processes. The agents report to a central manager or cluster of managers, which in turn report to a collector.

Table 2.3 summarizes the work on monitoring described above.

Table 2.3 Process monitoring approaches

Work	Framework	Technique	Comments
Momm [82]	Manageability infrastructure based on CIM and WBEM	WS is instrumented using sensors, EJB, and WS and performance data is analyzed based on BPEL	Monitors the execution of the process not specifically the SLAs; high overhead for extensive instrumentation
Vaculín [115]	Monitoring of OWL-S service interactions for error-handling using OWL-S Virtual Machine	Event taxonomy to create event handlers at the OVM for monitoring and logging OWL-S interactions	Clients require basic OWL machinery to interpret events, OVM executes the process
Tröger [114]	Adaptive Services Grid (ASG) Services Infrastructure (SI) architecture, based on WS standards, supports dynamic hosting and monitoring of stateful services	SI enforces service instantiation to be used in client applications and the resulting endpoint reference document is used for service invocation and monitoring	Service has to be instantiated at SI middleware to be used and monitored, all service state information is maintained at the SI
Sahai [105]	Management Service Provider (MSP) model for remote or outsourced monitoring and controlling of E-services	Requires E-services to be instrumented with specific APIs to enable transaction monitoring using agent technology by a manager	Does not support process monitoring by the manager
Sahai [104]	Distributed message tracking algorithm for SOAP based WS	Each service analyzes the message and attaches respective monitoring data to a message structure that is passed along the process	Increases each service's processing load and requires large amount of data to be passed through the network
Sahai [75]	WSMN agent network for SLA monitoring	Message tracking [104] used at intermediaries that are connected to SOAP toolkits on the WS servers	Has the overhead of message tracking and the intermediaries need to be managed as well

Work	Framework	Technique	Comments
CA [111] Uni-center ®	Uses reporting agents at WSs with a central management server	Uses message interception at intermediaries	Large software suite but for inter-organizational WS monitoring
IBM [23] EWLM®	Multi-agent monitoring and management framework	In-code instrumentation with ARM API is used, agents query data and report to central server	For inter-organizational setup, detail monitoring data obtainable at the cost of higher code maintenance
CA Wily SOA Manager ® [19]	Agent-based monitoring and enterprise management application suite	Agents use bytecode instrumentation and reside both on service provider and service consumer's ends	For inter-organizational setup, good for Web-based processes, incurs system and agent maintenance costs

Discussion of Related Work on SLA Monitoring

The different techniques applied for monitoring SLAs for composite Web services-based processes are summarized in Table 2.3. Two common Web service monitoring techniques are server-side instrumentation and message interception. Instrumentation techniques allow extensive monitoring capabilities and provide the most accurate data. However, depending on the instrumentation type and monitoring details, the maintenance cost and monitoring overhead can vary considerably. Monitoring data is either queried as necessary or clients can subscribe to notifications for specific events at the server.

Message interception is the most popular monitoring technique for SOAP-based Web services. It is implemented either at the intermediary, which is external to the server that provides the Web service, or on the server itself as part of its message processing layer. Intermediaries require additional maintenance whereas any update of the server software may require an update of the message interceptor modules. Based on the monitoring requirement, the technique has to be chosen carefully.

Commonly, the monitoring tool is integrated with the process execution engine as in Vaculín *et al.* [115] and Tröger *et al.* [114]. Vaculín *et al.* [115] propose OVM for execution and management of semantic Web services-based composite processes. Clients can subscribe to OVM for fault notifications during process execution through definition of specific event handlers using an event ontology.

Tröger *et al.* [114] propose a stateful ASG SI middleware framework for invocation and monitoring of Web services. Query interfaces are provided to get service status information from the SI for a specific process identified by a process ID in the SOAP message header.

Momm *et al.* [82] propose CIM based process modeling for monitoring purposes using WBEM standards in a centralized monitoring framework. The authors illustrate their approach for three different instrumentation techniques in the ORACLE-PM environment, and show that ORACLE bytecode instrumentation gives the best performance in terms of monitoring overhead. However, it is specific to the environment used, and therefore, may not be usable in other environments.

Sahai *et al.* in three different works propose different monitoring techniques. Their first work [105] describes the MSP model for remote monitoring, where services are instrumented for monitoring by agents and the performance data can be queried by a manager. This architecture does not support process management. Their second work [104] describes a message data structure, which stores Web services' state information as part of the SOAP message as it passes through various services during process execution. This is an interesting approach that uses the message interception technique for SLA monitoring. However, there is a risk of data loss and increase in data transfer over the network. Also each service in the workflow has to be programmed to analyze the data to check for possible exceptions and augment it with new state information, which is simply not feasible. The third work of Sahai *et al.* [75] uses WSMN, a logical overlay network of intermediaries, for monitoring SLAs in a federated environment and uses the same message tracking algorithm as the second work.

The other three works are commercial management application software suite from CA and IBM namely the CA Uni-center ® [111], IBM EWLM ® [23] and CA Wily SOA Manager ® [19], which are excellent for enterprise service and process management. The common short-coming that we notice in all the

solutions is that inter-organizational distributed monitoring is not supported by most of the frameworks and it is mainly due to the problem of accessibility to other services' status information for a specific process. We, therefore, strongly believe that a simple inter-organizational process monitoring framework is necessary in the current SOA. Also to maintain simplicity additional management overhead should be avoided if possible by implementing message interception on the server instead of using additional network of intermediaries.

2.4 Summary

We define some common concepts and lay out the background on Web services research in this chapter. We start with introducing Web service in Section 2.1 and then describe service composition, Web service life cycle, and the commonly used standards to maintain interoperability.

In Section 2.2, we describe the necessity of having a service management infrastructure. We support our arguments by presenting the layout of the heterogeneous Web service environment, the complexity involved in managing such an environment, and the main goals and aspects of Web service systems management.

We present the literature study on Web services-based process management in the second part of the chapter. Our research addresses three main Web service management aspects which are: autonomic process management, SLA negotiation and distributed SLA monitoring. Accordingly, we presented our literature study under three sections. We discuss the state-of-the-art research in the above three areas, which includes selected recent research and industry works. We also summarize the literature study in table format to highlight specific aspects of each work in each research area.

Finally, we provide a detailed discussion of the strengths and weaknesses of the different works that leads our motivation to pursue specific problems and research directions in those areas.

Chapter 3

The Comprehensive Service Management Middleware

The greatest potential of Web services lies in the possibility of weaving together multiple services dynamically to generate multi-organizational business workflows. The process of orchestrating and executing a Web services-based workflow, however, involves multiple steps. Depending on the workflow requirements and the available services, each of the steps can incur considerable cost and complexity on the consumer. We propose a conceptual framework of the Comprehensive Service Management Middleware (CSMM) to enable complete or partial automation of the job of creating, executing and managing a Web services-based business process.

A detailed analysis of the state-of-the-art research on process management is presented in Section 2.3.1. Based on that analysis, we deem that it is necessary to have a more flexible, modular, and autonomic middleware framework, which can serve comprehensive or specific management requirements of the consumers without inflicting any bindings on the other management tasks.

In this chapter, we first describe the different tasks that need to be carried out to use a composite Web service process and some of the factors that are contributing to the growing complexity of these tasks. We then propose our approach to simplifying the tasks using the CSMM. We present a scenario to explain the usability of the CSMM. In conclusion, we summarize the contributions of our work with respect to some of the existing approaches.

3.1 Steps to Create and Execute a Workflow

There are several steps for using a Web service in a business process, which can add considerable overhead depending on the type and usage of services and the complexity of the process. These steps, as listed below, are common for all service consumers, however, the complexity of each step may vary.

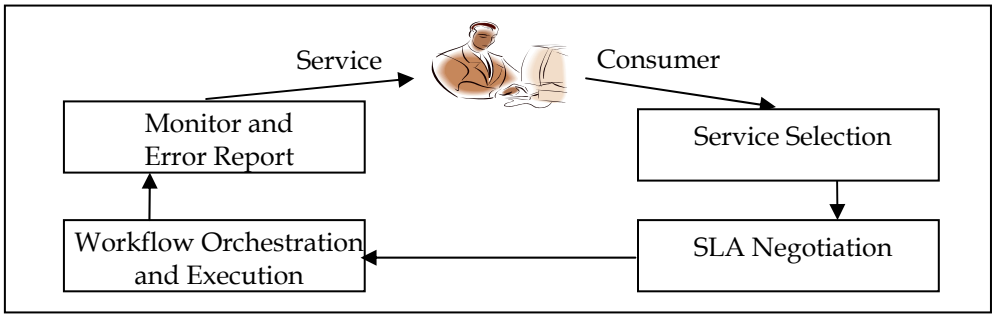

Figure 3.1 Steps to execute a Web services-based process

- Service Selection: Select a service based on some predefined criteria to complete a business process or replace a service to recover from failure.
- SLA Negotiation: Negotiate the Service Level Agreements (SLAs) based on customer requirements and service offerings.
- Workflow Orchestration and Execution: Design a workflow for a business process by organizing selected Web services in order with properly matched input and output parameters. Ensure possible error check points, alternative paths to handle exceptions, implement corrective measures, and thereby, execute the workflow.

- Monitor and Error Report: Monitor the services' performance to verify compliance with the SLAs and optionally report Quality of Service (QoS) information to a specified knowledgebase to enable quality-based service selection. Also check for possible failure to allow quick recovery.

Figure 3.1 shows four steps that we identify as the four main tasks to be carried out by the service consumer in order to compose and execute a Web services-based workflow.

3.1.1. Complexity and Challenges

Service Selection: Consumers first need to decide about what types of services are required to build the desired workflow and then look up in the UDDI [82] directory for the services that meet their selection criteria. The complexity of service selection varies with the complexity of the selection criteria. As multiple providers are currently providing similar services, additional selection criteria on non-functional properties [60], such as QoS and reputation, are considered to enable selection of more reliable and trustworthy services. The QoS ratings are optionally published by the service providers, while the reputation information is generally built from users' experience reports or by monitoring tools. If a service failure is detected in a workflow, replacement of the faulty service with a similar service can enable seamless execution of the workflow. Therefore, automatic service selection can provide an efficient scheme of recovery for service-based workflows. Some of the challenges in automatic service selection are listed below.

- Specification of process and service requirements.
- Specification of service offerings.
- Decomposition of process requirements into multiple sub-processes or tasks to allow selection of services from the UDDI that best match the requirements.

- Semantic matching of the functional and non-functional service requirements based on some ontology specification [78] .
- Semantic matching of the input and output parameters to allow the services to be linked to create a workflow.
- Context matching [68] [77] in service selection is also getting much attention in the area of ubiquitous and pervasive computing.

SLA Negotiation: In e-Business services are priced for their usage and a SLA is typically set up between the service provider and the service consumer to guarantee satisfactory service performance. The dynamicity of e-Business requires on-demand and efficient SLA negotiation. Time consuming and costly negotiation process can downgrade the ease and efficiency of executing business processes on the Web. Therefore, automated and efficient negotiation of SLAs on the Internet for business processes is an important research problem [12].

Service offerings should state the negotiable issues, functional and non-functional, such as price, quantity, date, availability of the service, throughput, response time, delay, and may include bonus offers. Bilateral bargaining type negotiations [130] over the Internet can consume considerable bandwidth and negotiation time [127]. The priorities for trade-offs [43] between different issues of negotiation and the service offerings may vary depending on the business goals and contexts of the negotiating parties. Following are some of the challenges in this area.

- Specification of fixed and negotiable issues in the service offerings.
- Specification of the preferences of each negotiating party, namely, the service consumer and the service provider, for negotiation.
- Specification of the protocol for message exchange for fully automated or tool-based negotiation, that is, how the offers and counter-offers are exchanged.

- Definition of the decision support system for automated SLA negotiation, which governs how offers are calculated, trade-offs are made and decisions are taken during the negotiation process.
- Specification of the SLAs upon successful negotiation.

Workflow Orchestration and Execution: It is necessary to orchestrate the selected services into a workflow and express it formally for efficient execution and monitoring. Independent services and sub-processes can be executed in parallel to reduce the total execution time. Parameters of the adjacent Web services in the workflow should be checked for compatibility and necessary type conversions. Check-points should be inserted for efficient fault and failure detections, and enabling speedy recovery. Some of the challenges in this area are as follows.

- Analysis of service dependency for efficient execution.
- Create a Web service orchestration with the selected services.
- Formal representation of the business process for monitoring and anomaly detection.
- Implementation of error detection and recovery measures for seamless execution of the workflow.
- Determination of the suitability of centralized vs. distributed execution of the workflow based on specific process requirements.

Monitor and Error Report: A process needs to be monitored on both the service providers' end and the service consumers' end to guarantee satisfactory execution and verification of the SLAs. Failure to meet the SLAs incurs penalties and jeopardizes the reputation of the service provider [45]. Due to the distributed nature of composite Web service systems and dependency on the network, monitoring of service performance on the client-side poses a very challenging problem. Distributed monitoring generally requires a coordinated framework to

setup performance monitoring at multiple check points, and to report the performance statistics to a concerned authority or management endpoint. A set of rules or policies may be specified for the analysis of the reports to detect the nature of failure and execution of proper recovery procedure. Some of the major challenges in this area include:

- Set up a distributed monitoring framework external to the service providers' systems.
- Get access to servers that provide the Web services for setting up monitoring check points to get accurate and network independent performance data.
- Specification of the SLAs and reporting the performance data to the data analysis modules for verification purposes.
- Design and set up measurement procedures for specific performance attributes. There are specific challenges in measuring the different performance attributes, such as reliability, response time, and service delay [59], some of which may not be possible to measure from external check points.
- Compilation of the analyzed results to decide about possible recovery actions.

3.2 The CSMM Framework

We propose the CSMM as a solution to the problems of client-side management. It contains several distributed modules and knowledge repositories as shown in Figure 3.2, which together provide complete management functionality as a Web service. Each of the four main modules in the CSMM presents a significant research area in Web services and the implementation of the CSMM will follow the definition of all its components. At

this stage, we present our hypothesis about potentially viable and effective approaches to implement the various modules of the CSMM.

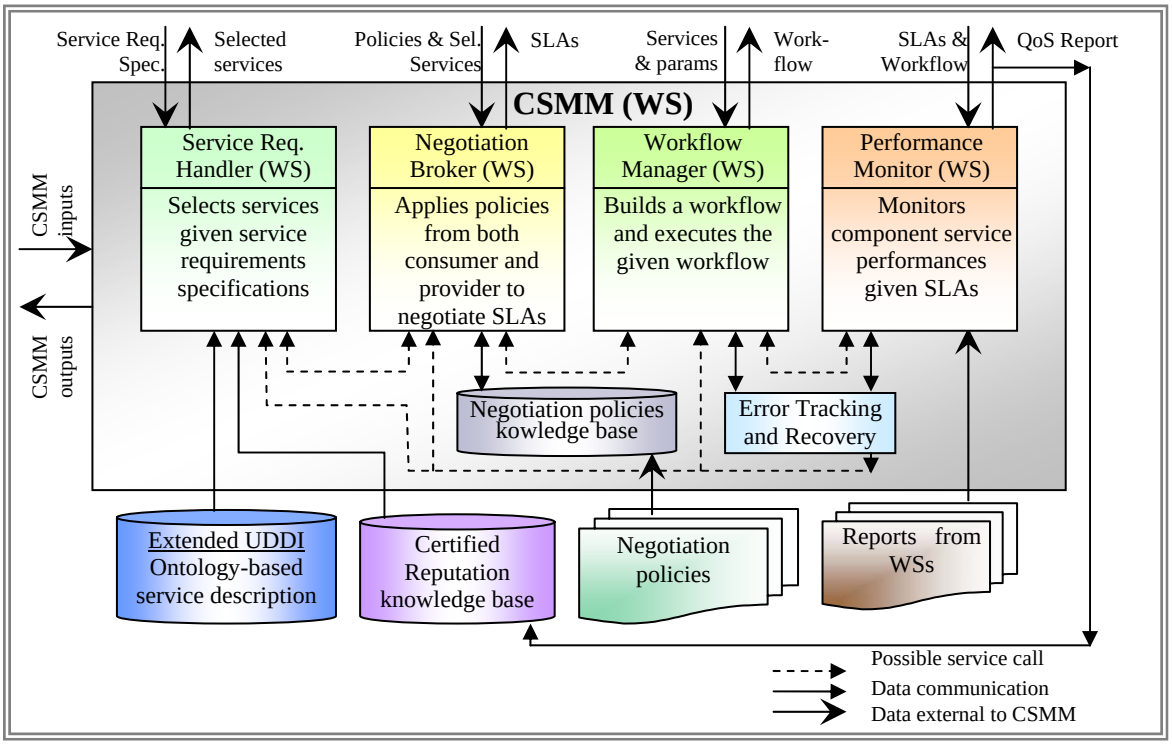

Figure 3.2 Comprehensive Service Management Middleware (CSMM)

In order to enable independent service provisioning by the four main modules in the CSMM, they are designed as Web services and provide services for one of the four main tasks of client-side process management as presented in the previous sections. The modular architecture of the CSMM allows the service consumers to selectively outsource one or more tasks to the corresponding modules of the CSMM, or alternatively, to request a comprehensive service of process management. We use the Autonomic Web Service Environment (AWSE) [142] framework to build the Web service modules within the CSMM in order to endow them with self-managing capabilities. We describe the various components of the CSMM below in further detail.

3.2.1. Service Requirements Handler

Service Requirements Handler (SRH) finds required services for the user based on some specified selection criteria. It accepts specifications describing process requirements in a formal language and returns a set of selected services in the order of execution. The language, which we will refer to as a Service Requirements Specification Language (SRSL), should be based on XML (eXtensible Markup Language) [87] and Web service ontology. SRSL is not specified in the thesis and is subject to future research. The Semantic Web Services group is currently working on the Web Ontology Language for Services (OWL-S) [125], which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of Web services in an unambiguous and computer-interpretable form. Other researchers are also working on semantic specification of Web services [11] [77] [135].

An *Extended UDDI* can be used to store semantic markup information about service offerings, properties, parameters, and return values for service discovery. A SRSL based on standards like OWL-S, can match the semantic service selection criteria against the information in the extended UDDI. To enable QoS-based service selection, we propose the use of a certified *Reputation Knowledge Base* as shown in Figure 3.2. In Xu *et al.* [126], we propose a reputation enhanced QoS-based service selection approach, where the reputation knowledge base is built from consumer feedback, and therefore, may contain anomalous data. The reputation knowledge base in the CSMM is very reliable as it is automatically generated by the monitoring module of the CSMM, which uses automated monitoring approaches.

The SRH is also used to find a replacement service to recover from failure caused by the unavailability or failure of a service during the execution of a workflow. When comprehensive service is requested, the SRH communicates directly with the negotiation module to initiate the next step for building and

executing the workflow. If negotiation fails to reach an agreed SLA for a specific service, the SRH can be invoked to find an alternative service.

3.2.2. Negotiation Broker

Negotiation Broker (NB) takes an ordered list of selected services and the negotiation policies from all the service providers and the service consumers. The policies specify the context of the negotiators, their goals, constraints, preferred values and priorities of the negotiable issues that may influence the decision process. A *Negotiation knowledge base* stores the negotiation policies, which can be used to derive improved negotiation strategies and provide assistance in the case of uncertainties in negotiation issues using artificial intelligence techniques. Also stored policies can be retrieved for subsequent negotiations between the same consumer-provider pair. We implemented the NB of the CSMM and it is described in detail in Chapter 4.

The NB performs the negotiation locally as a broker service and returns SLAs to both parties upon successful negotiation. This can reduce network traffic, and security issues in negotiation. However, the negotiating parties have to trust the broker to convey their goals and policies. We assume that the NB is a trusted service and WS-Trust [94] can be used as a guideline for the trust relationship. The NB can be extended to conduct more general and multi-party negotiations, and support multiple decision making strategies.

3.2.3. Workflow Manager

Orchestration describes how Web services can interact with each other at the message level, including the business logic and execution order of the interactions. These interactions may span applications and/or organizations, and result in a persistent, transactional, and multi-step process model. Workflow Manager (WM) takes an ordered list of selected services with the necessary input

parameters for each of them, and generates a Web Service Business Process Execution Language (WS-BPEL) [88] specification of a service orchestration. The workflow can be returned to the consumer to be executed locally or can be executed by the WM.

The WM is designed as a Web service to provide workflow composition and execution services independent of the other modules in the CSMM. The WM defines workflows with check-points for exception handling and monitoring purposes and executes them locally as the BPEL workflow engine. Centralized execution allows better handling of the exceptions and can be designed to re-invoke services with fewer constraints in the case of failure.

3.2.4. Performance Monitor

Performance Monitor (PM) takes the SLAs and workflow specification as input and performs the SLA compliance checking for the service consumer. The PM has two sub-systems: the main PM Web service and the secondary distributed monitoring check-points. The check-points of the PM measure performance data for each Web service in a workflow as it is executed, and send performance reports to the main PM Web service. The PM analyses the reports, verifies SLAs, and reports to the PM service consumers, which are typically the workflow executors.

Notifications are also sent to the *Error Tracking and Recovery (ETR)* sub-module when violation of an SLA is detected to initiate corrective actions. For example, the ETR can request the SRH to find a replacement service, the NB to negotiate SLAs, and the WM to build and execute the revised workflow with the replaced service.

The PM helps to build a certified statistical *Reputation Knowledge Base* from the statistical performance data to enable QoS-based service selection. Our implementation of the PM is illustrated in Chapter 5.

Monitoring Check-Points

We propose a distributed monitoring approach using the Message Interception technique for monitoring the performance of the component Web services in a workflow. Our approach uses the standard Simple Object Access Protocol (SOAP) [119] messaging protocol for Web services, and a custom handler of SOAP messages on the service providers' end. The handler intercepts all SOAP messages going into and out of the server, and thereby, measures the performance of the Web service in terms of its response time. We design a very simple custom handler with the assumption that it can be integrated into the standard SOAP message processing layer on the servers.

```
1  <soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
2    <tns:reportLog xmlns:tns="http://CSMM.server/xsd">
3     <tns:PID>1</tns:PID>
4     <tns:Manager_URL>http://localhost/axis2/services/PerformanceMonitor</tns:Manager_URL>
5     <tns:Response_Time/>
6    </tns:reportLog>
7  </soapenv:Header>
```

Figure 3.3 SOAP message sent to a service

When a Web service is invoked by a SOAP message, the header section of the message includes specific information for monitoring purposes. The custom handler on the server-side intercepts the message, extracts the monitoring information, measures performance data, and sends the report to designated receivers as defined in the monitoring information. Figure 3.3 shows an example of the monitoring information included in a SOAP message header in between the "reportLog" XML tags, which specifies the process ID or "PID", the attribute name that should be monitored, namely, "Response_Time", and the "Manager_URL" where the performance report should be sent.

Figure 3.4 shows the performance report. Within the "reportLog" section, "Service" refers to the service name, "Manager_URL" identifies where the report should be sent, "RepTime" indicates the reporting time, and "Response_Time" (as defined in Figure 3.3) shows the measured value in milliseconds, and

"Report" notifies "success" or "failure". The report contains minimal information to limit the network traffic and help trace the process workflow. In contrast Sahai *et al.* [104] present a heavier reporting approach where the status logs are added to the SOAP message incrementally as it passes along the process, and the final Web service would send it to the originator of the process. Although this approach features better message tracking, it incurs a greater message processing and network load.

```
1  <soapenv:Body xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
2    <tns:reportLog xmlns:tns="http://CSMM.server/xsd">
3      <tns:PID>1</tns:PID>
4      <tns:Manager_URL>http://localhost/axis2/services/PerformanceMonitor</tns:Manager_URL>
5      <tns:Service>SelectLocation</tns:Service>
6      <tns:RepTime>1029200613:11:06</tns:RepTime>
7      <tns:Response_Time>100</tns:Response_Time>
8      <tns:Report>success</ tns:Report>
9    </tns:reportLog>
10 </soapenv:Body>
```

Figure 3.4 SOAP message with reports

3.3 Autonomic Process Management

The CSMM provides a complete framework for Web services-based process management by enabling the process to manage itself without consumer intervention. Once the consumer provides a valid requirement specification, the process is configured, executed, and monitored by the CSMM framework with self-healing and implicit self-protecting capabilities. The SRH, the NB, and the WM enable the self-configuration feature by selecting, negotiating SLAs, composing, and thereby, executing the process. The self-healing feature is enabled jointly by the WM, the PM, and the ETR by monitoring and analyzing the performance data and deciding about the recovery action. Once the action plan is in place, all the components in the CSMM may be involved to execute the action.

There are a limited number of options for self-optimization in an ongoing process. The WM tries to optimize the workflow at configuration time by allowing independent services and sub-processes to run in parallel. The SRH and the NB enables selection of a service from a set of similar services based on the optimal service and process SLAs. Further optimization may be done by monitoring the current status of the process and the services to be invoked to be able to select the best available service dynamically during execution time.

The self-protection feature for the process is strongly tied to the self-protection feature of the CSMM as each module in the CSMM provides a broker service for managing the process. Therefore, trust and access control schemes should be implemented for all the service components of the CSMM. Additionally, the messages used for communication and execution of the process can be encrypted for higher security concerns.

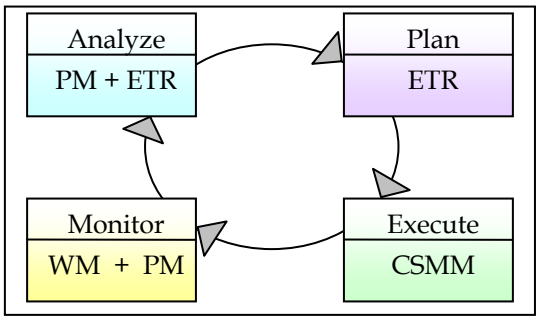

Figure 3.5 MAPE loop in the CSMM

The Monitor-Analyze-Plan-Execute (MAPE) loop for autonomic management [46] in the CSMM is shown in Figure 3.5. Monitoring is done by both the WM and the PM. The ETR gets error notifications from both these modules and analyzes the errors to plan for a recovery action based on predefined policies and/or knowledge base. The implementation and design of the ETR is in our future work plan. The PM also does some analysis of the performance reports and based on the SLA verification results of the component Web services of the process, it generates notifications for the ETR and the service consumers. Once

the action plan is decided, all the modules in the CSMM may have to act to implement the changes.

3.4 Example Scenario

We describe the CSMM with an example scenario of a consumer wanting to plan a vacation using Web services. This will typically require multiple services such as location selection, travel planning, hotel reservation, and tourist services, to be chained together into a composite service process. The CSMM can assist the customer in creating, executing and managing this composite service. Since service requirements specification is the key that guides the activity of the CSMM, it should be specified properly. An intelligent user interface with a knowledge base of possible services and service options can facilitate the specification. Figure 3.6 shows an example of some of the information that the service consumer need to convey to the SRH through a SRSL specification.

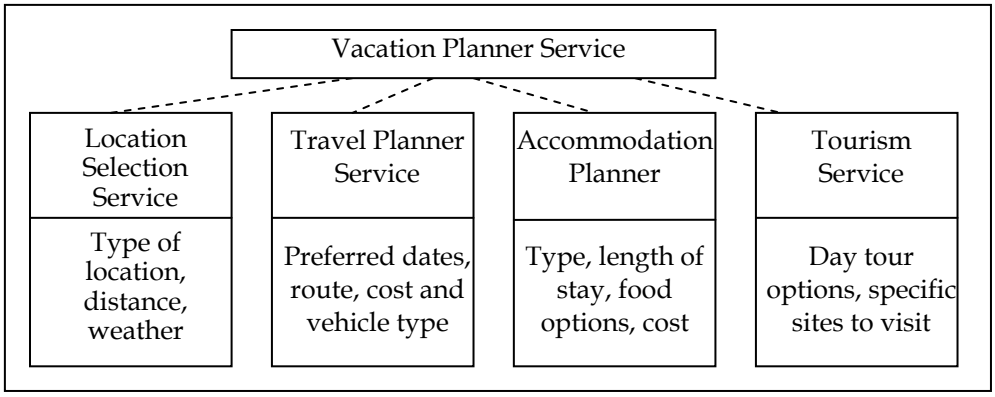

Figure 3.6 Service requirement specification basics

Once the SRH decides what types of services it needs for the process, it finds one service of each type for the specific task based on their reputation ratings, returned information and data types. If a service cannot be found, for example for location selection, the process is reanalyzed to find an alternative break down of tasks. For example, there may be a single service for location selection and

flight booking, which can be used. Selection of one service instead of two separate services also optimizes the execution of the process. The SRH returns selected services in the order of execution either to the customer if only the SRH service is invoked, or to the NB if the comprehensive CSMM service is invoked.

```
1  <wsp: Policy xmlns:wsp=
       "http://schemas.xmlsoap.org/ws/2004/09/policy"
       xmlns:nb="http:// CSMM.nb/policy">
2    <nb:NegotiationPolicy>
3      <nb:NegotiationContext = SelectLocation>
4        <nb:Role>ServiceProvider</nb:Role>
5        <nb:Service>Selects Location</nb:Service>
6      </nb:NegotiationContext>
7      <nb:Goals>
8        <nb:Goal>
9          <nb:Target>Maximize_Profit</nb:Target>
10       </nb:Goal>
11       …
12   </nb:Goals>
13   <nb:Issues>
14     <nb:Issue>
15       <nb:Name>Price</nb:Name>
16       <nb:Type>Decimal</nb:Type>
17       <nb:Unit>Dollar</nb:Unit>
18       <nb:Preference>0.8</nb:Preference>
19       <nb:Option>
20           <nb:Name>Gold</nb:Name>
21           <nb:Bestval>15</nb:Bestval>
22           <nb:Worstval>10</nb:Worstval>
23       </nb:Option>
24     </nb:Issue>
25   </nb:Issues>
     …
```

Figure 3.7 Negotiation policy specification for the "SelectLocation" service provider

The NB collects negotiation policies from both the service consumer and the service provider for each component Web service into the negotiation knowledge base. Figure 3.7 shows an example of the negotiation policy specification of the "SelectLocation" service provider. A detailed model and policy specification is given in Chapter 4**Error! Reference source not found.** for the NB. The policy specification shows the owner information in between the <NegotiationContext> tags (line 3). High level business goals are specified as <Goals> (line 7). The negotiable issues and options and their corresponding values are also specified in the policy by the <Issue> and <Option> tags (line 14 and 19). Each negotiation

issue may offer multiple options for different categories of consumers, such as gold, silver, or bronze. One of the options for the price issue for a gold member may offer $15 maximum price (line 21) and $10 dollars minimum price (line 22) based on the number of use for a certain period of time. The relative priority of an issue or option is denoted by the <Preference> tag (line 18). Based on the negotiation policy specifications, the NB selects appropriate negotiation strategy for each party, conducts negotiation locally and returns SLAs to both parties upon successful negotiation. If a negotiation fails, it either notifies the service consumer or the SRH as the case may be, to select an alternative service. When an alternative service is found, the NB does the negotiation and the cycle continues until a service is successfully selected with a set of agreed SLAs. Detailed functionality and the framework for the NB are presented in Chapter 4**Error! Reference source not found.**.

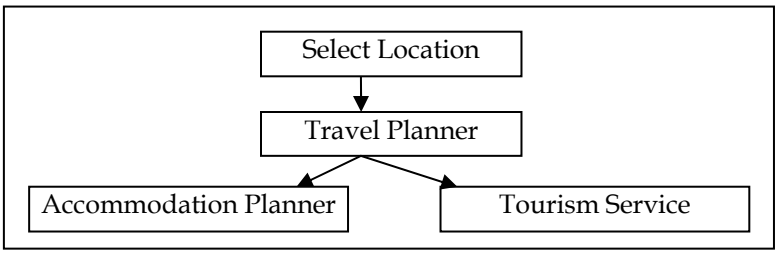

Figure 3.8 Workflow for vacation planning service

When a set of services are selected with corresponding SLAs for the process, the WM is called with the consumer choices, which are used as input parameter values for the services, the ordered list of the selected services, and the process information. Figure 3.8 shows a very simple and straight forward workflow of our example vacation planning composite service process. The arrows indicate information and control flow and the square boxes represent the services in the process. The WM builds a workflow with the selected services in sequence and the parameter values obtained from consumer preferences or previous services in the workflow. For example, output from the “SelectLocation” service is passed as the input to all other services. The travel service uses the location information

along with other user preferences to book travel arrangements. As the travel plans are made, the dates and location information are used by the other two services to book accommodation and plan tours. Since the latter two are independent of each other, they can be executed in parallel.

Before executing the process, the WM requests the PM for monitoring services. In response the PM returns a SOAP message to the WM in the form shown in Figure 3.3. This information is exactly copied in the message header when the WM invokes the component services during the execution of the workflow. In Figure 3.3 and Figure 3.4, the PM is designated as the receiver of the performance reports in line 3. However, the consumer can also choose to receive a copy of the reports. The PM obtains a performance report for each service as they are invoked during the execution of the workflow. A missing report indicates a failure or unavailability on the service's part. After a threshold period, if a report is not received the PM notifies the ETR to initiate corrective actions. In case of a successful execution, the location selection service would select a location, the travel planner service would book or buy tickets for traveling to that location, the accommodation planner service would book the hotel, and finally, the tourism service would book tours for the consumer.

3.5 Contribution

We propose the conceptual design of a modular framework for broker-based composite process management using existing Web services standards and protocols. As discussed in Chapter 2 and summarized in Table 2.1, most of the existing work in process management focus on only one or more of the specific management aspects, or provide an integrated service for multiple tasks. The approaches proposed by Liao *et al.* [72], Monge *et al.* [84], Yan *et al.* [128], and Zeid *et al.*[132] propose interdependent hierarchical or P2P agent-based frameworks for process composition and management. We divide the process

management into four independent tasks to facilitate independent service provisioning for any of those tasks. Each module can be customized to suit independent requirements and integrated with existing organizational systems management framework. Some of the existing approaches to service discovery or composition can be adopted for designing the SRH and WM of the CSMM. For example, the workload management approach of Pautasso *et al.* [101] can be used in the WM module. The semantic composite service search engines and SOA-based process modeling approach proposed by Dustdar [39] can be applied in the SRH module.

The modular design approach of the WSML of Cibrán *et al.* [28] is based on similar concept as our CSMM. WSML modularizes the various tasks by creating specific purpose Aspect beans on demand but requires client-code to be instrumented and services to be registered with the WSML. Our approach is based on more general standards and the client simply needs to invoke specific broker services as any other Web services as required.

We propose a flexible and AWSE-based framework, which provides comprehensive autonomic process management services. We assume the CSMM to be a trusted service provider. Each module is independent of the other, extensible and customizable. Consumers can optionally request services for a specific process management task, or request the comprehensive service when the modules coordinate to create and execute a composite process seamlessly. Moreover, the novel approach of architecting the middleware based on Web service technology leverages outsourcing of management responsibilities partially or completely as desired, and thereby, makes use of Web services most cost effective. The usability of the individual modules can be extended to provide other similar services. Although multiple solutions exist today for the different client-side management tasks, for coherent and seamless management functionality a coordinated management framework such as CSMM, is essential.

3.6 Summary

Based on the study of the state-of-the-art research on process management as outlined in Section 2.3.10, we first describe the necessity of a distributed framework for comprehensive client-side process management. We categorize the process management job into four tasks and highlight the challenges and complexity in each of these tasks, which represents an important research area in service management. Our CSMM framework is presented next as a solution to client-side process management, which provides four main broker services for the four main tasks. We also describe the concept of autonomic process management with regards to the CSMM framework. An example scenario is used to elucidate the function of each module in the CSMM. Finally, we describe the contributions of our approach in the context of the existing solutions.

Chapter 4

The Negotiation Broker

Performance of Web service is evaluated based on a contractual agreement called a Service Level Agreement (SLA) between the service provider and the service consumer. SLAs are important in business processes to maintain Quality of Service (QoS). Negotiation of SLAs poses a nontrivial problem for composite Web service systems where negotiations should be preferably automated, fast, time-bounded, reliable, and unbiased. We presented the state-of-the-art research on negotiation systems in Service Oriented Architecture (SOA) in Section 2.3.2 in Chapter 2 and analyzed the various approaches summarized in Table 2.2. Some of the approaches focus only on the core negotiation decision support model while others address the specification of the negotiation problem, definition of the negotiation framework, and communication between the negotiating parties for multi-party or bilateral negotiations. Existing negotiation systems and the approaches proposed by researchers do not satisfy all the requirements for SLA negotiation in a SOA.

In this chapter we present the Negotiation Broker (NB) framework for bilateral bargaining of SLAs between the service provider and the service

consumer in a SOA. The NB addresses the problems of specifications, automation with optional consumer feedback, and broker-based service provisioning in a flexible intelligent agent-based framework. We validate the framework with an experimental prototype and agent-simulations. In the sections to follow, we first briefly present common concepts in negotiation. Then we describe the three negotiation decision functions that we apply in our model followed by the methodology behind the design of the framework. Next we present the NB framework and our implementation of the model. We propose enhanced protocols and algorithms to support consumer feedback during the negotiation process, and demonstrate the ability of the NB to support multiple negotiation strategies. We conclude the chapter with a summary and an analysis of the contribution of our work with respect to some of the related work in this area.

4.1 Background

Negotiation has been an interesting area of research since the 1960s [102]. The journey started with theoretical models and evolved into computer-based Negotiation Support Systems (NSS) in the 1980s that provided tools to help human negotiators with computation for decision making [63]. As the network systems improved, standard protocols [44] were defined for communication of the offers through the network among the negotiating parties. With the evolution of the Internet, buying and selling products advertised on different websites raised the issue of conducting negotiations over the Web media, which inspired many researchers to work in this area. Negotiation systems were proposed to enable multi-party negotiations over the Web [109]. The advancement in Web technologies further drove the researchers towards automated negotiations [26].

E-commerce and e-marketplace have long been an active research area targeted towards designing efficient Internet marketplaces for buying and

trading commodities on the Web. Kasbah, eBay, SmartSettle, Negoisst, and CyberSettle ElectronicCourthouse [12] are examples of some of the commercial and research-based NSSs. Besides the e-marketplace, negotiation is used as a resource allocation technique in systems management, particularly, in grid systems; in resolving policy conflicts; in establishing agreements for authorized and secured access, and in SLA negotiation for services in cloud computing and network performance management. Currently, researchers are working on devising robust algorithms for predicting opponent's behavior to generate more effective offers [18], machine learning algorithms to negotiate under uncertainty [85], knowledge acquisition for adaptive negotiation [69], and negotiation languages [55], frameworks [109], and algorithms [54] for automated broker-based negotiations.

We describe some common concepts and terminology of negotiation and different negotiation types in this section. A brief overview is also presented of the various decision approaches that have been applied to negotiation by the researchers. Our current research builds on one of these negotiation theories and targets the application domain of SLA negotiation for Web services.

4.1.1. Common Concepts

Negotiation is generally called for when there exist common interests as well as differences among the negotiating parties. In order to resolve the differences and come to a common term, the problem and the solution requirements of the concerned parties at each end must be clearly defined, which includes the following:

- Negotiation issues that denote the items of conflict, for example, issues for a shipping Web service may be "price", "delivery type", and "weight".
- Available options for each issue, for example, "delivery type" issue may have two options, "expedite delivery" and "normal delivery".

- The acceptable range of values for each issue and option, e.g., $5-$15 for "price".
- The relative importance of each issue.
- The inter-dependency of the issues, for example, "expedite delivery" results in a higher increase in value of the "price" issue.
- The desirability of reaching a consensus, for example, 0.1 if not much interested, and 0.9 if highly interested in reaching an agreement.
- Constraints that define the unacceptable conditions, which can include the maximum time for negotiation or a combination of different values of the issues. For example, ("price" > $10 and "weight" < 2 lbs) is not an acceptable combination although individually the values of "price" and "users" may be within the corresponding acceptable ranges of values.

Apart from the above, *negotiation goals* and *contexts* can also influence the negotiation process and are typically used in a hierarchical management infrastructure. The higher level managers define the goals as business objectives for the company for specific negotiation and client contexts. For example, if the consumer is a "small business" or an independent user, then "maximize profit" may be the goal of the service provider to make the best out of a single business deal. However, if the service provider is a new company with an intention to expand its business, desirability may be higher with more uniform preferences for all the issues. When the consumer is a reputed multinational business organization, the goal may be "longer contract". Mapping rules can be defined to translate the higher level goals and contexts information to low level parameters that are used to execute the negotiation process. For example, when the goal is "maximize profit", the "price" issue will have the highest preference value of all the issues. Given a basic policy specification, it can be customized to suit the current negotiation goal and context information. For example, increase the relative preference value of the "price" issue or decrease desirability.

Two (bilateral) or more (multi-lateral) parties can be involved in a negotiation process. Communication among the negotiating parties during the negotiation process is done via messages. The messages follow certain rules and sequences, which are defined by a ***negotiation protocol***. For example, a negotiation may be initiated by a CFP (Call-For-Proposal) in answer to which the other parties send their corresponding offers in separate messages. Different types of negotiations such as auction, various types of bidding, and bargaining follow different protocols depending on the number of participants.

There can be one or more issues in a negotiation process. Each negotiating party follows a ***tactic*** to generate the next value to propose to the other parties for a single issue (e.g. price, availability, number of users) based on some criteria, which can include the maximum time for negotiation, preferences for specific values of an issue, priority of the issues, constraints that must be satisfied in the agreement, and desirability to reach a consensus [41]. The relative importance of the different criteria can also vary during the negotiation process. The way the tactics change over time during a negotiation process is called a negotiation ***strategy***. ***Trade-off*** is an important negotiation tactic where a party compromises with worse value of a less important issue in order to gain a better value for a more important issue. Decisions regarding the issues for trade-off and the values with which to compromise depend on the interdependency and relative preferences of the issues.

Based on the information provided by the negotiating parties, a ***Decision Support System (DSS)*** defines an appropriate ***Decision Model*** for each party, which in turn, applies various decision making tools and methodologies to define the tactics and strategy. A decision model basically represents the brain, which includes mathematical and logical models for generating offers and making decisions about the next action. A negotiation process terminates when the parties reach a common consensus regarding the values of the different

issues or a predefined terminating criteria is satisfied, which can be defined by the maximum time for negotiation or failure conditions.

A ***Negotiation Framework or System*** facilitates the negotiation process by supporting message exchanges between the negotiating parties, providing tools to generate offers based on various tactics and strategy, and defining decision models based on consumer preferences to guide the negotiation process i.e., help the consumer to decide about the next move. Specification of consumer requirements is mandatory for automated negotiation systems. The three main pillars of a negotiation system are *protocols*, *decision models,* and the *knowledge base*. The knowledge base stores data necessary to define negotiation protocols, tactics and strategies; generate negotiation outcome in a desired format, and enable intelligent decision making.

Researchers have proposed different negotiation frameworks and strategies based on the types of negotiations to suit different user groups, technology, and purposes. Negotiation systems that provide tools to facilitate the computation needed for decision making by human negotiators are called ***Negotiation Support Systems (NSS)***. A NSS typically includes a DSS that uses these tools or methodologies to decide about strategies for generating the offers, and resolving relative importance of the different issues for trade-off. Some NSSs also facilitate message exchanges between the negotiating parties over the network. With the advancement in Web media, automation of the negotiation process has received much attention. In ***automated negotiation systems***, an autonomous intelligent software agent is typically used to represent each party and conduct the negotiation process on a party's behalf. Specification of the preferences of each party for negotiation and interpreting the specification to map onto the agent's decision model is one of the major challenges in automated negotiation systems.

4.1.2. Negotiation Theory

The tools used for decision support in the NSSs vary in type and methodology. Most researchers emphasize mathematical support tools [12] [36] [130] such as decision trees, forecasting, and so forth. Others propose tools that address behavioral characteristics and cognitive perspectives of the negotiators [66] [12] [74]. Some of the commonly used techniques and strategies in negotiation systems are: economic or cost-benefit models defined by utility functions for evaluating goodness of an offer [12]; time-based functions where offers are calculated based on time only with a given maximum time and acceptable ranges of values of the negotiable issues [41]; statistical models for predicting probable next offers of the other parties based on the previous offer values [54], and various machine learning algorithms, such as genetic algorithms [12], game theory [43], and reinforcement learning [85], to learn other parties strategies from a known data set in order to make a more effective counter-offer.

Game theory is focused on achieving specified goals by making strategic moves considering the opponent's expected behavior [43]. A similar approach can be taken for negotiation to achieve particular objectives and deriving algorithms for automated negotiations. Machine learning approaches from the Artificial Intelligence paradigm have been used to design *intelligent agents*, which build their knowledge during the negotiation process in automated broker-based negotiation systems. Different learning methods and cognitive capabilities are incorporated in the negotiation strategies for the agents to acquire and update knowledge, and understand an opponent's moves in order to make intelligent decisions [66] [69] [85].

Genetic algorithms have also been applied to design negotiation systems. Generally, it starts with software agents that use various randomly generated negotiation strategies against each other under predetermined rules [12]. When the negotiation completes, it marks the end of a "generation". At that point, the

parent negotiation strategies are evaluated and using cross-over and mutation of strategies of different agents, new sets of child negotiation strategies are formulated. More successful strategies are chosen to be parents with a higher probability and other relevant factors such as initial population, number of generations, and crossover rate are chosen as parameters of the algorithm.

Decision theory provides decision makers with a wide range of instruments, which can be applied to different situations to uncover existing relationships and to help represent, analyze, solve and evaluate the decision problem [12] [66]. The selection and use of a specific method is, however, inherently subjective and guided by the decision maker's preferences. DSSs typically support the analytic perspective that constitutes the specification of needs and preferences, the specification of constraints and bounds, and the choice mechanisms defined on objectives, goals and preferences. Conflicts arise in negotiation when decisions have to be taken about alternative options where no decision can satisfy all the needs yet an option has to be selected based on priorities. Thus conflict resolution and constraint satisfaction are inherent perspectives of the decision making process in negotiations [109].

4.1.3. Types of Negotiation

There are three principal forms of negotiation, namely *bidding, auction*, and *bargaining* [12], in the decreasing order of simplicity. In bidding, the buyer specifies the product or service required and asks for bids from potential suppliers. Based on the bids, the buyer selects the supplier. In auction, a fixed auction protocol is followed, e.g. English auction, Dutch auction, and Vickrey auction. Auctions allow negotiation of only one issue, which is typically the "price". Bargaining is the most complex form of negotiation that requires multiple proposals and counterproposals to reach a mutual agreement or disagreement and can be bilateral or multi-lateral and involve more than one issue. Web service SLAs consist of multiple issues and involve the service

consumer and the service provider. Therefore, SLA negotiation for Web services is typically represented by bilateral bargain.

4.1.4. Complexity in Negotiation

Negotiation has several important characteristics that contribute to the complexity and difficulties inherent in developing adequate representations of the process. These include the involvement of two or more negotiating parties, each party having its specific requirements based on different perception of the problem and its solutions. As described in Section 4.1.2, the core decision model of a negotiation system can be based on a single theoretical foundation or a combination of approaches devised from multiple strategies. Selection of the appropriate strategy and tactic, mapping of the preferences of the negotiating parties correctly to the parameters of the mathematical decision models, and evaluation of offers based on consumers' choice of priority for different issues adds to the complexity of the negotiation system.

Convergence is a critical attribute of negotiation protocols for automated negotiations to guarantee the termination of the negotiation process. In electronic NSS that operate over the networks, messages are often coded because of security and privacy concerns, and this adds a new level of complexity. Repeated communication in bargaining over the network requires considerable time and bandwidth. The dependency on the underlying network creates an additional point of vulnerability in the negotiation system. However, it is greatly compensated by the benefits of connectivity and increased use of online negotiation systems particularly for bidding and auctions in e-marketplaces.

4.2 Time-based Decision Functions

Due to simplicity, time-based decision functions are often used in negotiation systems [41] [71], which vary from 0 to 1 with time. Three types of time-based

functions are most common: an exponential function expressed as in Eq. 4.1; a polynomial function, expressed as in Eq. 4.2, and a sigmoid function, expressed as in Eq. 4.3. The graphs corresponding to the above equations are shown in Figure 4.1, Figure 4.2, and Figure 4.3, respectively. In all three equations, *Alpha (α)* is a function of time, which increases from k to 1 as time increases from 0 to t_{max}. k, a constant value within 0 to 1, represents the initial value of $\alpha(t)$. The slopes of the graphs are dependent on a parameter β (labeled as "beta" in the legends) and reflect the conceding nature of the negotiating party. A higher β value results in a steeper curve and faster increase in α with time, which indicates a more conceding attitude of the negotiating party.

$$\alpha(t) = e^{\left(1 - \frac{min(t, t_{max})}{t_{max}}\right)^{\beta} \ln k} \qquad \text{(Exponential)}\ldots\ldots\ldots\ldots\ldots\ldots \quad \text{Eq. 4.1}$$

$$\alpha(t) = k + (1-k)\left(\frac{min\left(t, t_{max}\right)}{t_{max}}\right)^{1/\beta} \qquad \text{(Polynomial)}\ldots\ldots\ldots\ldots\ldots\ldots \quad \text{Eq. 4.2}$$

$$\alpha(t) = \frac{1}{1 + e^{-\beta * (t - t_{mid})}} \qquad \text{(Sigmoid)}\ldots\ldots\ldots\ldots\ldots\ldots \quad \text{Eq. 4.3}$$

In the above three equations,

$k \le a(t) \le 1$, where $0 \le t \le t_{max}$

β is the parameter to control the slope

k is the initial concession at $t = 0$ (constant) ($0 \le k \le 1$) and

t_{mid} is the position of the middle point of graph on x axis

For the exponential and polynomial equations, $0 \le \beta \le \infty$. β>1 represents a ***conceding*** tactic and β<1 represents a ***boulware*** tactic. In a conceding tactic, a party shows a compromising attitude from the beginning while in a boulware tactic the initial behavior of a party is very conservative, which changes quickly to a very compromising attitude towards the end of the negotiation time. In both Eq. 4.1 and Eq. 4.2, $lim_{\beta \to \infty}\ \alpha \to 1$ and $lim_{\beta \to +0}\ \alpha \to k$, the initial value of α at t=0. At α=1, the value of an issue reaches its maximum point of compromise, for

example, the maximum price a consumer can pay for a service. Therefore, if the acceptable ranges of values for all the issues overlap for the negotiating parties, they reach an agreement when simple time-based functions are used.

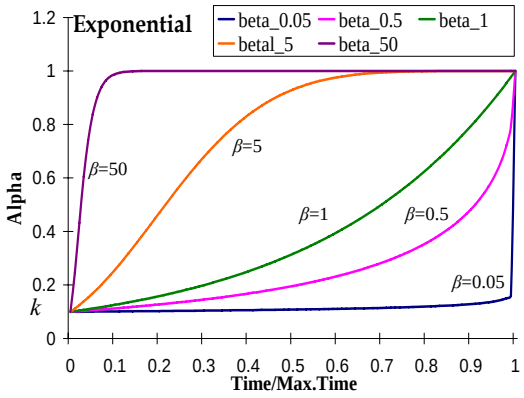

Figure 4.1 Exponential function

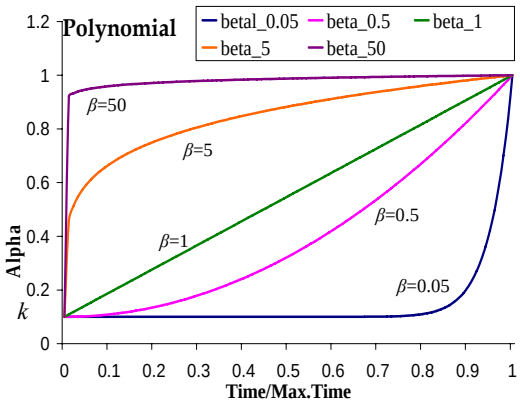

Figure 4.2 Polynomial function

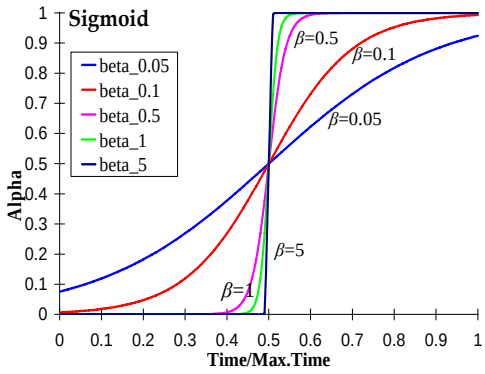

Figure 4.3 Sigmoid function

The sigmoid function is also known as a logistic function. It differs from the exponential and polynomial functions in a few aspects. It shows two transitions. From the graph, the initial boulware tactic changes to a conceding tactic towards its midpoint and then again becomes a boulware tactic towards the end. With a given maximum time for negotiation, sigmoid functions have comparatively less flexibility in terms of variation of slope because as $\lim_{\beta \to +0} \alpha(t) = 0.5$ between t=(0, t_{max}) and the curve becomes a straight line with constant value of 0.5. The minimum and maximum points of the curve occur at $t \to \pm\infty$, beyond the limit

of t_{max}. At β=5, α has a very steep increase around t_{mid}. Therefore, the range for variation of β is very small compared to the other functions. For the same reason, it is difficult to define an initial value of k for the sigmoid function for time-limited negotiations. In Eq. 4.3, t_{mid} shows the shift of the graph towards positive x-axis to indicate a logical time value > 0. For the above characteristics, sigmoid functions are less used as a time-based decision function for negotiation.

For each issue in negotiation, based on the priority of that issue and its range of acceptable values for each party, a time-based function is defined that guides the change of value of that issue in subsequent offers of that party. Negotiation strategy for a process defines the use of these functions to generate offers over the negotiation period.

4.2.1. Utility Function

The goodness of an offer is evaluated using utility functions. First a utility value is calculated for each issue in the offer. Based on the importance of the different issues, a normalized weight is used to combine the utility values of the individual issues to compute the value of an offer. Increasing values of an issue in subsequent offers may cause decreasing or increasing utility values of that issue for a negotiating party depending on the party's context. For example, if "price" value increases, a purchaser's utility value decreases but a seller's utility value increases. If a negotiating party defines a range of values $x_j \in \lfloor min_j, max_j \rfloor$ for an issue j, then the utility value for that issue is calculated using Eq. 4.4. The utility value for an offer comprising n issues is calculated using Eq. 4.5.

$$V_j(x_j) = \begin{cases} \dfrac{max_j - x_j}{max_j - min_j} & V_j \text{ increases as } x_j \text{ decreases} \\ \dfrac{x_j - min_j}{max_j - min_j} & V_j \text{ increases as } x_j \text{ increases} \end{cases} \quad \ldots\ldots\ldots\ldots \quad \text{Eq. 4.4}$$

$$V^{offer} = \sum_{1 \le j \le n} w_j V_j(x_j) \qquad \text{..... Eq. 4.5}$$

where:

V^{Offer} is the total utility value of the offer

w_j is the weight associated with j^{th} issue $\sum_{1 \le j \le n} w_j = 1$

$V_j(x_j) \in [0,1]$ is the utility value for the value x_j of the j^{th} issue

4.3 SLA Negotiation for Web Services

The increasing use of Web services in e-Business requires formalization and specification of business rules and agreements in order to ensure QoS of the Internet service subscriptions particularly for business processes. SLAs establish a formal contractual agreement between a service consumer and a service provider for a specific service context. A service provider can also be a service consumer for another service. Breach of the agreement on the service provider's end jeopardizes business reputation and incurs financial penalties [3] [53]. Therefore, it is important to negotiate the SLAs carefully and preferably automatically in e-Business for faster creation and execution of Web services-based composite processes.

Negotiations for Web services are typically carried out between a service provider and a service consumer as bilateral bargains involving one or multiple issues. A customer requiring this service would have to first decide on the issues and options based on preferences and service offerings before starting the negotiation process. Service offerings play an important role in SLA negotiation. Tosic *et al.* propose Web Service Offering Language (WSOL) [113] to describe service offerings containing the specifications of fixed and negotiable parameters, measurement units, and categories of services. The authors argue that a proper specification of the service offering can help negotiate, describe and categorize

service contracts (SLAs) based on the different aspects, such as functional, qualitative, and infrastructural aspects; highlight the service attributes that are negotiable, and manage Web service compositions efficiently.

An example of a 'stock quote service' offering is given in Table 4.1. Typically a Web service is offered as different priced packages, for example Gold, Silver, and Bronze. Each package may include some service parameters that have fixed values (response time) and some that have negotiable values (availability, price, and users), and other complimentary service parameters (bonus). While the service consumers may want a certain range of availability for a specific price range, service providers may also offer special discounts to specific categories of consumers based on their context information (e.g. large organizations, location) or business potential (e.g. economic value, length of contracts). An automated negotiation system [12] can be very effective in such cases where there are limited numbers of specific negotiable issues.

Table 4.1 Offerings for a Stock Quote Service

Option Type	Options in Offer	Package Offers		
		Gold	**Silver**	**Bronze**
Fixed	▪ Response Time ▪ Bonus	1 s. 10 free	2 s. 5 free	3 s. 0
Negotiable	▪ Price (per month) ▪ Number of users ▪ Availability	30$ - 50$ 500 (max) 98.9-99.9%	20$ - 30$ 350 (max) 97.9-98.9%	10$ - 20$ 200 (max) 96-97.9%

4.3.1. Our Approach

The existing approaches to Web services SLA negotiation, as presented in Chapter 2, do not cover all the aspects involved in SLA negotiation for autonomic process management. In most cases, user preferences are specified as low level time-based utility functions [30] or as rules specified by negotiation experts [71], which are difficult to customize for different organizations. The negotiation frameworks are integrated with the process management framework

making it difficult to only request the negotiation service [25]. Most frameworks do not take into account the changing status of the organizations during the negotiation. For example, a company may receive a large number of concurrent service requests during the negotiation process and should not settle for a SLA based on its initial status before the negotiations. A resource check before the final decision can help to establish more practical SLAs that a company can meet. Furthermore, the existing frameworks perform bargaining over the Internet [109], which requires each party to either maintain its own negotiation systems or use systems hosted by other parties. In most cases, the frameworks support a pre-specified negotiation strategy [41] [48].

We propose an autonomic trusted Negotiation Broker (NB) framework to facilitate automated SLA negotiation of Web services. The NB accepts negotiation requests with high level business policy specifications from the negotiating parties and performs local execution of bilateral bargaining using intelligent agents. Upon successful negotiation, a set of SLAs is returned to the concerned parties. The policy specifies business level goals, negotiation and consumer contexts, consumer preferences, and constraints, which are converted to appropriate negotiation strategies to be used during the negotiation. The proposed approach addresses the following issues:

- The negotiating parties do not have to provide or maintain their own negotiation system.
- The negotiation process takes place locally within the broker framework and thus network delays and insecurities are avoided.
- The NB accommodates a party's feedback during an ongoing negotiation to reach better decisions.
- A rich knowledge base can be created from negotiation histories to support intelligent negotiation strategies for generating more effective

counter-offers or dealing with uncertainties during the negotiation process.

- Collection and storage of negotiation policies within the NB helps avoid repeated collection of policy specifications for multiple negotiations.

The NB is one of the main modules of the CSMM framework presented in Chapter 3. Development of the NB framework is founded on negotiation theory, which has been studied for decades and includes various types of negotiation, communication protocol, messaging format, decision support systems, convergence strategy, and trade-off mechanisms. We use the Internet as the underlying communication network and agent technology for broker-based negotiation. Agents perform the negotiation process in the Negotiation Broker (NB) locally and impartially.

In the context of the CSMM, the NB communicates with the other modules within the CSMM especially the Service Requirements Handler (SRH) for repeated service selection and negotiation. The SRH selects a preliminary set of services based on the service offerings, and then the NB negotiates with each of these services to reach a SLA. If a service replacement is called for due to failure, the service selection and negotiation modules are duly notified by the Error Tracking and Recovery (ETR) module.

We use a policy-based approach to define necessary parameters for negotiation. Each of the negotiating parties provides the NB with a well-defined negotiation policy that contains goals, contexts, preferences, and constraints specifications. This information is used by a Decision Support System (DSS) to initialize the decision model and the strategy for an autonomous agent, which conducts the negotiation on behalf of a negotiating party using a pre-defined negotiation protocol. Since Web service SLAs are typically signed between a service provider and a service consumer, we use bilateral bargain in our NB framework. A notification about the result of the negotiation is sent to a negotiation manager, which converts the results either into a set of SLAs in case

of successful negotiation, or failure messages to send out to the respective endpoints of the negotiating parties. When the negotiation reaches a critical point, the agents can request an update of consumer preferences given the negotiation status or additional information about resource availability through an external resource manager. The information about ongoing negotiation can be collected and stored in a negotiation knowledge base to implement learning mechanisms to improve negotiation strategies. The policy information containing the context information of the negotiating parties also provides a very useful knowledge base for devising new rules for decision making based on contexts.

As described above, the process of negotiation is divided into three phases, *pre-negotiation*, *negotiation*, and *post-negotiation*. The *pre-negotiation* phase collects the negotiation policies and initializes the negotiation process by creating customized agents. The *negotiation* phase carries out the bilateral bargain negotiation process. The *post-negotiation* phase creates and sends reply messages to the negotiating parties.

In the following sections, we present our NB framework starting with the description of the framework, then the policy model that we define for the specification of consumer preferences. We define the negotiation protocol next followed by the DSS. Our implementation of the NB is illustrated in three stages. The initial implementation uses the exponential time-based function in a cost-benefit model. We subsequently propose our adaptive algorithm to accommodate consumer feedback during a negotiation process. The improvements achieved with the adaptive algorithm are justified by a measure of Combined Utility Value (CUV) for both parties involved in the negotiation since the negotiation represents a win-win situation. Finally, we show the flexibility of the framework in supporting multiple negotiation strategies and selecting the appropriate time-based strategy depending on consumer preferences.

4.3.2. The Negotiation Broker Framework

Figure 4.4 presents our Negotiation Broker (NB) middleware framework, which provides a trusted broker service for SLA negotiation through a *Negotiation Broker Web Service (NBWS)* endpoint. Either both negotiating parties invoke the NBWS with their corresponding policy specifications, or when a negotiation request is received, the NB requests the other party for its policy specification as a consent to move forward with the negotiation.

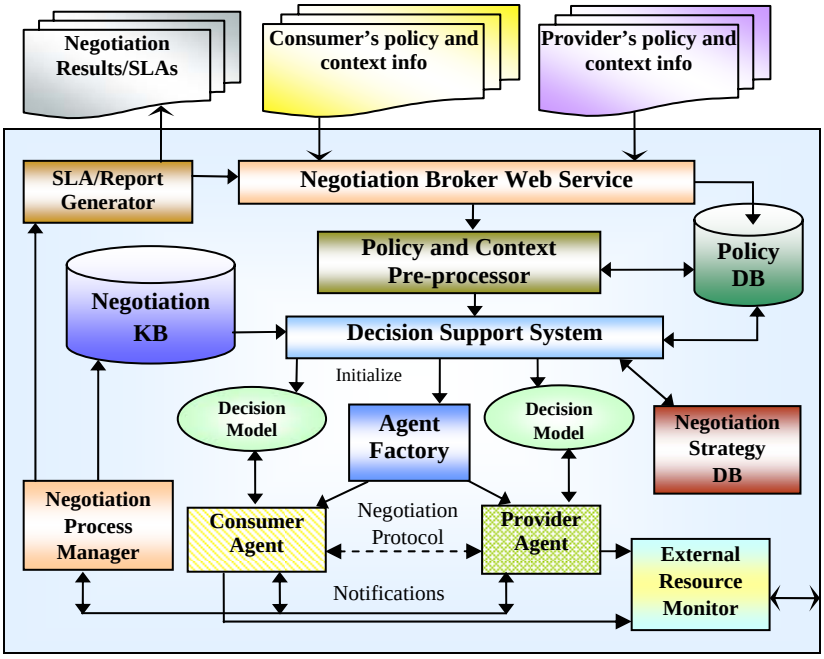

Figure 4.4 Framework of the Negotiation Broker

In the *pre-negotiation* phase, the negotiation policies received by the NBWS are processed by the *Policy and Context Pre-processor (PCP),* which stores the policies into a local *policy database (PolicyDB)* enabling easy retrieval and updates of policies. At this point, the *Decision Support System (DSS)* uses the pre-processed policy information from the PolicyDB, and if necessary, the negotiation history from the *Negotiation Knowledge Base (NegKB)* to choose an appropriate negotiation strategy from the *Negotiation Strategy Database (StrategyDB)* for each negotiating party. The DSS also initializes a *Decision Model (DM)* for each party with the

strategy and the policy information. The DM computes the parameters of the strategy model, and applies rules and algorithms to make decisions and generate counter-offers using the specified strategy. The rules may apply to goal-based decision making, trade-off, and adaptation of negotiation tactics and constraints.

Once the DMs are defined, the *Agent Factory (AF)* is used to create an autonomous agent for each negotiating party. The agent uses its DM to decide about the next move and counter-offers and communicates with the opponent's agent independently using the messages defined in the negotiation protocol. This approach reserves the privacy of policies provided by each party and enables impartial negotiation. A *Negotiation Process Manager (NPM)* receives notifications from the agents about the status of the negotiation process and the offers exchanged, which are stored in a NegKB. The *External Resource Monitor (ERM)* waits for notifications from the agents to communicate with the corresponding parties when the threshold values specified in the policies are exceeded during the negotiation process. The parties at this point have the opportunity to redefine the values of the negotiation issues, update the constraints, or guide the decision for the next step, which is a powerful feature of the NB framework.

The context information of the parties described in the policy can also be stored in a separate context database within the NegKB to use later as a reference to resolve ambiguity in case of uncertainty or to apply learning strategies. We assume that the context information provided by the parties is authentic since it is also used to define their personal negotiation strategies.

Upon completion of a successful negotiation, the NB enters the post-negotiation phase. The NPM sends the necessary information to a *SLA/Report Generator*, which generates a set of formal SLAs for the NBWS to send out to both parties.

4.3.3. Negotiation Policy Model

The NB accepts policies from the negotiating parties that define negotiation parameters namely: context, goal, issues, preferences, constraints, and other necessary metadata using a domain specific schema. Policies are basically sets of high level governing rules, which define assertions or actions to be taken when certain conditions are met. In other words, policies depict long term goals, and the preferred ways to achieve the goals, and thereby, guide the decision making process. The rules in a policy specification can be defined as blocks of if-then-else clauses that can be grouped together using "And", "Or", "Not". Relational operators (=, >, <, etc.) are generally used to express rules as equations. In the NB, rules from the policy specifications are used to select appropriate parameters for the initialization of the decision models and for constraint checking and decision making during the negotiation process.

Policies have long been used in the areas of network and resource management, security, and privacy. A number of XML-based policy and negotiation languages have been proposed by different researchers, which also include a number of rule-based policy languages [4] [26] [92] [118] [121] [125]. Most of these languages are designed for specific application domains and have specific requirements in terms of processing and usability. The standardization effort for a universal policy language is still ongoing (Policy Language Interest Group [118]).

We use the XML-based WS-Policy [121] standard in our NB framework because of its generality. Policy contents, however, can be very specific to different domains. Therefore, a *Domain Specific Schema* is used to define a domain specific policy within the WS-Policy framework. Data types, other than the basic ones, and tags are specified in the domain specific schema, which is referred to in the WS-Policy schema definition.

As shown in the UML (Unified Modeling Language) [95] class diagram in Figure 4.5, a WS-Policy specification can contain multiple Negotiation Policies for the different functionalities provided by a Web service. Each negotiation policy defines parameters for different negotiation contexts and can refer to the context of the other party with regards to the specification of negotiation rules. The negotiation parameters are explained below in further detail. Ontology can be defined to correspond to the schema definitions for a more general negotiation policy specification. An example specification of a subset of the Stock Quote service provider's policy is shown in Figure 4.6. All line numbers described in this section refer to Figure 4.6.

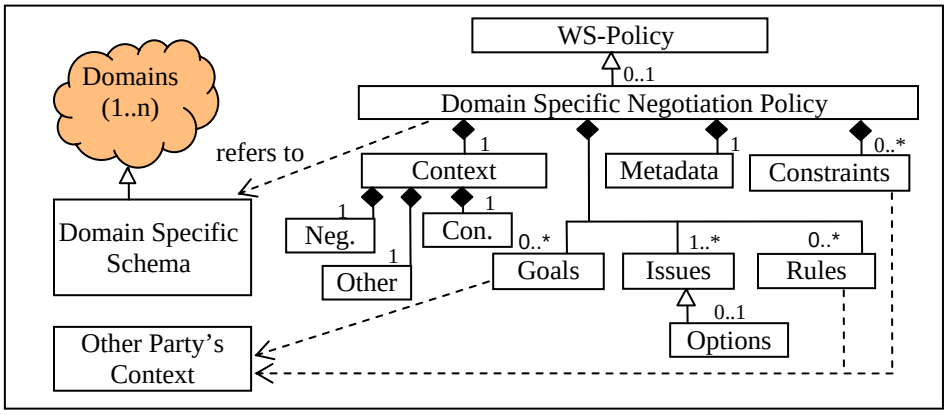

Figure 4.5 UML diagram of the contents of a policy specification

Context: Three types of context information is referred to in the policy specification; *Negotiation Context* (line 3 to 7) refers to the specific negotiation context (role of the party, a brief service description etc.) to which the policy applies; *Consumer Context* (line 62 to 66) refers to the party's own context information, and *Other Context* (line 11) refers to the context of the other negotiating party. Negotiation Context also contains a parameter called *Desirability Factor (DF)* (0<DF<1.0), which indicates the consumer's desire to reach an agreement in the current negotiation (line 6). In the absence of the DF specification, a value of 0.5 is used.

```
<wsp: Policy xmlns:wsp=
            "http://schemas.xmlsoap.org/ws/2004/09/policy"  xmlns:nb="http:// CSMM.nb/policy">
  <nb:NegotiationPolicy>
    <nb:NegotiationContext = StockQuoteService >
      <nb:Role>ServiceProvider</nb:Role>
      <nb:Service>Provides Stock Quotes</nb:Service>
      <nb:DesirabilityFactor>0.7</nb:DesirabilityFactor>
    </nb:NegotiationContext>
    <nb:Goals>
      <nb:Goal>
        <nb:Target>Maximize_Profit</nb:Target>
        <nb:OtherContext>personal</nb:OtherContext>
        <nb:Preference>0.7</nb:Preference>
      </nb:Goal>
      <nb:Goal>……</nb:Goal>
    </nb:Goals>
    <nb:Issues>
      <nb:Issue>
        <nb:Name>Price</nb:Name>
        <nb:Type>Decimal </nb:DType>
        <nb:Unit>Dollar</nb:DUnit>
        <nb:Preference>0.6</nb:Preference >
        <nb:Option>
          <nb:Name>Gold</nb:Name>
          <nb:Bestval>50</nb:Bestval>
          <nb:Worstval>30</nb:Worstval>
        </nb:Option>
        <nb:Option>……..</nb:Option>
      </nb:Issue>
      <nb:Issue>
        <nb:Name>Availability</nb:Name>
        <nb:Type>Decimal</nb:Type>
        <nb:Unit>Percentile</nb:Unit>
        <nb:Preference>0.4</nb:Preference>
        <nb:Option>
          <nb:Name>Gold</nb:Name>
          <nb:Bestval>98.9</nb:Bestval>
          <nb:Worstval>99.9</nb:Worstval>
          <nb:Thresholdval>99.5</nb:Thresholdval>
        </nb:Option>
        <nb:Option>……..</nb:Option>
      </nb:Issue>
    </nb:Issues>
    <nb:Constraints>
      <nb:Constraint>
        <wsp:Policy>
          <wsp:All>
            <nb:Condition>
              <nb:Issue>Price</nb:Issue>
              <nb:Operator>&gt;</nb:Operator>
              <nb:Value>40</nb:Value>
            </nb:Condition>
            <nb:Condition>
              <nb:Issue>Availability</nb:Issue>
              <nb:Operator>&lt;</nb:Operator>
              <nb:Value>99.4</nb:Value>
            </nb:Conditiont>
            <nb:MaxNegTime>100</nb:MaxNegTime>
          </wsp:All>
        </wsp:Policy>
      </nb:Constraint>
    </nb:Constraints>
    <nb:ConsumerContext>
      <nb:Location>Canada</nb:Location>
      <nb:EntityType>Company</nb:EntityType>
      <nb:Size>Medium</nb:Size>
    </nb:ConsumerContext>
    <nb:Metadata>
      <nb: PolicyName>…</nb:PolicyName>
      <nb:PDate>…</nb:PDate>
      <nb:ConsumerInfo>...</nb:ConsumerInfo>
    </nb:Metadata>
  </nb:NegotiationPolicy>
</wsp:Policy>
```

Figure 4.6 Negotiation Policy Specification

For example, a service provider's policy may state that a limited version of its service may be offered for a lower price to be used on mobile devices, or different prices apply for personal use and for business use. ConsumerContext refers to a party's location, type (company or individual user), size of the company or the number of employees who may be using the service, and so on. From the perspective of the service provider, a large company and a good credit record may seem to have more potential than an individual consumer with unknown credentials. Consequently, the DF will be higher when the other party is a large business organization than an individual consumer. So, the context of the other party, which is referred to as OtherContext in the policy, can influence the amount of interest in establishing a contract.

Goals: Each party in the negotiation has a goal. Typically the service consumer has a simpler goal than the service provider namely, subscribing to the service. High level goals such as maximize profit (line 10) or number of users, obtaining long or short term contracts, targeting large reputed companies or specific consumer groups based on location, age, or education, are based on the strategic business plans of the service providers. Service consumers may define short term contracts or lowest price as their goals. Each party in the negotiation can specify multiple goals (line 8). Based on the higher level goal, detailed policy specifications can include further details in the form of rules regarding how to achieve the goal. This kind of hierarchical organization of policy specifications can be very useful in large organizations where different levels of policy can be specified by an expert of the corresponding administration level.

Issues and Options: Issues (line 16, 17) are the negotiable parameters in a service offering and options (line 22) are the different values that can be taken by the negotiable parameters. The negotiating parties need to specify the best (line 24, 36) and worst (line 25, 37) acceptable values for each issue and option, the normalized preference weights to indicate the relative importance of the issues (line 21, 33), and optionally a threshold value (line 38). If during negotiation the

threshold value is exceeded, then an external resource, which can also be the negotiating party, is contacted for decision making. For example, to make an offer containing a response time that is beyond the threshold value specified by the service provider, the party is contacted to verify feasibility of the new offer.

Constraints: Constraints (line 43) are combinations of conditions (line 47, 52) that define unacceptable values of multiple issues and are declared explicitly in a policy specification to rule out options and narrow down the acceptable set of solutions. For example, a constraint can state a combination of values of different issues is not acceptable, such as price higher than $12 and number of users less than 50. The maximum time for negotiation is optionally set as a constraint (line 57). Otherwise, a default maximum time is chosen by the NB either as per the other party's time constraint (the lowest of the two), or the usual average negotiation time.

Preferences: The negotiating parties can define relative priority values (line 12) for the different goals when more than one goal applies, and for issues and options (line 21, 33) to facilitate the trade-off between the different issues. Typically numerical normalized weight values are used to indicate the relative importance of a set of issues where the sum of the weights equals to 1.

Metadata: The metadata (line 67 to 71) contains information about the consumer who invokes the service of the NB, and the name and date of the policy specification for easy reference to retrieve the policy from the NB repository if required.

4.3.4. Negotiation Protocol

Researchers have proposed several protocols [44] [71] [109] for automated negotiation purposes. However, they all refer to messaging over the network and therefore, contain more messages than what we need for our broker service. Figure 4.7 and Table 4.2 show a subset of the FIPA Contract Net Interaction Protocol [44], which is used in the NB. Most of the messages shown in Figure 4.7

are two-way i.e., any party can send the message to the other party. Messages are typically exchanged upon receipt of a message from the other party as described below.

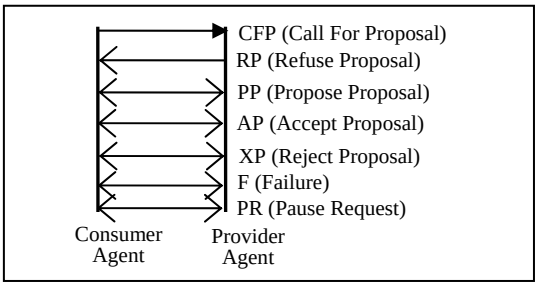

Figure 4.7 Negotiation Protocol

Call-For-Proposal (CFP) is used to request an offer from the service provider. In response, the service provider can either send a *Propose Proposal (PP)* or *Refuse Proposal (RP)*. Propose Proposal is also used for repeated offers and counter-offers in bargaining. Refuse Proposal indicates a party's unwillingness to participate in negotiation. A successful negotiation ends with *Accept Proposal (AP)* while failure to reach an agreement is decided by a *Reject Proposal (XP)*. *Failure (F)* indicates inability to receive, transmit, or interpret a message as explained in the message content, to which the other party may reply by resending the previous message. We define an additional message *Pause Request (PR)*, which is sent by an agent when values of issues cross the thresholds and the negotiation process needs to be suspended temporarily for user feedback. The other party waits for a default period within the specified maximum time for negotiation, otherwise it issues a "XP".

Table 4.2 Negotiation Protocol used in the NB

Call For Proposal	**CFP**	Request for service offering/proposal
Propose Proposal	**PP**	Send a proposal
Refuse Proposal	**RP**	Refuse sending a proposal
Accept Proposal	**AP**	Accept the proposal
Reject Proposal	**XP**	Reject the proposal
Failure	**F**	Any kind of failure notification
Pause Request	**PR**	Request to pause the negotiation process

4.3.5. Decision Support System

In this section we define our core Decision Support System (DSS). We later present advanced algorithms for an adaptive DSS. We apply integrative or collaborative negotiation in our NB framework where both parties reach a win-win situation [102] since both parties benefit from the negotiation. Game theory [43] assumes disclosure of the opponent's information and goal, which is not realistic in business negotiations. Genetic algorithms [12] have shown promising results in several instances but require considerable time for the whole process, which is not feasible in real world business processes. In our prototype we use a time-dependent cost-benefit model for the negotiation strategy for both parties.

Faratin *et al.* [41] describe time-based, resource-based, and behavioral-based negotiation approaches using different mathematical functions and cost-benefit models. We propose a similar time-based approach but resource-dependency is implemented as an external controlling factor through the definition of threshold values. Behavioral dependency can be implemented in different ways in the NB either through decision algorithms such as our adaptive algorithm, or through other AI-based behavioral approaches such as statistical regression analysis [54] or learning approaches [18] using the negotiation knowledge base.

The DSS in the NB selects an appropriate negotiation strategy from the Strategy DB and initializes two separate Decision Models (DM) for the two negotiating parties based on consumer preferences given in the respective policy specifications. The DMs compute the parameter values for their respective strategy and guide the corresponding agents, which carry out the negotiation process on behalf of the two parties. The DMs are also initialized with the constraints and rules, which are used with the decision support algorithms. These algorithms are used to decide about the next move in the negotiation process and to compute the values of the issues for the next offer or counter-offer. The two parties in a negotiation process typically have conflicting interests.

For example, a service consumer prefers a lower price whereas a service provider prefers a higher price. Issues for which both parties have mutual interests are excluded from the negotiation process by selecting the best possible values for those issues.

The three major aspects of a decision model are: convergence of offers towards either timed termination of the process, or acceptance or rejection; decision regarding the next action (accept, reject, or make counter-offer), and finally, definition of a mathematical model for generating counter-offers. The convergence aspect of negotiation in our model is satisfied by a timeout constraint to ensure termination of the process in a finite period. An offer consisting of values of n negotiable issues from an agent a to an agent b at time t is expressed as:

$$x^{t}_{a \to b} = \langle x_1, \ldots, x_n \rangle \text{ where } x_j \in [\min^{a}_{j}, \max^{a}_{j}] \text{ and } j \in [1, n]$$

The second aspect, i.e., the acceptability criterion for an offer, is defined using its utility value, which is a measure of the goodness of an offer. When a party receives an offer, Eq. 4.4 is used to compute the utility value corresponding to the value of each issue in the offer. The best and worst values of each issue as specified in the policy are used to determine whether V_j increases or decreases with x_j and the maximum and minimum allowable values of the issue. The goodness of the whole offer is measured using Eq. 4.5, where the weight values are obtained from the policy specification. An agent a infers its next action at time t' based on the offer received at time t $(t < t')$ using the logic in Eq. 4.6. If the maximum negotiation time is exceeded the agent sends an XP message and the process ends. Otherwise, if the other party's offer has higher utility value than the offer that would be made next at t', then the opponent's offer is accepted and the process ends. When none of the previous conditions are true, a counter-offer is proposed and the negotiation continues.

$$I^a(t', x^t_{b\to a}) = \begin{cases} \text{reject} & \text{if } t' > t^a_{max} \\ \text{accept} & \text{if } V^a(x^t_{b\to a}) \geq V^a(x^{t'}_{a\to b}) \\ x^{t'}_{a\to b} & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots \text{ Eq. 4.6}$$

where

$x^t_{b\to a}$ is the offer a received from b at time t

$x^{t'}_{a\to b}$ is the offer a should make to b at time t'

t^a_{max} is the constant maximum neg. time for a

The third negotiation aspect addresses the calculation of the counter-offer. Eq. 4.7 is used to compute the value of each issue to generate a counter-offer at time t from the maximum and minimum allowable values and the time-based function $\alpha^a_j(t)$ for the issue j and agent a. Any of the time-based functions from Eq. 4.1, Eq. 4.2, or Eq. 4.3 can be used to compute $\alpha^a_j(t)$. Different issues can use different time-based functions for generating an offer.

$$x^t_{a\to b}[j] = \begin{cases} min^a_j + \alpha^a_j(t)(max^a_j - min^a_j) \\ \quad \text{If } V^a_j(x_j) \text{ decreases as } x_j \text{ increases} \\ min^a_j + (1 - \alpha^a_j(t))(max^a_j - min^a_j) \\ \quad \text{If } V^a_j(x_j) \text{ is increases as } x_j \text{ increases} \end{cases} \quad \ldots\ldots\ldots\ldots \text{ Eq. 4.7}$$

Each of the time-based functions has a parameter β, which defines the curvature. Eq. 4.1 and Eq. 4.2 have a parameter k for the initial concession value and Eq. 4.3 has a parameter t_{mid}, which represents the shift of its midpoint towards positive x-axis. We define the equations to calculate the values of these parameters from the policy specification for the respective issue as part of the policy mapping model described below.

4.3.6. Policy Mapping Model

We define a policy mapping model to map consumer policy specifications to the parameters of the strategy model and to rules in the DM as shown in Figure

4.8. Based on the common goals and contexts, mappings rules are pre-defined in the PolicyDB of the NB. For example, the goal *maximize profit* will result in a very high priority and a narrow boundary for price. A goal *longer contract* will on the other hand, give low priority to price and high priority to *contract period*.

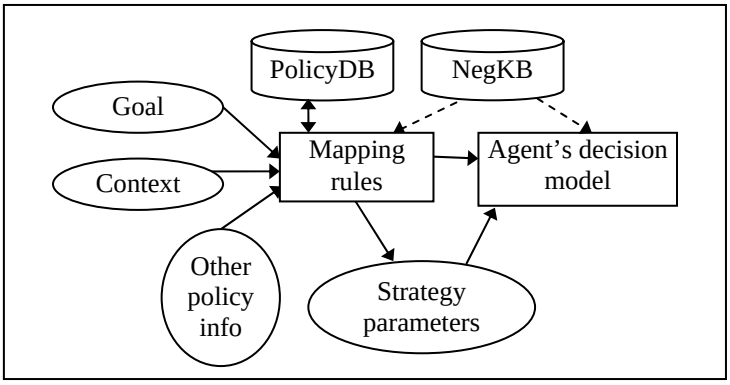

Figure 4.8 Policy mapping model

The mapping rules use information from the current policy specifications, the PolicyDB and the NegKB to derive parameters for the selected negotiation strategy model and rules that are used in the DM for strategic decision making during the negotiation process. When detailed information such as boundary values of the issues and preferences are given in the policy specification, mapping rules are only used to derive the parameters of the mathematical negotiation strategy model. In the absence of detailed information, if only goals are specified, previous policy information and goal mapping rules can be applied to obtain other necessary information for negotiation. Preference is taken into account to select appropriate policy mapping rules in the case of conflicting goals and constraints and previous cases can also be consulted from the NegKB. New rules can be formulated automatically based on the existing set of rules and information by applying various machine learning techniques for intelligent decision making, which are stored in the PolicyDB.

We define mathematical models for initializing the parameters of the time-based functions as part of the policy mapping model in our current research that

can be further customized based on the data from the NegKB. As shown in Eq. 4.1, Eq. 4.2 and Eq. 4.3, the following parameters need to be defined: β for the curvature, k for the initial concession value for the exponential and polynomial functions, and t_{mid} for the shift of the middle point for the sigmoid function. We calculate β and k for the exponential and polynomial functions using Eq. 4.8, Eq. 4.9, and Eq. 4.10, which have been revised from our earlier work [140] for better scaling of the parameters with the values of preferences and DF. β for the sigmoid function is calculated using Eq. 4.11.

$$adjustedPref = (1/numIssues) - pref \quad \ldots\ldots\ldots\ldots \text{Eq. 4.8}$$

where $numIssues$ = total number of issues

$$\beta_{exp\_poly} = \delta' * DF * e^{\delta * adjustedPref} \quad \ldots\ldots\ldots\ldots \text{Eq. 4.9}$$

where $(\delta', \delta) > 0$ are constants

$$k_j^a = (\lambda * max_j^a) \Big/ (min_j^a * t_{max}^a) \quad \ldots\ldots\ldots\ldots \text{Eq. 4.10}$$

where $\lambda > 0$ is a constant

$$\beta_{sigmoid} = \delta' * e^{\left(-1 * \delta * \frac{pref}{DF}\right)} \quad \ldots\ldots\ldots\ldots \text{Eq. 4.11}$$

where $(\delta', \delta) > 0$ are constants

Proper choice of the parameters provides the desired conceding patterns of the graphs. Generally, we are less conceding for more important issues and vice versa. A high β value results in a more conceding pattern whereas a low value of β (<1) results in smaller incremental values in subsequent offers. Therefore, the concept behind the equations is that the higher the preference the smaller β should be for the offers to be less conceding and vice versa. If β is too small, the changes in the offer values are very small. As a result, a long negotiation time is required. For exponential and polynomial functions, one party is likely to concede rapidly towards the end of the negotiation time (due to the nature of the curve), which results in less optimal negotiation outcomes.

We use exponential equations to derive β values for the decision functions. In the exponential equation $y=e^x$, $0<y<1$ for $x<0$; $y=0$ for $x=0$, and $y>1$ for $x>0$. *adjustedPref* is computed using Eq. 4.8. *adjustedPref*<0 when preference is greater than the equal-preference value (1/*numIssues*), and vice versa. As a result, for preference>equal-preference, Eq. 4.9 produces β values that are less than 1, and vice versa. The higher the preference, the lower the β values and the lower the preference, the higher the β values. For the same preference, a party is generally more conceding when the desirability is high. Therefore, β increases proportionally with DF in Eq. 4.9.

As shown in Figure 4.1 and Figure 4.2, β can vary in a large range (anywhere from 0.001 to 100) for a small preference range of 0 to 1. For better mapping of the β, we replaced the linear mapping function as shown in Eq. 4.12, which was proposed in our earlier work [140], with an exponential mapping function as shown in Eq. 4.9. We also replaced the corresponding Eq. 4.13 for computing k with Eq. 4.10, to get a better initial concession value for all ranges of values. k is calculated based on the highest and lowest acceptable values of the issues and the maximum negotiation time is scaled by the constant λ as shown in Eq. 4.10. However, any constant value, for example, 0.1 can also be used ($0<k<1$), without using the mapping equation, and then initial concession is 0.1 of the maximum concession. The values of the constants δ, δ', and λ in the above equations are selected by experiments for scaling purposes.

$$\beta = \frac{\delta * DF}{pref} \qquad \text{.....} \qquad \text{Eq. 4.12}$$

$$k_j = \frac{\lambda * (max_j^a - min_j^a)}{max_j^a * t_{max}^a} \qquad \text{.....} \qquad \text{Eq. 4.13}$$

An exponential mapping equation is used to compute the β parameter of the sigmoid function. In Eq. 4.11, β is minimum for high preference and low DF values. Low β results in a more slanted curve and low increment of α in the

subsequent offers. δ' and δ in Eq. 4.11 are assigned constant values. The maximum and minimum points of the curve of Eq. 4.3 should be very close to 1 and 0 respectively, and lie on the positive x-axis between t=0 and t_{max}, the specified maximum negotiation time. If the maximum and minimum points of the curve go beyond t=0 and t_{max}, then δ' is incremented while β is computed repeatedly until the curve lies within the specified time range. Then t_{mid} is calculated as the middle point of the curve on the time axis.

4.4 Prototype Implementation

We describe implemention of an agent simulation on a partial prototype of the NB framework to verify our approach to broker-based automated negotiation in three separate stages. In the first stage, we implement only the exponential time-based function with the basic decision logic of Eq. 4.6 and a linear policy mapping function for the parameters as shown in Eq. 4.12 and Eq. 4.13. In the second stage, we use the improved parameter mapping functions given in Eq. 4.8 to Eq. 4.10, and the enhanced decision algorithms for adaptive negotiation. In the third stage, we implement the polynomial and sigmoid time-based functions and show how the appropriate function can be selected based on the policy specification for better negotiation outcomes. For each stage we present the objectives, experimental setup and observations. Finally, we discuss the improvements made to the NB through the various stages of implementation.

4.4.1. Experimental Environment

We implemented an agent simulation in Java [62] to validate our approach to automated policy-based negotiation using intelligent agents. We used an IBM Intel Pentium 2.66 GHz desktop with 512 MB of RAM, Windows XP 2002 SP2, and JRE 1.4 [62] to develop and execute the simulation. The different modules in

the NB are all implemented as Java classes and Java thread and socket classes are used to simulate the negotiation agents.

To validate the impartial behavior of the agents although they run in the same framework, we describe below the organization of the Java classes used to implement the prototype. The `DSS` class represents the Decision Support System. It initializes two Java `DecisionModel` class instances for the two parties with their corresponding policy information. Then two instances of the `Agents` thread class are created using an `AgentFactory` class. Each `agent` is initialized with a pointer to the corresponding `DecisionModel` instance, which acts as the brain for the agent. The `Provider Agent` is initialized as a server socket, which immediately goes into the listening mode. The `Consumer Agent` is initialized as a normal socket and sends a CFP first before going into listening mode. The `Provider Agent` gets the CFP, uses the `DecisionModel` to generate its first offer and sends it to the `Consumer Agent`. The `Consumer Agent` replies back as decided by its `DecisionModel`. The implementation of the agents is therefore, independent of each other and guided by their corresponding `DecisionModels`. The `Issues` and `Options` are defined as sub-classes of an `Abs_Issue` Java class. Other classes are `Strategy` that implements various negotiation strategies, and `Constraints` that validates an offer based on the constraints defined in the policy.

4.4.2. Evaluation Criteria

Since the negotiation results in a win-win situation, we use *Combined Utility Value (CUV)* as a measure of the goodness of the negotiated offer in our experiments. CUV is the sum of the Utility Values (UV) of the negotiated offer for the consumer and the provider. The UV of an offer is calculated using Eq. 4.4 and Eq. 4.5. For example, if the negotiation result is $Offer_n$ the CUV is defined as:

$$CUV = ConsumerUV(Offer_n) + ProviderUV(Offer_n) \quad \text{..... Eq. 4.14}$$

4.4.3. First Stage

The first stage of implementation simply validates the basic organization and functionality of the framework using the exponential time-based function with the cost-benefit model. We applied the logical and mathematical decision support equations described in Section 4.3 and the policy mapping equations given in Eq. 4.12 and Eq. 4.13.

Objective: The experiments conducted at this stage test the viability of our approach to automatic policy-based negotiation by verifying if it meets the three important requirements of negotiation namely: convergence, offer generation, and decision support. We apply the basic decision models and mapping equations to validate our approach and the agent-based NB framework. We study the negotiation results to identify the limitations and possible areas of improvements.

Experimental setup: We ran the experiments for the Bronze service offerings of the provider as shown in Table 4.1 with DF=0.7 and maximum negotiation time=200 for both parties. We used δ=5 and λ=20 in Eq. 4.12 and Eq. 4.13. We assume that the best-worst value pairs for the issues "price", "users" and "availability" are respectively (10-15), (150-100), and (0.979-0.97) for the consumer and (20-10), (100-200), and (0.96-0.979) for the provider (Table 4.1). The preferences of the issues are 0.5, 0.2 and 0.3 respectively for both parties.

Observations: Due to the nature of the time-based decision functions, each party concedes to the opponent's offer towards the end of the negotiation process to reach an agreement, provided that the acceptable ranges of values of both parties for all the issues overlap. For example, if the provider's price limits are \$20~\$10 and the consumer's price limits are \$8~\$9.5, which do not overlap, the process terminates without an agreement at the maximum time. In any case, the negotiation terminates in finite time due to the maximum time constraint. Each party starts with its best offer with highest UV. Figure 4.9 shows how the

provider's offers gradually concede to the opponent's offers as the two lines representing UVs of the offers received and offers proposed converge.

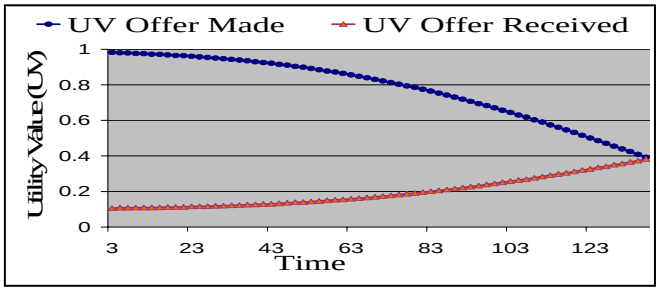

Figure 4.9 Provider Utility Value

Table 4.3 shows the actual values of the issues in the offers exchanged between the two parties. We consider time as discrete values that represent the count of the offers generated at a certain point of time during the negotiation process. The important thing to note in the table is that at time 61 and 62 both the offers contain user=134, which should ideally be in the final negotiated SLA. However, because each party continues to make counter-offers following only the time-based function without considering the opponent's offer, the final offer contains only 103 users. We address this issue in the next stage along with the implementation of the mechanism to allow consumer feedback during the negotiation process.

Table 4.3 Negotiation Process

Consumer's offers				Provider's offers			
Time	**Price**	**User**	**Avail**	**Time**	**Price**	**User**	**Avail**
2	10.085	149	0.978	3	19.736	102	0.960
50	10.315	139	0.978	51	19.150	125	0.960
60	10.406	135	0.978	**61**	**18.934**	**134**	**0.960**
62	**10.42**	**134**	**0.978**	63	18.885	136	0.961
100	11.03	115	0.977	101	17.528	172	0.964
136	12.13	103	0.973	137	15.268	193	0.971
138	12.218	**103**	0.973	* OA : Offer Accepted			

A third observation made during the experiments is that if any lower priority issue in the opponent's offer has a better value than the specified best value in

the policy, then we get a very high overall UV of the offer using Eq. 4.4 and Eq. 4.5 although other more important issues have very little UV. As a result, the offer gets accepted. For example, if the provider's offer has users=180 where for consumer the best value is 150, then although the more important issue price=10 (worst acceptable value), the offer may get accepted by the consumer agent, which results in a non-optimal solution.

4.4.4. Second Stage

In this stage, we mainly focus on the decision algorithms and protocols. Based on our observations in the first stage, we devise an improved version of the *Decision Algorithm* that is used to compute and compare UVs of the offers. We also present a new *Adaptive Algorithm* to adapt the time-based function to the changing status of the negotiation process. The adaptive algorithm allows the consumers to update their preferences when certain threshold values are exceeded during negotiation, which also results in a greater overall UV for both parties. Last, we present the *Agent Algorithm* that is used by the intelligent agents to respond to the offers received from the other party according to the specified negotiation protocol.

Decision Algorithm

Figure 4.10 shows our improved decision algorithm, which is executed upon receipt of an offer from the opponent. We refer to all the lines in Figure 4.10 in this section. As a remedy to the problem observed in stage 1 of incorrect computation of UV due to extremely good values of the issues, we add the check in the basic algorithm as shown in line 8 of Figure 4.10. For example, in the case an offer contains users=180 when the best allowable value is 150 for the party, Eq. 4.4 should be modified to use x_j=best allowable value, which should give $V_j(x_j)$=1. This approach guarantees correctness of the computation of the overall UV of the opponent's offer.

```
1   If (message == "CFP") then
2     Send first proposal
3   End If
4   If (ConstraintsDefined) then
5     CheckConstraints
6   End If
7   If (ConstraintsSatisfied or notDefined) then
8     If (value of an issue is better than the best value) then
9       Consider utility value for that issue to be 1
10    End If
11    Calculate utility value of the offer received (UVOR)
12    If (curTime > maxNegTime)
13       nextOfferTime = maxNegTime
14    else
15       nextOfferTime = curTime
16    End If
17    Calculate the offer to make next at nextOfferTime
18    Calculate utility value of the offer to make (UVOM)
19    If (values of all issues are within or better than acceptable limits) then
20      If (UVOR >= UVOM) then
21         Accept offer and send "AP" and return
22      End If
23      If (curTime > maxNegTime)
24         Send reject offer "XP" and return
25      End If
26    End If
27  End If
28  For (every issue i)
29      If   ((UVOR_i >= UVOM_i) and (UVOR_i > 0)  and (UVOR_i < 1) and notAdapted) then
30        Apply adaptive algorithm to redefine utility functions
31        Compute revised value of issue i in the next offer to make
32      End If
33  End For
34  Send next offer "PP" to the other party
```

Figure 4.10 Decision Algorithm

Line 4 checks the constraints if defined in the policy to confirm the validity of the offer received. Line 11 calculates the overall utility value of the offer received (UVOR) if values of all the issues are within acceptable ranges. Each party concedes to its worst value at maxNegTime (maximum time for negotiation). To consider this best offer from the opponent at curTime>maxNegTime, a little adjustment is made to time in line 12 to 16 to calculate party's own last offer. Lines 17 and 18 calculate the offer to make next and its utility value (UVOM). Based on the condition in line 19, the basic decision algorithm of Eq. 4.6, as given in lines 20 to 25, is executed. An offer is accepted if its utility value (UVOR) is higher than or equal to the utility value of the offer to make next (UVOM). Otherwise, if current time is greater than the maxNegTime then negotiation terminates without an agreement.

The basic strategy uses only the exponential time-based function to calculate the values of the issues for the next offer without taking the opponent's offer into consideration. Our earlier work revealed that such strategy results in lower utility values of less important issues. We add lines 28 to 33 in the decision algorithm to take into account the opponent's offer for generating the counter-offer. We compare the utility values of the offer received and the offer to make for each issue *i*, $UVOR_i$ and $UVOM_i$, and if the opponent's value is equal or better, we redefine the boundaries of the corresponding time-based function. Since each negotiating party starts with the best values of the issues and gradually concedes towards the worst values, the above strategy simply sets a new improved worst value limit.

In the example of Table 4.3, the consumer will set a new worst value of 134 for "users" at time 62. Once this new worst value is set, the utility values computed for the corresponding issue thereafter results in 0 based on the new worst value because already that value is reached. Therefore, we add the clauses ($UVOR_i > 0$) and notAdapted in line 29 in Figure 4.10 to avoid reapplying the change of limit. ($UVOR_i < 1$) is checked because we want to avoid changing the best value limit and when the opponent's value is better than the best value limit, $UVOR_i$ is set to 1. In line 30, provided all conditions are satisfied, the adaptive algorithm is applied to redefine the boundary values and parameters of the time-based function so that the $UVOM_i$ is as close as possible to $UVOR_i$. Based on the new function, next offer is recalculated and then sent off to the other party.

Adaptive Algorithm

We propose an adaptive algorithm as shown in Figure 4.11 to allow dynamic modification of the time-based functions during negotiation to adapt to changing status. It adjusts the time-based function for an issue with a new set of boundary values and updated parameter values to provide a certain value at a certain point of time during the negotiation process. This allows the example in Table 4.3 to

stick to a value close to 134 users in subsequent offers instead of settling for a less optimal value, such as 103 users.

```
1   If there is a new best or worst value then
2     Set the modified flag to true for this issue
3     Save the original limiting values
4     Set the new limit
5   End If
6   If current_time=0 then      // Parameter update is not necessary
7     Return
8   End If
9   If Strategy is exponential function then
10    Calculate β for α (current_time) = 0.999 from eq. 4.1
11    Set new β and return
12  Else If Strategy is polynomial function then
13    Compute α (current_time) as newAlphaT from eq. 4.2
14    Do
15      oldAlphaT = newAlphaT
16      Increment β by ((1 - oldAlphaT) * 50)
17      Set new β
18      Compute α (current_time) as newAlphaT
19    While (newAlphaT - oldAlphaT)>0.001
20    While ((1- newAlphaT) > 0.01 and (newAlphaT <> oldAlphaT))
21      Increment k by (1 - newAlphaT) * 0.1
22      Set new k value
23      oldAlphaT = newAlphaT
24      Compute α (current_time) as newAlphaT
25    End While
26  Else If Strategy is sigmoid function then
27    t_mid = ((double) time) * 0.5;
28    beta = ((-1)  * ln ((1/ 0.999) - 1)) / t_mid;
29    Set new β
30    Set new t_mid
31  End If
```

Figure 4.11 Adaptive Algorithm

Generally, β is incremented to a higher value so that $\alpha(t)$ reaches 1 and the corresponding issue reaches its new worst value at a given time which is less than t_{max}. In Figure 4.11, line 10 uses a value for $\alpha(t)$ which is very close to 1 to compute β using back calculation because $\lim_{t \to t_{\max}} \alpha(t) \to 1$ towards the end of the negotiation in Eq. 4.1, Eq. 4.2, and Eq. 4.3. For the example in Table 4.3 at time 62 the consumer would apply the adaptive algorithm to reset its lowest acceptable value to 134 and to recompute β such that at time 62, the next offer has users=134. Since this is the lowest value, it is maintained in the subsequent offers unless the opponent proposes some other value of "users" that has better UV than 134 with respect to the original boundary values. Then the boundary is reset to another new worst value.

For the polynomial time-based function, back calculation is not feasible. Therefore, we gradually increment β to make $\alpha(t)$ close to 1 where t represents the current time in the negotiation process. We noticed that after a saturation point, large increments in β results only in a very negligible increment in $\alpha(t)$. As a result, we need to increment the initial value of the function, k, to make $\alpha(t)$ close to 1. This is shown in lines 20 to 25 in Figure 4.11. In the case of sigmoid function, the maximum value of $\alpha(t)$ is considered to be 0.999, where t represents the current time and t_{mid} is (0.5 * t). From Eq. 4.3, we back calculate the value of β. Then the old parameters are replaced by the newly computed values.

A second contribution of this algorithm is in enabling the consumer to redefine preferences and constraints during a negotiation process when an offer exceeds pre-defined threshold values, which is a powerful feature of the NB framework. For example, if a consumer defines a threshold value of 99% availability in the policy, then the consumer is notified when the offer to propose to the other party contains a value that is greater than this value. Thus the consumer can reconfirm the availability of resources to provide this QoS. The consumer at this point can reply back with a revised 98% availability limit. The adaptive algorithm then adjusts the time-based function to reflect this new limit. A new message type, *Pause Request*, is proposed in the negotiation protocol to pause the negotiation process for this purpose.

A third advantage of the algorithm is that it allows the best value limit to be modified when the opponent's offer contains better value of an issue than its best value.

Agent Algorithm

Figure 4.12 shows the algorithm executed by the intelligent agents when they receive an offer from the other party. In most part the algorithm is self-explanatory. We exchange acknowledgement messages in the case of accept proposal message, "AP". When a party receives an "AP", as shown in line 4, it

checks if the last message sent was an "AP". If so, it deems this message as a confirmation and terminates the process. Otherwise, it sends another "AP" to confirm the receipt of the "AP" and terminates.

```
1   If ("RP" or "XP") then          //Reject or Refuse proposal
2      Terminate negotiation process and return
3   End If
4   If ("AP") then      //Accept proposal
5      If (LastProposalSent== "AP") then   //acknowledgement
6          Accept and terminate
7      Else
8          Send "AP" as acknowledgement of "AP"
9          Terminate and return
10     End If
11  End If
12  If ("F") then        // Failure
13     Retry sending the last message up to a maximum number of
           retries and then send "XP" and return
14  End If
15  If ("CFP" or "PP") then    //Call for or Propose proposal
16     Consult DecisionModel to find the next move and send that
           message to the other party
17  End If
```

Figure 4.12 Agent Algorithm

Objective: With the new and improved algorithms and the policy mapping functions given in Eq. 4.9 and Eq. 4.10, we first demonstrate our selection of the optimum values for the constants δ, δ' and λ by rigorous experiments. We also show how the negotiation time varies with these values. Then we validate the effectiveness of the adaptive algorithm by comparing the CUVs of the negotiation outcomes with and without use of the adaptive algorithm.

Sometimes the opponent's offer contains a better value than the best value of an issue for which we consider UV=1 as explained before. In our current algorithm we do not change the best value limit because it is beyond the acceptable range of values for the issue. When the adaptive algorithm is used, for the new worst values UV is calculated as zero if we use Eq. 4.4. For example, if "users" increases consumer's UV increases, and for the new worst value, which is also the current value of the issue (x_j), Eq. 4.4 gives $V_j(x_j)$=0 although the new worst value is better than the old worst value. Therefore, we calculate the UV using the original boundary values ($orgmax_j$ and $orgmin_j$) as given in Eq. 4.15.

$$V_j(x_j) = \begin{cases} \dfrac{org\,max_j - x_j}{org\,max_j - org\,min_j} & V_j \text{ increases as } x_j \text{ decreases} \\ \dfrac{x_j - org\,min_j}{org\,max_j - org\,min_j} & V_j \text{ increases as } x_j \text{ increases} \end{cases} \quad \text{..... Eq. 4.15}$$

We prove the following hypotheses by the experiments:

- The new mapping equations given in Eq. 4.8 to Eq. 4.10 provide better result and higher CUV than the equations proposed in our earlier work.
- The proposed policy mapping model provides a feasible, effective and automatic way to map high level policy to low level strategy model.
- Parameters computed using the mathematical equations correctly reflect user preferences in the negotiated results, i.e., more preferred issues have better values than the less preferred issues.
- The adaptive algorithm improves the general performance of the NB in terms of the CUV besides enabling the parties to provide their input during an ongoing negotiation.

Experimental setup: We use the same range of values for the issues for both parties as in the first stage of experiments. We use Eq. 4.9 and Eq. 4.10 with $\delta=\delta'=5$ and $\lambda=20$, respectively, to compute the parameters of the exponential time-based function for all the experiments in this stage. To observe the effect of preference and DF on the parameters and the negotiation result, we conducted experiments for different values of DF and preference (of one issue) for one party while keeping the same DF and preference for the other party. We varied the consumer DF and the preference of the "price" issue from 0.1 to 0.9 with equal preference for the other two issues. For the first set of experiments, the provider agent has DF=0.5 and all issues have equal preference. To observe the performance of the adaptive algorithm for different values of DF and combination of preference of both parties, the second set of experiments is

conducted where we vary the DF and the preference for the "price" issue for both parties with equal preferences for the other two issues. In most of the experiments presented in the next section, maximum negotiation time=100.

Observations: We conducted the same experiment as shown in Table 4.3 with the revised Eq. 4.9 and Eq. 4.10. Table 4.3 shows that with the old equations the consumer agent accepted the offer at time 138 for {price: 12.218, num_users:103, availability: 0.973} and CUV 0.78. With the new mapping equations but without the adaptive algorithm the consumer now accepts an offer at time 75 for {price: 11.93, num_users: 104, availability: 0.973} and CUV 0.79. The new equations, therefore, provide better performances in terms of CUV and negotiation time. With the adaptive algorithm the consumer accepts an offer at time 123 for {price: 13.19, num_users: 135, availability: 0.973} and CUV 0.8. Therefore, the adaptive algorithm provides an even better result in terms of the CUV although a longer negotiation time is required than when the algorithm is not used. The adaptive algorithm controls the highly conceding nature of the less preferred issues in order to gain a better overall CUV, which comes at the cost of a slightly worse value of the more preferred issue. In the above experiment, the CUV is highest with the adaptive algorithm at the cost of a higher "price". The result contains 135 users instead of the obvious value of 134 with the adaptive algorithm because with the new equations, after receiving 135 in the offer, the consumer agent computes 135 in the counter-offer to make. Since both have same UV, the new worst value limit is set to 135 at this point, which is maintained in the final result.

We now justify our choice of the constant values in the mapping equations. We used δ=5 and δ'=5 in Eq. 4.9 and λ=20 in Eq. 4.10 for all experiments in this stage. These values are chosen based on an experimental study of the negotiation outcomes, which vary the most with the value of δ'. Figure 4.13, Figure 4.14, and Figure 4.15 show the comparative results in terms of total negotiation time in milliseconds (ms), the variation in β values, and the CUV of the negotiation

outcomes for δ' values of 5, 3, and 1.5. We also varied consumer price preference and DF from 0.1 to 0.9 with equal preferences for the other two issues. We set DF=0.5 and uniform preferences for all the issues for the service provider. The label Ad_5_Pr_0.1 on the x-axis denotes use of adaptive algorithm, δ'=5 and consumer preference for `Price`=0.1. Figure 4.13 and Figure 4.15 show that with δ'=5, negotiations converge in around half the time than with the other values of δ' with insignificant differences in the CUV. Figure 4.14 shows that the β ranges from 0.02 to 14.5 as the consumer's preference and DF varies from 0.1 to 0.9.

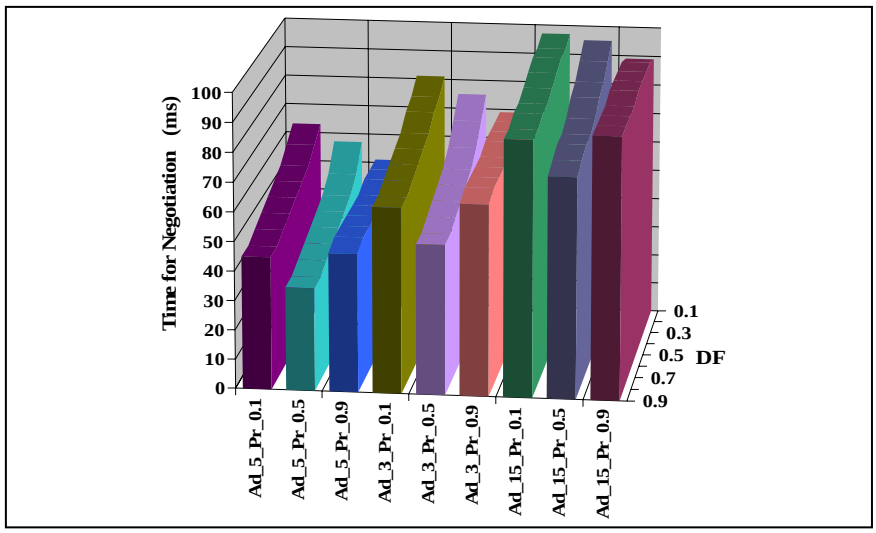

Figure 4.13 Time for negotiation for different values of δ'(5, 3, 1.5) and preferences of the price issue (0.1, 0.5, 0.9) and DF

Low preference implies a high conceding nature but results in low UV. High DF increases the conceding nature even more. In Figure 4.15, therefore, CUV is low at pref=0.1 and high DF. As DF decreases, the consumer is less conceding and the CUV goes up. At pref=0.9, the consumer is very conservative and the CUV is high on average. At DF=0.5, both parties have equal DF and with high preference, the CUV reaches its peak. At pref=0.5, which is closest to the provider's equal preference value (0.33), DF has greater influence on the result. At high DF, the greater conceding nature of the consumer results in low CUV. As DF decreases the party becomes more conservative and the CUV increases for

the same reason. When both parties have similar and equal preferences, we observe that the CUV is generally low, which is also prominent in Figure 4.16.

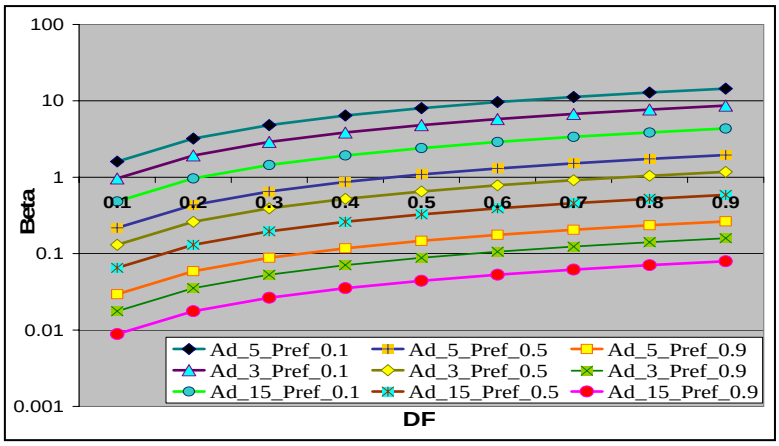

Figure 4.14 Mapping of parameter β for different values of δ'(5, 3, 1.5) and preferences of the price issue (0.1, 0.5, 0.9) and DF

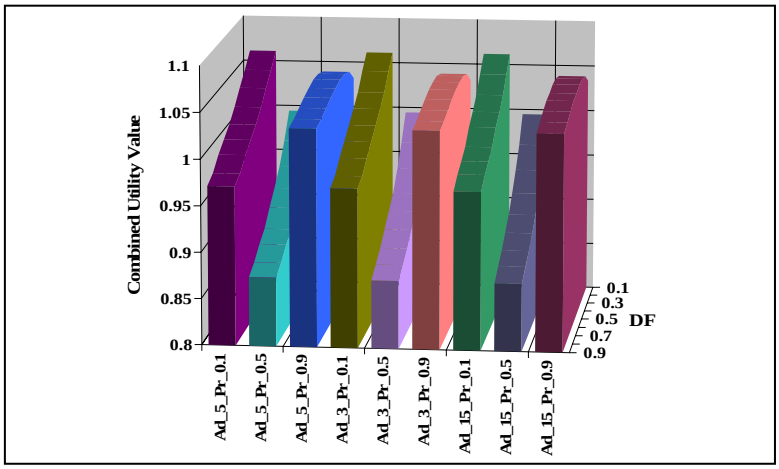

Figure 4.15 Combined Utility Value (CUV) for different values of δ'(5, 3, 1.5) and preferences of the price issue (0.1, 0.5, 0.9) and DF

Figure 4.16 shows the effectiveness of the adaptive algorithm for lower preference values for exponential function. The graphs show the CUV for different consumer DF and preference values for negotiations with and without the adaptive algorithm. For low preference the CUV is higher for the same DF with the adaptive algorithm than without the algorithm, but as the preference increases negotiations without the adaptive algorithm perform better. The CUV

is low around the equal preference value for all DF values. Without the adaptive algorithm, for higher DF values as preference increases the CUV has a greater increase. The reason is that the adaptive algorithm restricts trade-off for issues with lower preference to gain better values for issues with high preference.

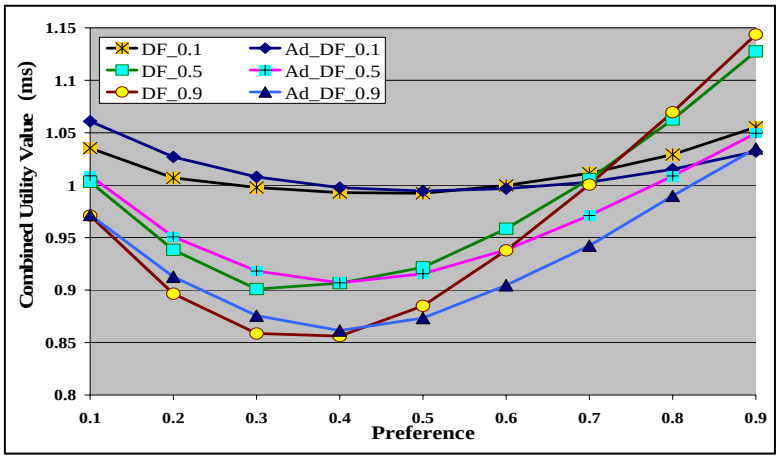

Figure 4.16 Combined Utility Value (CUV) for different preference values of the price issue

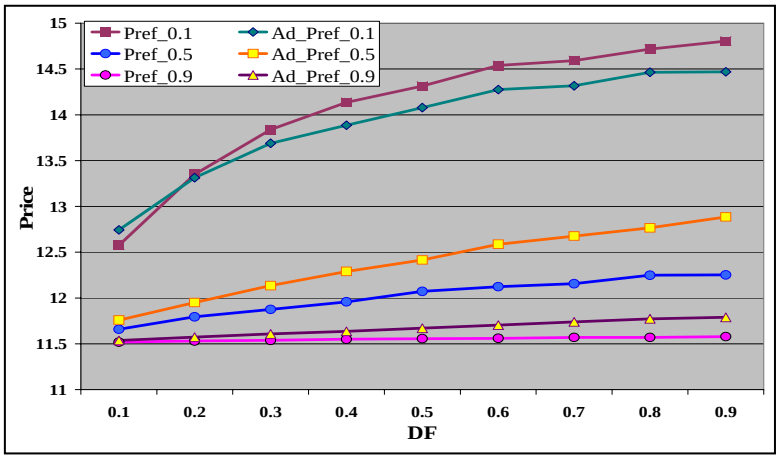

Figure 4.17 Negotiated price for different DF and preference values

Figure 4.17 shows the final negotiated prices for different consumer "price" preference values for negotiations with and without the adaptive algorithm. The DF_0.1 and Pref_0.1 legends stand for DF=0.1, pref=0.1, respectively and that the adaptive algorithm is not used, while Ad_Pref_0.1 implies that the adaptive algorithm is used in negotiation. In Figure 4.17 we see that for very low preference "price" is lower with the adaptive algorithm than without the

algorithm. Therefore, the adaptive algorithm provides a remedy for the second observation in the stage 1 experiments. For higher preferences the difference in performance for the two cases (with and without the algorithm) reduces. The experiments verify that the adaptive algorithm can effectively adapt the time function to new boundary value settings. From Figure 4.16 and Figure 4.17 we can additionally infer that the algorithm provides better CUV at lower preference and DF values.

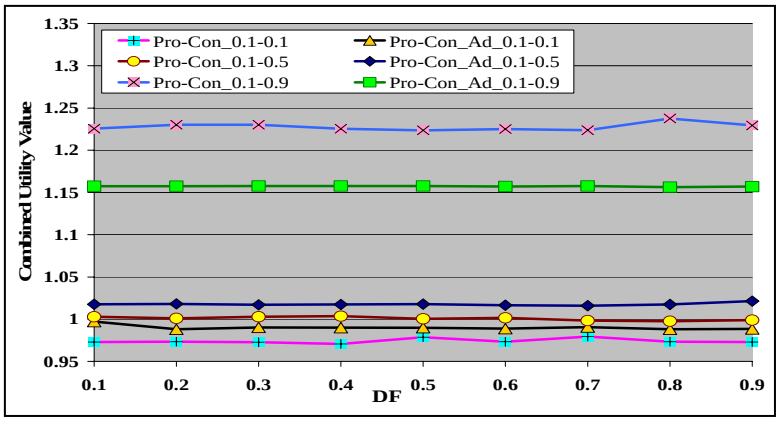

Figure 4.18 Combined Utility Value (CUV) for provider preference value of 0.1 of the price issue

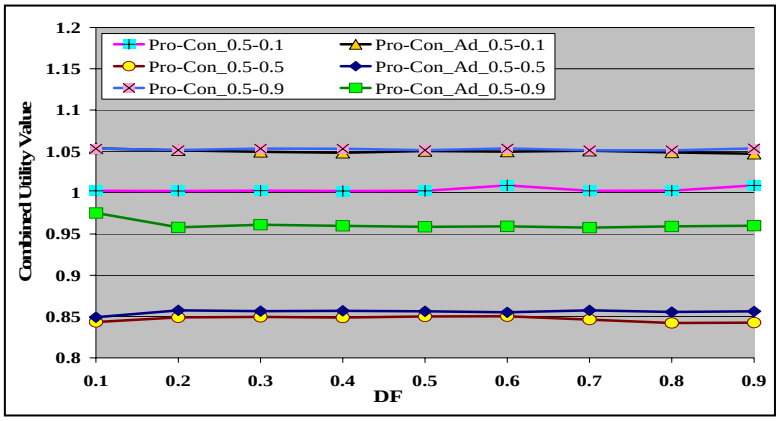

Figure 4.19 Combined Utility Value (CUV) for provider preference value of 0.5 of the price issue

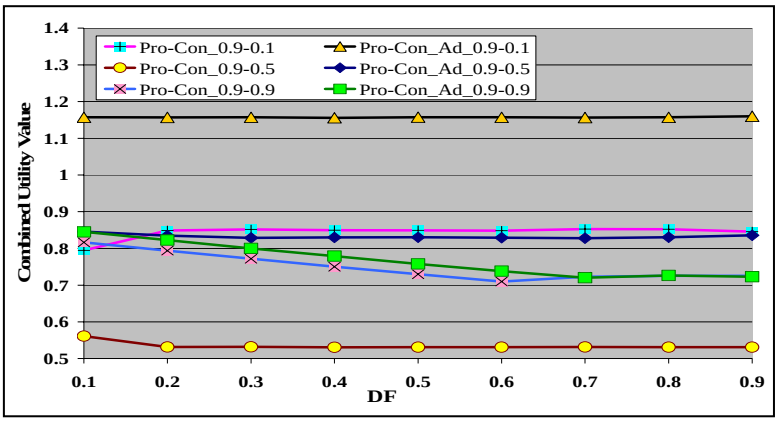

Figure 4.20 Combined Utility Value (CUV) for provider preference value of 0.9 of the price issue

In Figure 4.18, Figure 4.19 and Figure 4.20 we vary the preference and DF values for both parties where Pro-Con_0.5-0.1 implies provider and consumer priority values of 0.5 and 0.1 respectively and Pro-Con_Ad_0.1-0.5 denotes that adaptive algorithm is used in negotiation. All the Figures show that higher CUV is achieved for lower consumer preference values. There are several interesting observations that we can make from these experiments. Figure 4.20 shows that the CUV is very low about 0.53 on average for Pro-Con_0.9-0.5. In this case the provider agent is very conservative or boulware on the "price" issue, which has best-worst values of (20-10). So, the provider agent starts its offer with $20 and very slowly decreases the value in subsequent offers. The consumer agent, on the contrary, has moderate preference for "price" and starts its offer with $10 where best-worst values are (10-15). So, by the time the provider's offer enters the acceptable range of the consumer, the consumer reaches close to its worst values with very low utility value. Therefore, the CUV of the final offer is low. Since the overlapping regions for two of the issues start at worst values of the consumer, the CUV is generally low when the consumer is more conceding than the provider.

Although in Pro-Con_0.9-0.1, the consumer is more conceding for the "price" issue than Pro-Con_0.9-0.5, it has higher CUV than the other. The reason is that

the other two issues have higher preference values for Pro-Con_0.9-0.1, and therefore, the overall UV is higher than Pro-Con_0.9-0.5, where all the issues concede equally to their worst values.

We also observe that the adaptive algorithm performs better in almost all the cases in Figure 4.18, Figure 4.19 and Figure 4.20 with the exceptions where the consumer preference is 0.9, Pro-Con_0.1-0.9 and Pro-Con_0.5-0.9. When both agents have "price" preferences of 0.9, the other issues have very low preferences and are highly conceding. In this case the adaptive algorithm performs better by controlling the agents' conceding behavior at the optimal values of the issues. The adaptive algorithm improves the CUV for Pro-Con_0.9-0.1 in Figure 4.20 but degrades the CUV for Pro-Con_0.1-0.9 in Figure 4.18 although in both cases the combination of preferences is 0.1 and 0.9. Pro-Con_0.9-0.1 performs poorly (CUV is around 0.85) because of the conceding nature of the consumer agent towards its worst values in the acceptable range of the opponent as explained before. For Pro-Con_0.1-0.9, the provider agent concedes only half way towards its worst values to enter the acceptable range of its opponent, and therefore, has higher CUV than Pro-Con_0.9-0.1. With the adaptive algorithm, Pro-Con_Ad_0.1-0.9 and Pro-Con_Ad_0.9-0.1 have similar CUV (about 1.16), but since Pro-Con_0.9-0.1 performs poorly, adaptive algorithm makes a big improvement by restricting the highly conceding behavior of the consumer agent. The restriction enforced by the adaptive algorithm also stretches the time for negotiation as shown in Figure 4.21 and Figure 4.22.

When both parties have high preferences, at very low DF values the negotiation process often converges at the maximum time due to the extremely boulware nature of both parties and the behavior of the time-based function. Which party makes and accepts the final offer can affect the CUV at that time. For example, if the provider agent makes the final offer with its worst values, they will be the best values for the consumer agent, but all the worst values of the consumer are not the best values of the provider. Therefore, if the consumer

agent accepts the final offer, the CUV is likely to be higher than when the provider agent accepts the offer. Since we measure time as the number of offers exchanged between the agents with the provider agent sending the first offer at time 0, at the end of maximum time of 100, the last offer is made by the provider agent in the case of highly boulware behavior and the CUV is better given the acceptable range of values of the issues. But the results may be different for a different preference specification and maximum time.

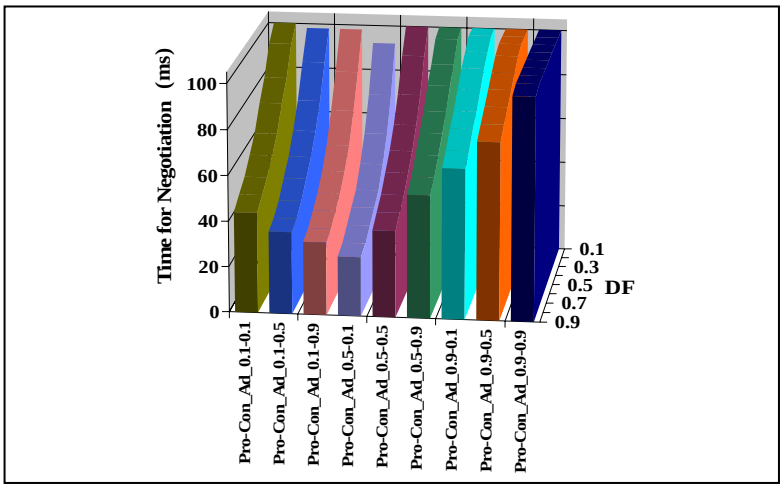

Figure 4.21 Time for negotiation for different DF and preference values with the adaptive algorithm

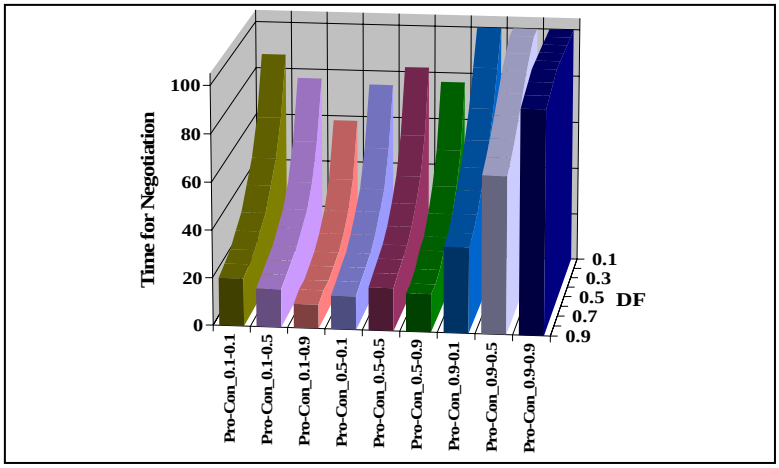

Figure 4.22 Time for negotiation for different DF and preference values without the adaptive algorithm

4.4.5. Third Stage

In the third stage we implemented two other time-based functions, the polynomial and the sigmoid functions, represented by Eq. 4.2 and Eq. 4.3 respectively, in the strategy repository of the NB. When multiple strategies are available, the most appropriate strategy is selected by the DSS to initialize the DM based on the policy specifications. Researchers who are working on behavioral strategies [18] [54] try to predict an opponent's behavior with different approaches. Our adaptive approach and the selection of different time-based function for different issues (mixed strategy) make prediction more challenging while at the same time, achieve higher CUV than the regular approach that uses a single time-based function for all the issues.

Objective: The first set of simulation experiments shows how we choose the values of the constants δ' and δ in the mapping Eq. 4.11 for the sigmoid function given in Eq. 4.3. Similar study is conducted for the polynomial function given in Eq. 4.2 to select the best values for the constants to compute its parameters using Eq. 4.9 and Eq. 4.10. Next we show the comparative performance of the three time-based functions, exponential, polynomial, and sigmoid, in terms of the CUV when used in bilateral bargaining of 3 issues. The experiments also demonstrate the feasibility of using all three different time-based functions for automated broker-based negotiation and the effectiveness of our corresponding mathematical policy mapping models as given in Eq. 4.8 to Eq. 4.11. We, thereby, prove our hypothesis that a mixed strategy, i.e., using different time-based function for different issues, can provide better CUV provided the strategy is chosen intelligently based on the preference of the issue and the DF.

The strategies with their corresponding policy mapping equations are stored in the strategy repository. The DSS selects the appropriate strategy based on consumer preferences and DF values given in the policy specifications. We provide an algorithm for strategy selection for intelligent agent-based

negotiation of 3 issues. We use the CUV to measure the goodness of the negotiation results. Finally, we show similar experimental results for 2, 4 and 10 issues to verify the scalability of our policy mapping equations.

Experimental Setup: We use the same set of 3 issues and their corresponding range of values for the consumer and the provider agents as in the previous experiments. We maintain DF=0.5 for the service provider and vary it from 0.1 to 0.9 for the service consumer. The preference value for "price" is also varied for the consumer from 0.1 to 0.9 with equal preferences for the remaining issues. For 2 issues, we consider "price" and "users" only. For 4 issues, we consider a fourth issue, "ServicePeriod", which has [best, worst] value pair of [30, 90] for the consumer and [90, 30] for the provider.

Observations: Figure 4.23, Figure 4.24, and Figure 4.25 show our test results for three different values of δ' for the negotiation of 3 issues. The values of the constants for which CUV_{avg} is maximized and CUV_{avg_stddev} is minimized for all preference and DF values are selected. CUV_{avg} is calculated using Eq. 4.18, which represents the average CUV for M preferences and N DF values. CUV_{avg_stddev} is calculated using Eq. 4.16 and Eq. 4.17, where σ_{pref} and CUV_{mean} represent the standard deviation and mean of the N different CUVs for N different DF values for a single preference value, and CUV_{avg_stddev} represents the average of all the σ_{pref} values for M different preferences.

$$CUV_{avg} = (1/MN)\, \Sigma_{pref}\, \Sigma_{DF}\, CUV \qquad \text{.....Eq. 4.18}$$

$$\sigma_{pref} = \sqrt{(1/N)\, \Sigma_{DF}\, (CUV_{DF} - CUV_{mean})^2} \qquad \text{..... Eq. 4.16}$$

$$CUV_{avg\_stddev} = (1/M)\, \Sigma_{pref}\, \sigma_{pref} \qquad \text{..... Eq. 4.17}$$

In Figure 4.23, Figure 4.24, and Figure 4.25, N=9 where DF varies from 0.1 to 0.9 for each preference value, and M=9 where preference varies from 0.1 to 0.9. The values of CUV_{avg} in the above three figures are 1.03, 1.06 and 0.98, respectively and the values of CUV_{avg_stddev} are 0.13, 0.11, and 0.16, respectively.

For Figure 4.24, CUV_{avg} is maximum and CUV_{avg_stddev} is minimum, where δ'=3 and δ=1.00. Therefore, these constant values are selected for the rest of the experiments where sigmoid time-based function is used for the negotiation of 3 issues.

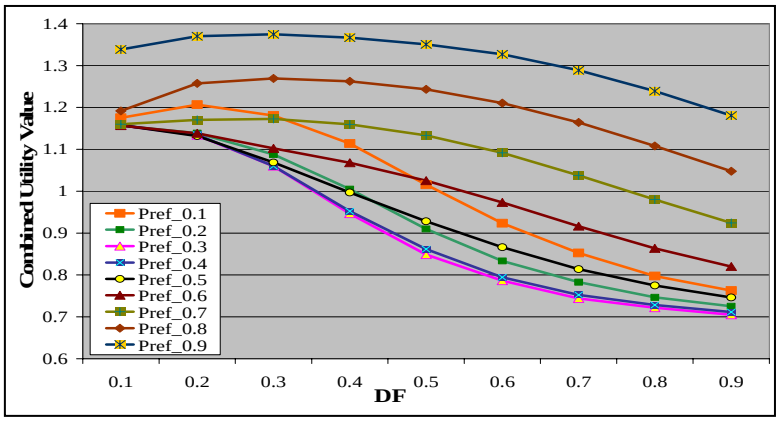

Figure 4.23 Sigmoid strategy δ´ = 2

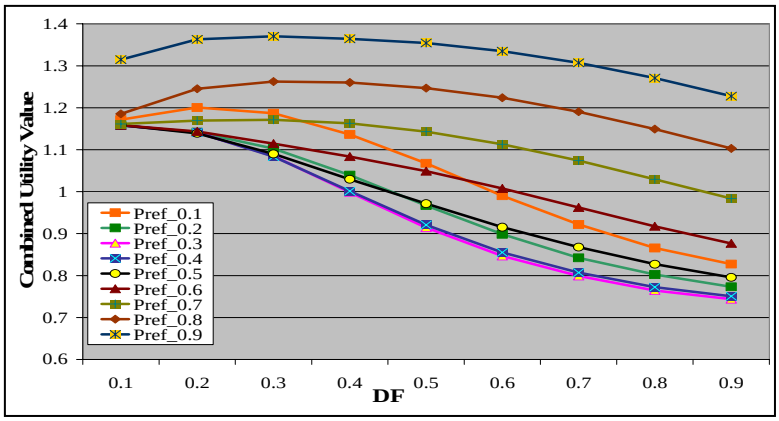

Figure 4.24 Sigmoid strategy δ´ = 3

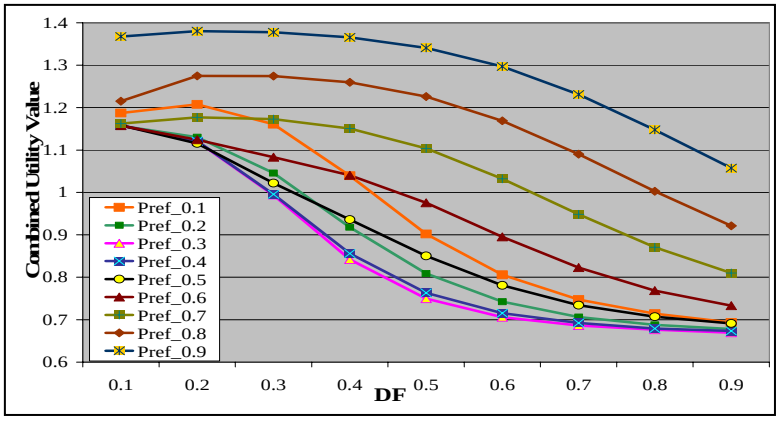

Figure 4.25 Sigmoid strategy δ´ = 4

We observe in Figure 4.24 that around the equal preference value, which is 0.33 for 3 issues, the average CUV is minimum (for pref=0.3 and 0.4), which was also observed for the exponential function in Figure 4.16. As shown in Figure 4.3, the sigmoid function is boulware in the beginning and in the end but conceding in the middle. Eq. 4.11 indicates that for the same preference value, as DF increases β also increases and the function becomes more conceding. For pref=0.1 to 0.6, the CUV is higher in the beginning and starts to drop rapidly after DF ≅ 0.5 because at that point the sigmoid function becomes conceding. For high preference values (pref=0.8 or 0.9) the sigmoid function is boulware and as a result, the CUV is high. So, sigmoid function performs generally well for very high preference values, and for low and medium preference values only up to DF=0.5 after which the CUV starts to drop.

We also implemented the polynomial time-based function given in Eq. 4.2 in the NB as another strategy for automated negotiation. Through similar experimental study as explained above for the sigmoid function, we determined the values of δ'=3 and δ=5 to use with Eq. 4.9 and λ=20 to use with Eq. 4.10, to compute β and k for polynomial function for the negotiation of 3 issues. The polynomial and exponential time functions use the same mapping equations.

We compare the performances of all three time-based functions in terms of the CUV in Figure 4.26, Figure 4.27, and Figure 4.28, where each function is used for all 3 issues in separate negotiations. We vary the preference values of the "price" issue and the DF for each function. Exp_0.1, Poly_0.1, and Sig_0.1 in the figure indicate the use of the exponential, polynomial and sigmoid function respectively for "price" pref=0.1. As shown in Figure 4.26, the polynomial function performs very close to the exponential function near the equal preference value. For preference values around and less than the equal preference values (pref=0.1 to 0.4), the polynomial function does better than the exponential function but the sigmoid function does the best until DF ≅ 0.5 and after that exponential function does the best of all three functions.

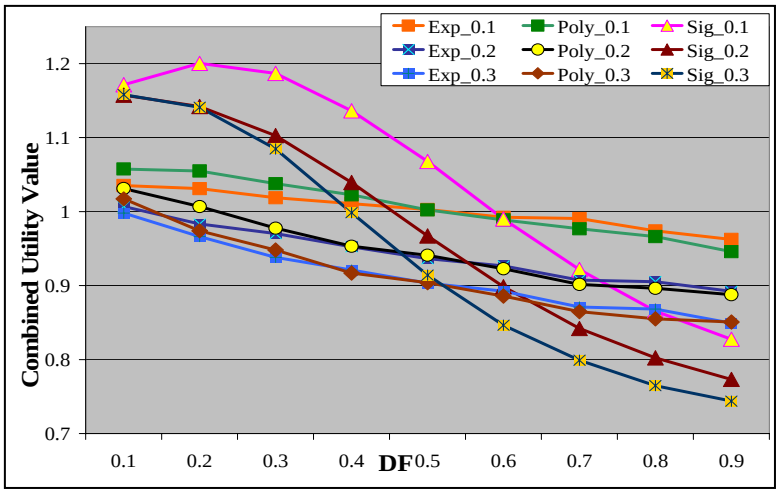

Figure 4.26 Exponential, Polynomial and Sigmoid strategies for 3 issues (pref = 0.1~0.3)

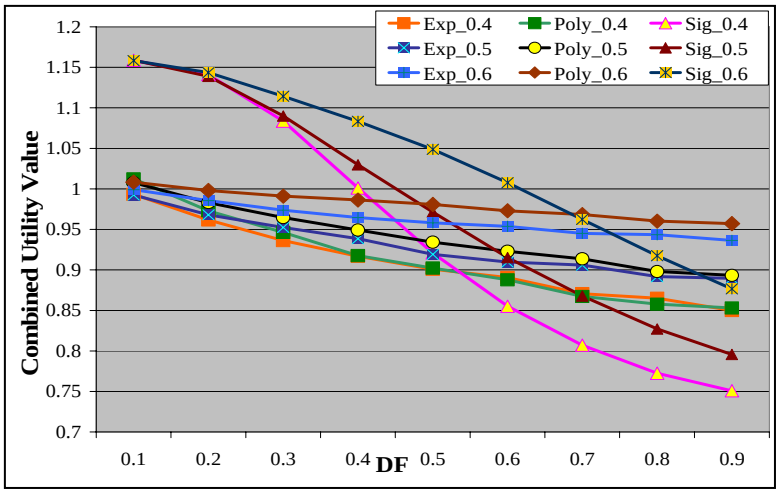

Figure 4.27 Exponential, Polynomial and Sigmoid strategies for 3 issues (pref = 0.4~0.6)

For preference values a little higher than the equal preference value (pref=0.5 to 0.7), polynomial equation does better than the exponential equation for all DF values but still the sigmoid function does the best until around DF ≅ 0.5, after which the polynomial function has the highest CUV of all three functions as shown in Figure 4.27.

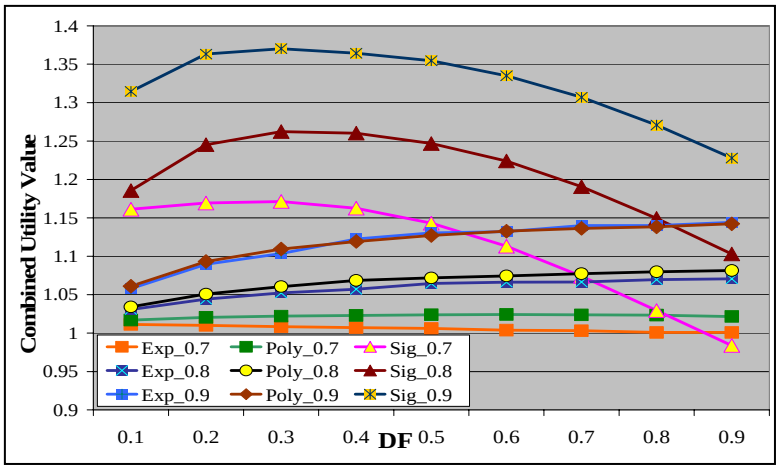

Figure 4.28 Exponential, Polynomial and Sigmoid strategies for 3 issues (pref = 0.7~0.9)

For very high preference values (pref=0.8 and 0.9) as shown in Figure 4.28, again the difference in the CUVs between the polynomial and the exponential function are insignificant, and the sigmoid function has the highest CUV than the other two functions for almost the whole range of DF.

The graphs described above also prove the feasibility of using the different time-based functions for automated negotiation and the validity of our policy mapping equations. The polynomial function behaves very similar to the exponential function, i.e., for medium and low preference values it has higher CUV for low DF values than for higher DF values. As DF increases, it becomes more conceding and the CUV decreases. For very high preference values, both the exponential and the polynomial functions are extremely boulware at low DF values, which results in non-optimal performance and lower CUV than for higher DF values. The reason behind the non-optimal performance is that the boulware function changes its offer values by very little amount, and in the end of the negotiation time, it concedes to its worst values resulting in low CUV. As DF increases, the function becomes more conceding and reaches an agreement with a better CUV. The behavior of the sigmoid function has already been explained earlier in this section. We see that all three functions demonstrate logical behavior pertaining to the specification of the preference and DF values

and the actual definition of the time-based equations. Therefore, our mapping equations provide effective models for mapping high level policies to low level parameters of the time-based functions.

Now we address our hypothesis about using a *mixed* strategy to provide better CUV where we select different time-based function for different issues based on the issue's preference value and the DF of the party. In all our previous experiments, we used the same function for all the issues of a party, which we will refer to as the *pure* strategy in this section. From our comparison of the performance of the different time-based functions, we observed that each function performs better than the others for certain preference and DF values.

```
1   Function SetStrategy (pref, DF, numIssues)  //reject or refuse
2      equalPref = 1/numIssues
3      prefRange1 = Math.ceil((equalPref * 0.25)*10)/10
4      prefRange2 = Math.ceil((equalPref  + 0.1)*10)/10
5      prefRange3 = Math.ceil((equalPref  * 2 + 0.1)*10)/10
6      If  (pref<prefRange1)
7         pt = Math.abs(pref - equalPref) * numIssues * 0.2 + 0.6
8         If  (DF <= pt)
9            strategy = SIGMOID
10        Else
11           strategy = POLYNOMIAL
12        End If
13     Else If  ((pref>=prefRange1) and (pref<prefRange2))
14        pt = Math.abs(pref - equalPref) * numIssues * 0.1 + 0.5
15        If  (DF <= pt)
16           strategy = SIGMOID
17        Else
18           strategy = EXPONENTIAL
19        End If
20     Else If  ((pref>=prefRange2) and (pref<prefRange3))
21        pt = Math.abs(pref - equalPref) * numIssues * 0.2 + 0.5
22        If  (DF <= pt)
23           strategy = SIGMOID
24        Else
25           strategy = POLYNOMIAL
26        End If
27     Else If  (pref>=prefRange3)
28        pt = Math.abs(pref - equalPref) * numIssues * 0.2 + 0.5
29        If  (DF <= pt)
30           strategy = SIGMOID
31        Else
32           strategy = POLYNOMIAL
33     End If
34     If (strategy == 0)
35           strategy = DEFAULT_STRATEGY
36     End If
37     return strategy
38  End Function
```

Figure 4.29 Strategy selection algorithm for three issues

Based on our observation of the negotiation for 3 issues, we divide the preference range 0~0.999 of an issue into four sub-ranges separated by 3 preference values (0.1, 0.4, and 0.8). The sub-ranges vary for different number of issues. We divided the preference range by observing and analyzing the points of intersection of the exponential, polynomial, and sigmoid graphs in Figure 4.26, Figure 4.27, and Figure 4.28. In each preference range, we further identify ranges of DF values for which a strategy performs the best, and accordingly, select that strategy for a given preference and DF value. We present an algorithm in Figure 4.29 for selecting the most efficient time-based function for the negotiation of 3 issues using the mixed strategy. The algorithm selects the time-based function based on a set of specified preference and DF values. In the algorithm the preference values are calculated based on the point of equal preference value (0.33), which we call `equalPref` in Figure 4.29. In our current work, the points of intersections are determined by manually analyzing the graphs of the different time-based function.

As an example of how we identify the different preference and DF ranges for the algorithm, we see that in Figure 4.26 and Figure 4.27, the sigmoid function does best for pref=0.1, 0.2, 0.3, and 0.4 until DF≅0.6, 0.55, 0.5, and 0.45 respectively. After that the exponential function does the best. So, the algorithm is designed to choose the sigmoid function for the above situation. Similarly, from Figure 4.28, the sigmoid function does best for all DF values for pref=0.8 and 0.9, and hence it is chosen by the algorithm for those preference values. Manual determination of the points of intersections of the different functions is a tedious job since the points of intersection are different for different preference values. We did this as an initial step to prove our hypothesis about the effectiveness of using a mixed strategy. In future we plan to implement a learning algorithm to automatically analyze the graphs to determine the intersecting points, which can be executed offline. The knowledge can be used

afterwards in the place of the different range and DF values in the algorithm of Figure 4.29.

The results of using the mixed strategy are demonstrated in Figure 4.30, Figure 4.31, Figure 4.32, which demonstrate that if the right strategy is selected intelligently for the current preference and DF values, a higher overall CUV can be achieved. In the figures, Mix_01 indicates that the mixed strategy selection algorithm is used for "price" pref=0.1 while DF is varied from 0.1 to 0.9.

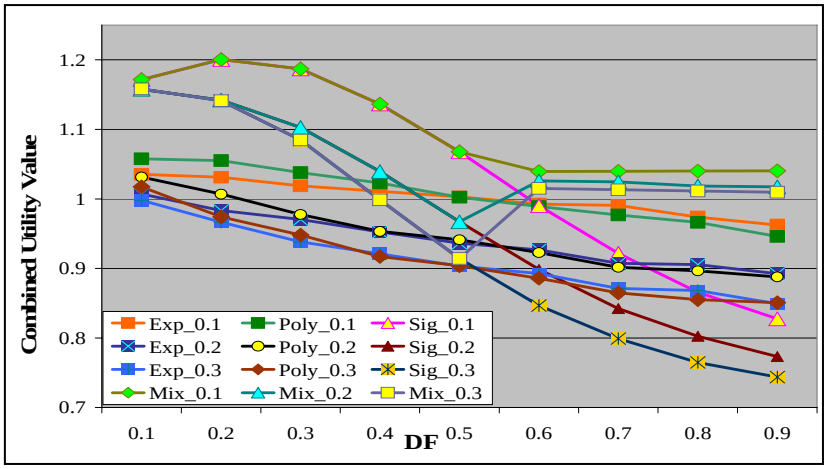

Figure 4.30 Mixed strategy for 3 issues (0.1~0.3)

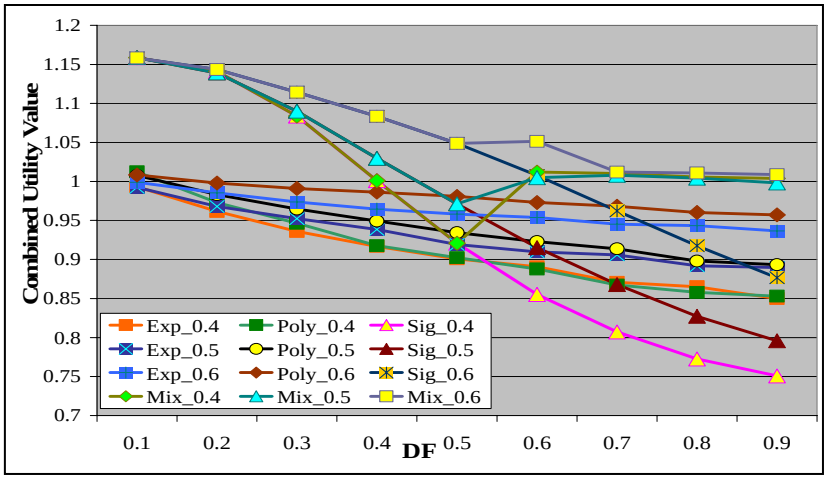

Figure 4.31 Mixed strategy for 3 issues (0.4~0.6)

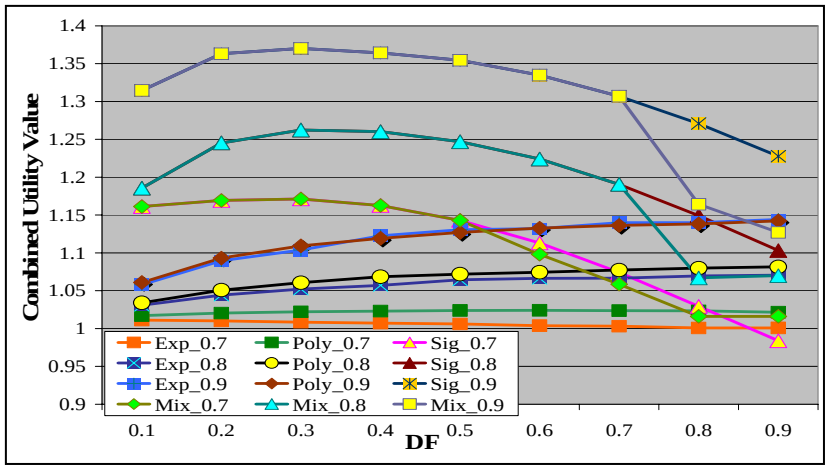

Figure 4.32 Mixed strategy for 3 issues (0.7~0.9)

We verified the hypothesis of the mixed strategy also for 2 issues as given in Figure 4.33, Figure 4.34, and Figure 4.35 where the values of δ' and δ are 5, 5 for the exponential, 5, 3 for the polynomial and 4, 1 for the sigmoid function respectively. For the exponential and polynomial functions, the value of k is set to 0.05. Again the points of intersection were carefully detected by manual analysis of the graphs and devising an algorithm to select the best strategy based on the preference and DF values.

To test the scalability of the mapping equations, we used the same equations with different constant values for 4 issues and 10 issues. We add 6 more issues with the 4 existing issues in the experiments to test with 10 issues. All the 6 new issues have same ranges of [best, worst] value pair of [80, 100] for the consumer and [100, 80] for the provider. The constants for all the experiments were determined by experimental study as explained earlier in this section. The results for 4 issues are shown in Figure 4.36, Figure 4.37, and Figure 4.38, and those for 10 issues are shown in Figure 4.39, Figure 4.40, and Figure 4.41, respectively. The values of δ' and δ are determined experimentally, which are [5, 5], [5, 3], and [2, 1] for the exponential, polynomial and sigmoid functions respectively for 4 issues, and [20, 20], [20, 20], and [2.5, 1] respectively for 10 issues. For the

exponential and polynomial functions, the value of k is set to 0.05 for both 4 and 10 issues.

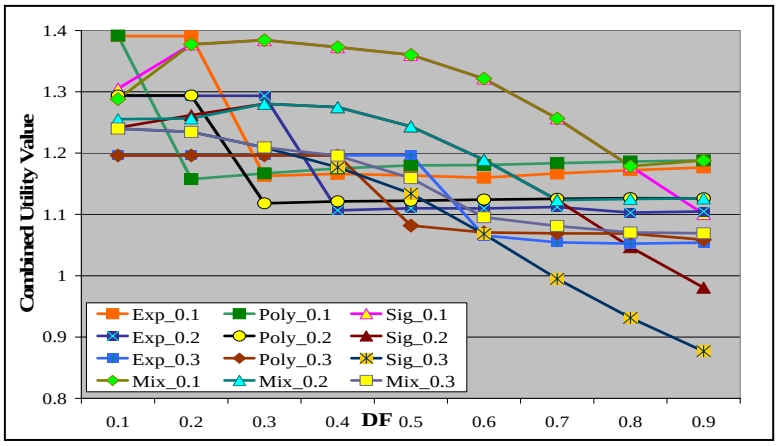

Figure 4.33 Mixed strategy for 2 issues (0.1~0.3)

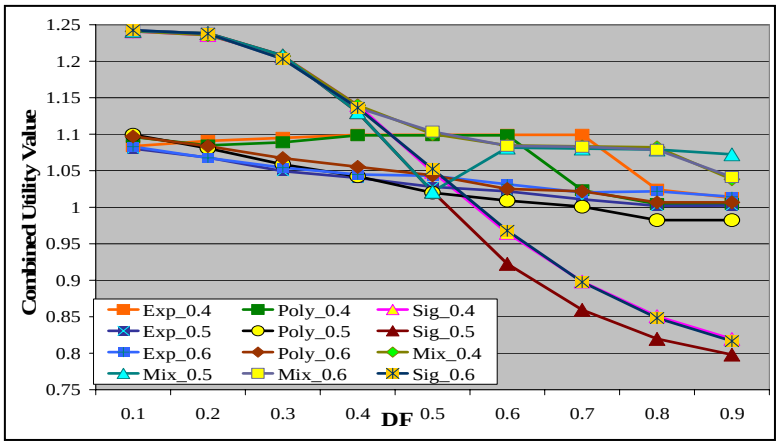

Figure 4.34 Mixed strategy for 2 issues (0.4~0.6)

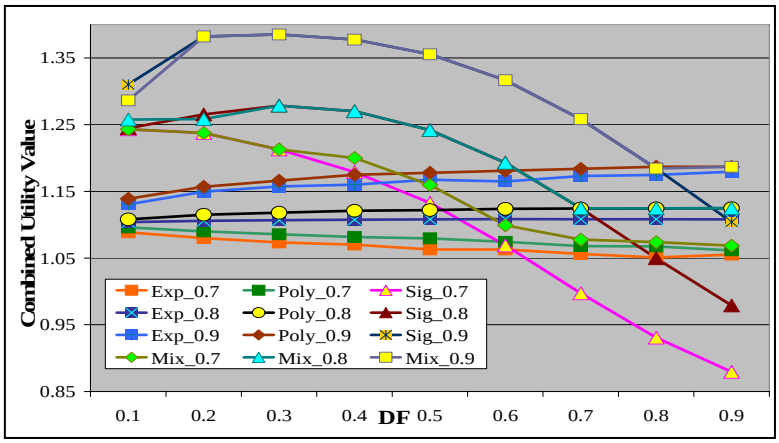

Figure 4.35 Mixed strategy for 2 issues (0.7~0.9)

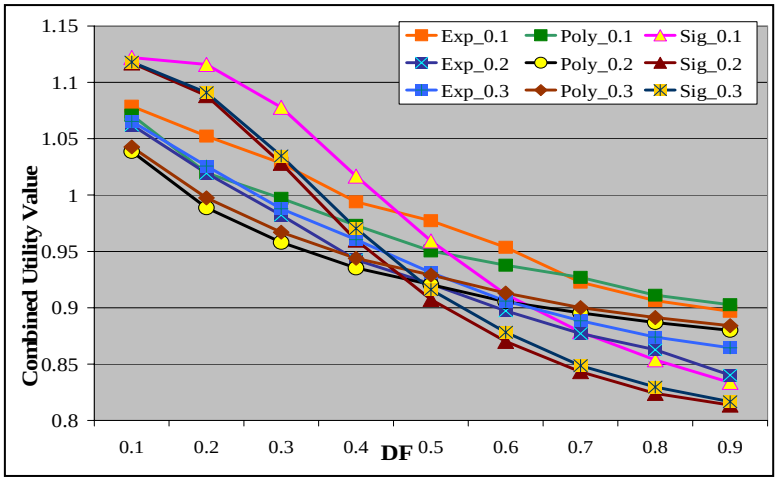

Figure 4.36 Exponential, Polynomial and Sigmoid strategies for 4 issues (0.1~0.3)

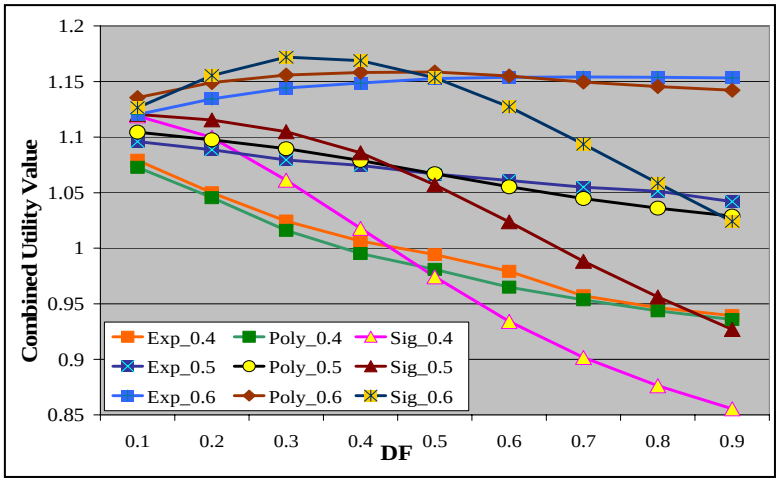

Figure 4.37 Exponential, Polynomial and Sigmoid strategies for 4 issues (0.4~0.6)

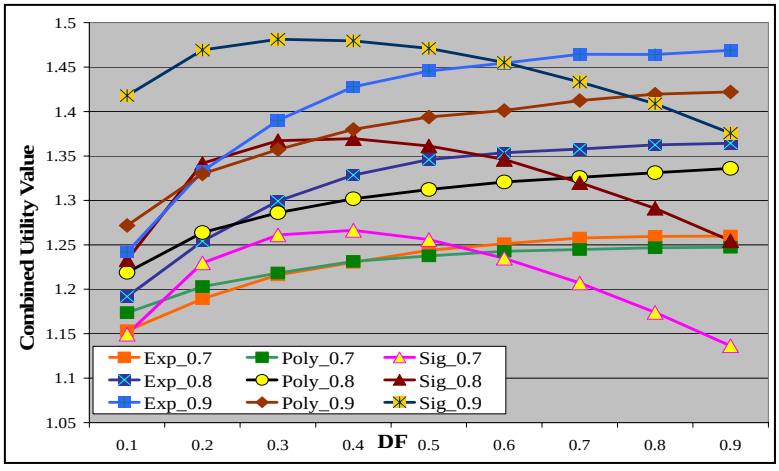

Figure 4.38 Exponential, Polynomial and Sigmoid strategies for 4 issues (0.7~0.9)

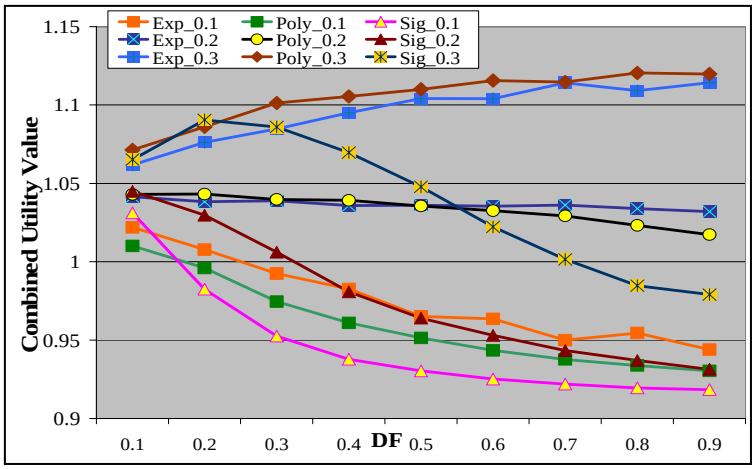

Figure 4.39 Exponential, Polynomial and Sigmoid strategies for 10 issues (0.1~0.3)

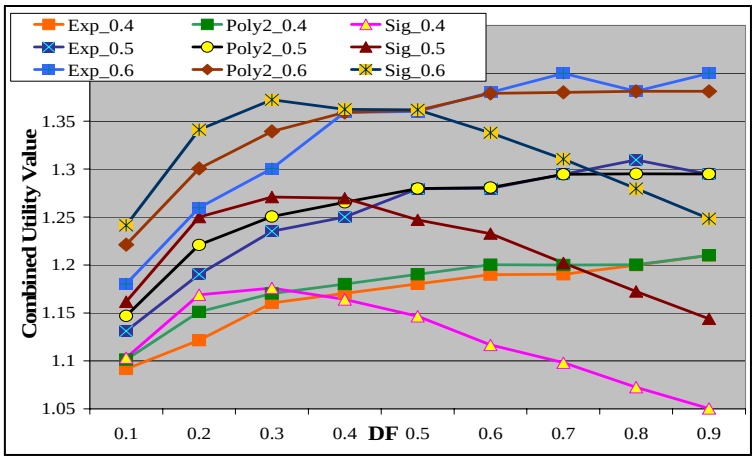

Figure 4.40 Exponential, Polynomial and Sigmoid strategies for 10 issues (0.1~0.3)

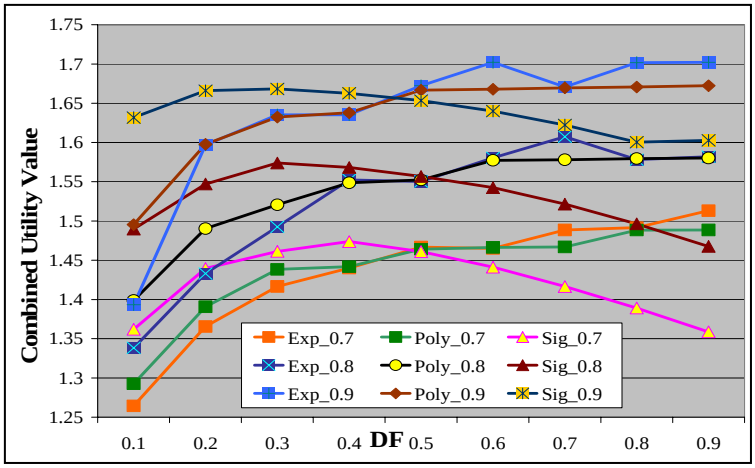

Figure 4.41 Exponential, Polynomial and Sigmoid strategies for 10 issues (0.1~0.3)

4.4.6. Discussion

The experimental results confirm the fact that the NB framework can be used effectively for automated negotiation of SLAs between two parties. The first stage of experiments with the exponential time-based function shows the feasibility of our approach and limitations caused by disregarding the opponent's offer when generating counter-offers.

We take into account the various observations made in stage 1 and the experiments of stage 2 show the improvements achieved by the revised mapping equations and the various decision algorithms. We note that our mapping equations successfully map the high level policy specifications to strategy models and decision algorithms, which is also evident from the results of the stage 3 experiments.

Our adaptive and decision algorithms take care of the problems of values of issues being better than the specified best value in the policy, and adapting the time-based functions during an ongoing negotiation. The adaptation of the functions is called for when values of issues in the offer exceeds a specified threshold value to receive user feedback and for resetting new worst values based on the opponent's offer values. The adaptive algorithm provides higher combined UV in most cases while ending the negotiation within the maximum negotiation time. Most importantly, it allows a user to exert a degree of control over the decision support system. We observe that better results may be obtained if the adaptive algorithm is used more selectively for lower preferences and DF values. Furthermore, the scaling factor δ' in Eq. 4.9 can be varied to control the time for negotiation while maintaining the CUV almost at the same level. It is also interesting to note that considering time as a count of offers exchanged introduces a discrepancy in the CUV when one party has high preference value of an issue compared to the other party.

In the stage 3 experiments we implement two other time-based functions. The results show the feasibility of our approach and the flexibility of the NB framework in supporting multiple negotiation strategies. We show that by careful analysis of the CUVs achieved by the individual functions, an algorithm can be developed for automatic strategy selection based on the policy specification. We observe that a mixed strategy provides the best CUV. It also makes prediction of the opponent's strategy more difficult.

4.5 Contribution

We propose an automated broker-based negotiation system for bilateral bargaining of SLAs between a service provider and a service consumer. Some of the state-of-the-art research on negotiations, such as Faratin *et al.* [41], Narayanan *et al.* [85] Brzostowski *et al.* [18], Hou *et al.* [54], and Lau *et al.* [69], addresses only the core decision support system. Others, such as Su *et al.* [109], Hung *et al.* [55] and Li *et al.* [71], address remote negotiation over the network which requires a more reliable negotiation protocol and message exchange format considering network dependency, vulnerability and security issues. Our approach conducts the negotiation locally within the broker framework. Therefore, it is faster, more reliable and free from network insecurities.

Hung *et al.* [55] and Chiu *et al.* [26] emphasize the representational aspect of negotiation information for expressing negotiation parameters and offers. We address the representational aspect differently using the WS-Policy standard. Li *et al.* [71], Gimpel *et al.* [48], and Commuzi *et al.* [30] use policies but in a different way than in our NB framework. Policies are typically used for expressing rules and low level negotiation parameters in the above approaches. We propose a policy model to express high level business goals, context, preferences, constraints, and metadata. We also define a policy mapping model to map the high level specifications to low level negotiation rules and strategies

automatically for the negotiation decision support system. Commuzi *et al.* [30] propose a negotiation framework that is similar to our NB, but it uses a policy definition that requires a low level strategy model and parameter specification and does not support adaptive negotiation. The NB provides a more flexible and extensible framework and supports additional information in the policy model, which is used to define the decision support system automatically from a high level goal and preference specification. Chhetri *et al.* [25] propose a hierarchical multi-agent framework for service selection and re-negotiation. CSMM is capable of supporting the same in a more independent way using the SRH and the NB.

Wilkes [123] argues about the necessity of a negotiation framework and proposes a utility function based flexible pricing scheme instead of a complex negotiation framework. The price model approach, however, does not support multi-issue negotiation and has a limitation on the number of SLOs it can map. The NB does not have that limitation and supports multi-issue negotiation.

Our NB framework provides a flexible and adaptive intelligent agent-based broker framework for multi-issue bilateral bargaining of SLAs. The decision functions are adapted during negotiation based on consumer feedback and opponent's offers. The high level policy model allows the framework to be used for different domains and the agent-based architecture supports possible future extensions to support multi-agent multi-party negotiations. Contributions in the area include the high level policy model, automated policy mapping model, the framework, and the various decision algorithms including the automatic strategy selection algorithm. The various algorithms can be further extended using machine learning theories to support more intelligent negotiation strategies. Through the technique of adaptation of the negotiation decision function, the research not only contributes to a higher combined utility value of the final offer for both negotiating parties, but also enables user feedback during the later stages in the negotiation process. The feature of automated negotiation combined with selective user control during negotiation, which is further improved by the

decision algorithms adds novelty to the research contributions with potential practical implications.

4.6 Summary

This chapter presents our NB framework and agent simulations executed on a partial prototype of the framework to validate our approach to intelligent agent-based negotiation in a trusted broker framework. We describe some background on common concepts about negotiation systems, negotiation theory, types and decision models. We then present our NB framework, policy model, policy mapping model, protocol, and the core decision support system. The DSS applies time-based negotiation functions with a cost-benefit model. We gradually introduce three different time-based functions namely, the exponential, polynomial and the sigmoid functions in the prototype implementation and validate our approach using a combined utility value of the final offer for both parties. We also validate our policy model, mapping model and algorithms by showing the improvements. The adaptive feature of the framework is demonstrated in the results of the experimental study. The intelligence of the negotiation agents is evoked by their corresponding decision models, which can also be extended by applying machine learning algorithms to the negotiation knowledge base to devise better negotiation parameters and algorithms. Finally, we present the contributions of this research in the paradigm of automated broker-based negotiation with respect to some of the recent related work.

Chapter 5

Performance Monitor Middleware

The World Wide Web is becoming the most efficient communication media, and to follow the current trend towards Service Oriented Architecture (SOA), business organizations are offering different services on the Internet that can be composed dynamically to create complex business applications. Quality of Service (QoS) is guaranteed by Service Level Agreements (SLAs) [31][75], which are negotiated between the service provider and the service consumer prior to service consumption. We presented our Comprehensive Service Management Middleware (CSMM) framework in Chapter 3 for complete client-side autonomic process management and our NB framework in Chapter 4 for automated broker-based negotiation of SLAs. In this chapter we present our Performance Monitor (PM) framework for SLA monitoring to ensure QoS of business processes.

SLAs are used in business processes to maintain service quality and protect the rights of the parties involved. To verify SLAs, efficient monitoring of the performance of the component Web services is essential both at the service provider and the service consumer. Distributed monitoring of business processes can be complex and costly. In Chapter 2 Section 2.3.3, we presented the state-of-

the-art research on Web service monitoring and discussed the various approaches summarized in Table 2.3. Most of the existing approaches require consumers to invoke services through a custom framework for monitoring purposes [82] [132]. Other industry software solutions address only intra-organizational monitoring [19] [57]. We propose a middleware solution, namely the Performance Monitor (PM) framework, to enable outsourcing of the task of SLA monitoring of both intra and inter-organizational composite Web service processes. In the rest of this chapter, we present our PM framework and experiments conducted to validate the prototype implementation of the framework. The experimental data from monitoring a composite Web service-based process establishes the effectiveness of the framework compared to the usual monitoring done by the workflow executor. We also discuss possible extensions to the framework for more generalized monitoring applications.

5.1 SLA Monitoring for Web Services

SLAs describe the legal binding between a service provider and a service consumer. A SLA typically includes parameters such as response time, availability, reliability, maximum number of requests at any time, service hours, average throughput, contract period, price, and penalties for not complying with the SLAs. For maintaining service quality in business processes, performance of the services ought to be measured both by the service provider and by the service consumer.

On the service provider's side the systems management software, such as our AWSE framework, monitors service performance and measures necessary parameters to ensure that the servers are maintaining expected service quality and throughput. Our goal is to facilitate client-side monitoring of Web services and thereby, provide a complete solution to client-side process management.

5.1.1. Problems in Client-side Monitoring

On the service consumer's side, monitoring is usually done at the point of service invocation or process execution, mainly, because the service is invoked over the network at another organization. The difficulties in client-side monitoring of Web services include the distributed nature of service-based processes, compositional complexity, dependency on the network, and inaccessibility to the service systems for monitoring purposes. In complex composite systems, a service may itself be a composite service. As such there can be processes containing sub-processes contributing further to the complexity in distributed monitoring.

Most of the current software products [19] [57] provide extensive server-side monitoring at the service provider's site but only work in organizational domains. Service performance statistics can be monitored at the workflow execution engine but these values include network delay. For SLA verification more accurate service performance statistics are necessary without significant impact of the network. To the best of our knowledge, a proper solution for multi-organizational distributed process monitoring currently does not exist.

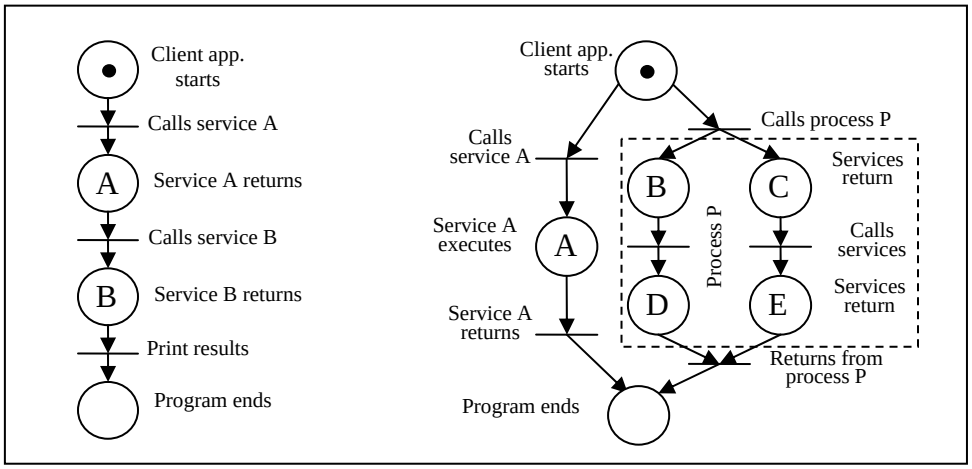

Figure 5.1 Petri net representation of (a) a simple workflow, (b) a complex workflow

5.1.2. Web Service-Based Workflows

A Web service-based workflow typically contains chains of Web services, which constitute a business process. Figure 5.1(a) shows a very simple workflow where the client application calls two different Web services consecutively. Figure 5.1(b) shows a more complex workflow having multiple chains of Web services that execute concurrently. Two of the three parallel branches constitute a sub-process (enclosed by the dashed line), which is composed of multiple Web services. Each sub-process and individual Web service component in a workflow has to satisfy a SLA, which is negotiated between the service provider and the service consumer before the process begins. When monitoring the SLAs, a sub-process is considered as a single Web service and the SLA is validated for the sub-process and not for its component services. For example, in Figure 5.1(b) the SLA is monitored for P, which is a composite service. We demonstrate the functionality of the PM using a simple workflow as shown in Figure 5.1(a) but the same principles apply to more complex workflows.

5.1.3. Monitoring Techniques

Communication with Web services is most commonly done using the standard SOAP (Simple Object Access Protocol) [119] messaging protocol. As a result, message interception is typically used to monitor Web services. A SOAP message is an XML document that contains an optional *Header* section for metadata and processing information, and a required *Body* section for the main message content. There are two common ways of monitoring Web services using message interception. One way is to build internal agents into the messaging framework at the servers that host Web services, which allow monitoring and reporting of the performance data. The agent should preferably be a standard part of the messaging framework and can provide monitoring data as an additional service. The other way is to build external intermediaries in between

the Web service environment and the consumer, such as with CA's Unicenter [111]. This approach allows easy maintenance at the cost of management overhead, an additional level of message redirection, possible bottlenecks and points of failure.

Other monitoring techniques include code level instrumentation with various monitoring and reporting functions or APIs (Application Programming Interfaces). Although this technique has the obvious advantage of reporting extensive and accurate monitoring data, the cost of maintaining the code can be considerable. Publishing management Web services for querying performance data or getting automated policy-based notification from the service providers can provide an efficient solution to the monitoring problem. The Site Manager in our Autonomic Web services Environment (AWSE) [112] [142] is an example of such a management Web service. However, it requires the service provider to implement custom management frameworks such as AWSE. In the PM, we use the internal agent-based technique because of its generality.

5.2 Our Approach: The PM framework

The PM can be used as an independent monitoring service, or as one of the main modules of the CSMM as shown in Figure 3.2. The PM takes a set of negotiated SLAs and a workflow description as input and monitors the performance of the component services to verify that the SLAs are satisfied. It can also be used independent of the CSMM to provide third party distributed workflow monitoring services. We show all the components in the CSMM that are directly connected to the PM in Figure 5.2 to indicate how the PM is used in the CSMM.

The PM comprises two types of disjoint sub-systems as shown in Figure 5.2 namely, a *Primary Sub-system (PS)* and multiple *Secondary Sub-systems (SS)*. The SSs monitor service performance at service providers' locations using one of the

monitoring techniques and send the reports to the PS. The PS accepts monitoring requests, receives monitoring reports, analyzes the reports to verify SLAs, and accordingly generates notifications for the respective service consumers. We design the SS using the internal agent-based message interception technique. We propose that the SS should be implemented as a standard integrated part of the message processing layer at the service provider's site, which can be optionally enabled to monitor selected services hosted by the server. This approach may require collaboration with the service provider, but reduces system maintenance and message redirection overhead and provides on site monitoring data that is independent of the network performance.

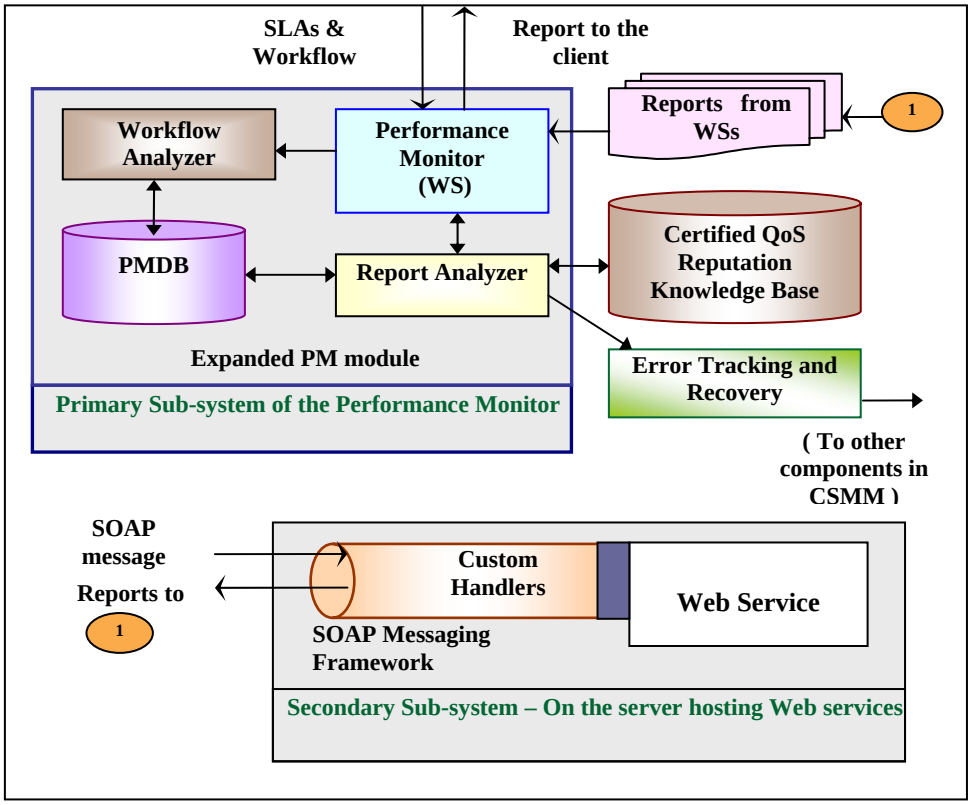

Figure 5.2 Architecture of the Performance Monitor

5.2.1. The Primary Sub-system (PS)

The Primary Sub-system (PS) collects and processes monitoring requests and reports and communicates with the other modules in the CSMM such as the

Error Tracking and Recovery module (ETR) and the Reputation Knowledge Base (RepKB). It consists of four main modules as described below.

Performance Monitor Web Service

The Performance Monitor Web Service (PMWS) receives requests for the monitoring service from consumers and reports from the SSs. The consumers provide workflow definitions and SLAs for all the component services as inputs when requesting the monitoring service. The PMWS forwards the information it receives to other modules for processing. After the monitor request is processed it sends a reply message to the service consumer, the SOAP header [119] of which contains the information necessary for monitoring by the SSs. This header information is included in every message that is used to invoke the component Web services in the workflow during its execution. If a violation of a SLA is detected in the analysis of the monitoring reports, the PMWS sends out notifications to the designated receivers such as the service consumer, workflow executor and the ETR.

Workflow Analyzer (WA)

The PMWS passes the workflow and the SLA information it receives from the consumer to the Workflow Analyzer (WA). The WA analyzes the workflow and the SLA specifications of each of the component Web services to determine their order of execution in the workflow and the QoS attributes that need to be monitored, and stores the information in a local database. It generates a Process ID (PID) for the workflow and a list of QoS attributes to be monitored for each component Web service and sends the information to the PMWS. The PMWS puts this information in the SOAP header of the message to send as a reply to the monitor request from the service consumer. The WA can be built to support any workflow specification language, such as the WS-BPEL [88], and SLA specification language, such as the WSLA (Web Service Level Agreement) [31].

Performance Monitor Database (PMDB)

A local Performance Monitor Database (PMDB) is used to store temporary workflow and SLA information, PIDs, and performance data from the reports collected from the SSs.

Report Analyzer (RA)

Performance reports sent to the PMWS are forwarded to the Report Analyzer (RA) module. The RA stores the reports according to their respective PIDs and Web service information in the PMDB. It then checks to verify if the SLAs are satisfied. If a violation of a SLA is detected, the RA prepares a report for the PMWS to send to the designated receivers as requested by the consumer and to the ETR.

5.2.2. The Secondary Sub-system (SS)

The Custom Performance Monitor Handler (CPMH) makes up the Secondary Sub-system (SS) and is installed as part of the SOAP message processing layer on the server that hosts the Web service. A SOAP message typically goes through several layers of processing after it reaches the destination Web server prior to reaching the appropriate Web service. One of these layers is the SOAP message processing layer, which can contain multiple handlers that intercept the messages, retrieve required information from the SOAP header, and perform necessary pre and post-processing.

The CPMH intercepts SOAP messages in both directions, i.e., to and from the Web service, in order to calculate the service response time. The CPMH checks for the PID and the URL to which to send the monitor report. In case of privacy and security concerns, additional privacy policies and encryption techniques can be used as proposed by Sahai *et al*. [104].

5.2.3. Associated CSMM Modules

The PM connects to an Error Tracking and Recovery (ETR) module, a certified Reputation Knowledgebase (RepKB) and the Workflow Manager (WM) in the CSMM. When a violation of a SLA is detected, the PM reports it to the ETR, which implements a policy-based decision making system to initiate proper action for recovery. In the recovery process, if a change in the workflow and SLA occurs, the PM is notified by the WM to make necessary changes in its records for monitoring. The RepKB is generated and updated from the monitor data received by the PM. Statistics are calculated from the monitor data based on which reputation scores are assigned to the different Web services monitored by the PM. The automation of reporting certifies the accuracy and dependability of the reputation information in the RepKB, which can be used for efficient service discovery. Implementation of these CSMM modules is subject to future research.

5.2.4. Computation of Service Statistics

We implement a Java class to generate the statistics of the monitored services based on the available data. The statistical data is stored in a table in the RepKB and used for generating reputation information. The table contains one record for every service that contains the average response time, average service availability, maximum response time, minimum response time, last access time, time of computation, service name, service URL and operation name. A service can provide multiple operations. For example, a book purchase service can provide book search, reservation and purchase operations. The average values are currently calculated from all existing records. It can be designed to calculate the average for the last N days, which can be used by the RepKB to compute a better reputation rating based on the age of the service statistics. The newer statistics are given more weight in calculating reputation scores. An example of

calculating the average response time based on the age of the data is given below:

$$avg\_resp\_time = \frac{1}{N}\sum_{N} resp\_time * \lambda^{\text{age of data in days}}$$

where $0 < \lambda < 1$ (we used $\lambda = 0.75$)

Finally, we calculate the total execution time for the workflow by summing up the response times of the component services:

$$p^{RT} = \sum_{i=1..N} s_i^{RT} \quad where \quad p = < s_1, ..., s_N >$$

Reliability of a service can be computed as the percentage of the number of times the service meets its SLAs with respect to the total number of times it is invoked. It is also possible to compute the performance of a service to a specific consumer for multiple processes, and thereby, verify that SLAs, such as 99% availability or average response time of 2~3 seconds with a specific service provider, are satisfied for all transactions of that service consumer.

5.3 Prototype Implementation

We illustrate the functionality and viability of the PM by using a prototype implemented in our lab to monitor a workflow similar to that of Figure 5.1(a). We use Java 2 platform SDK (Software Development Kit) 1.4.2 [62] with Eclipse 3.2.1 [40] on Windows XP to build the test platform, which includes the set up of the APACHE HTTP Web server [8], APACHE TOMCAT application server [9], AXIS2 1.1 [7] messaging framework, development of the example Web services used in the workflow, the multi-threaded load generator application and the complete PM framework.

APACHE AXIS2 [7] is an implementation of the SOAP 1.1 and 1.2 standard messaging protocols from W3C [119]. Web services used in the experiments are

created from scratch using the AXIs2 Object Model (AXIOM) and Application Programming Interface (API) [7]. Each service is programmed as a Java class with multiple methods, where each method performs different operation. The OMElement object of AXIOM is used as a parameter for the methods and the return values are also sent as an OMElement object.

For each Web service, a WSDL [120] file and a corresponding service descriptor file, services.xml, are created, which define the Java class to be used by the service and the appropriate message receivers. The WSDL and services.xml files for one of the example services, WSCompany, which is used as WS2 in the experiments are given in Appendix A. APACHE ANT 1.6.5 [6] is used to compile the code for each service and generate a compressed *.aar file with a specific directory structure. The *.aar is then placed in the AXIS2 services directory after shutting down the application server. When the application server is restarted, new services are installed automatically.

The handler code is developed using the same applications as mentioned above but coded differently following the AXIS2 1.1 handler specifications. ANT is used to compile the source files and generate a *.mar file, which is placed in the AXIS2 modules directory for installation. We developed our own example services to have more control in testing. The design and effectiveness of the PM framework is independent of the component services used in the workflow.

Each component in the PM is represented by a Java class. The PMWS is developed in a similar way as the example services. We observed that database initialization was affecting the response time of the services. Therefore, we implemented an additional *ConnectionPool* class to create and initialize a number of database connections in advance. Two other classes were developed for additional processing: one for generating reputation scores offline at a regular interval for the monitored services, and the other as a thread class for checking availability of the component Web services of the registered workflows. The availability information is updated in the local PMDB.

5.3.1. Evaluation Criteria

Most of the experiments evaluate the effectiveness of the PM in terms of the measured attribute, the service response time in milliseconds, against workload indicated by the number of clients invoking the service at the same time. We show the variation of the data measured by the PM from that monitored at the source code level of the Web service. The small difference in the measured data for the PM in comparison to the traditional monitoring done at the workflow execution code verifies the effectiveness of the PM.

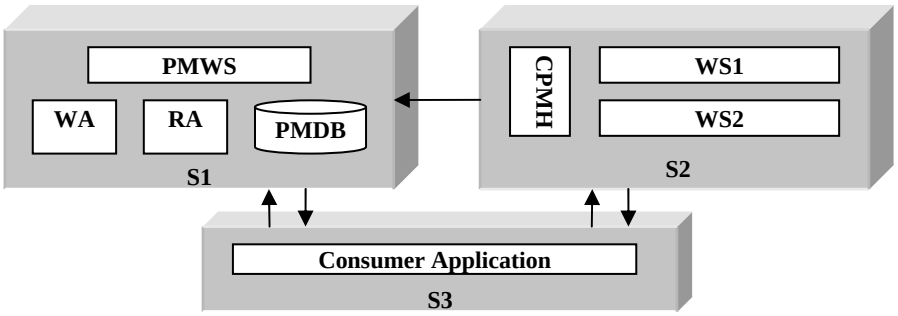

Figure 5.3 Layout of the servers for the prototype implementation

5.3.2. Experimental setup

The example workflow used in all the experiments is composed of two experimental Web services, WS1 (actual name WSOMAroma) and WS2 (actual name WSCompany), which are executed in sequence. A load generator application written in Java is used as the consumer. It first sends a request to the PM for monitor service and then executes the workflow. As shown in Figure 5.3, the primary sub-system (PS) (includes PMWS, WA, RA and PMDB) resides on server, S1, which runs the HTTP server and the application server to host the PMWS. WS1 and WS2 are hosted on a second server, S2, where the secondary sub-system (SS) or the CPMH is installed as a part of the message processing layer. The consumer application runs on a third machine, S3, which is similar to

the server machines but requires the Java Virtual Machine (JVM) and only the necessary libraries to run the application.

Our experimental Web service composition first calls the *executeQuery0* operation of the Web service *WSOMAroma* (WS1) and then the *companyQuery0* operation of the Web service *WSCompany* (WS2) in sequence. *executeQuery0* performs a sales related query on the *AromaDB* [56], which is a DB2 database containing 11 tables with about 70,000 records in one table including an XML data field. *companyQuery0* retrieves employee data from *CompanyDB,* another smaller DB2 database containing 6 tables.

```
1  <monitorRequest>
2   <Consumer_Info>
3    <Consumer_Name>…</Consumer_Name>
4    <Consumer_URL>…</Consumer_URL>
5    <Manager_URL>…</Manager_URL/>
6   </Consumer_Info>
7   <Workflow_Info>
8    <Service>
9     <Service_Name>…</Service_Name>
10    <Service_URL>…</Service_URL>
11    <Operation_Name>…</Operation_Name>
12    <Execution_Level>…</Execution_Level>
13    <SLA>
14      <Response_Time>…</Response_Time>
15    </SLA>
16   </Service>
17   <Service>
18    ...
19   </Service>
20  </Workflow_Info>
21 </monitorRequest>
```

Figure 5.4 Parameters for Monitor Request

We use an Apache 2.2.3 HTTP server [8] with an Apache Tomcat 6.0 application server and AXIS2 as the SOAP messaging framework for the Web services. The PMDB is created using IBM DB2 version 9.1. We use IBM Intel Pentium 4 Desktops with 2.66GHz CPU and 512MB of RAM as the server and the client machines with the Microsoft Windows XP Professional Version 2002 Service Pack 2 operating system.

```
<soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <tns:reportLog xmlns:tns="http://CSMM.server/xsd">
    <tns:PID>1</tns:PID>
    <tns:Consumer_Name>Farhana Zulkernine</tns:Consumer_Name>
    <tns:Consumer_URL>http://cs.queensu.ca/home/farhana/index.htm</tns:Consumer_URL>
    <tns:Manager_URL>http://localhost:8080/axis2/services/PerformanceMonitor</tns:Manager_URL>
    <tns:Create_Time>2007-06-09 03:01:39.14</tns:Create_Time>
    <tns:Response_Time />
  </tns:reportLog>
</soapenv:Header>
```

Figure 5.5 SOAP header for Web service calls

In the example scenario we monitor the response time and availability of the services by using the two-way Message Exchange Pattern (MEP) [117] i.e., all the requests for services are matched with a reply. We use simple XML specifications for the workflow and SLAs as shown in Figure 5.4, which the service consumer sends to the PM to request monitoring services. In response the PM returns a reply message containing a SOAP header block enclosed by the tag <reportLog> as shown in Figure 5.5, which states the necessary information for the SS.

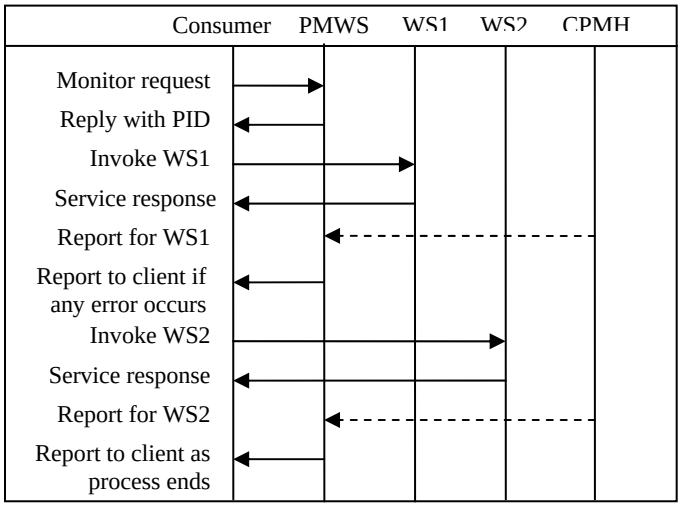

Figure 5.6 Message flow in monitoring using the PM

After receiving the reply the service consumer starts executing the workflow and embeds the header information in the SOAP header of the messages used to invoke the component services in the workflow. Figure 5.6 shows the message sequence chart for our example scenario. As services are invoked, the CPMH

sends the monitoring data of the specified QoS attributes to the URLs specified in the SOAP header. The dashed arrows indicate reports from the CPMH. In absence of the required information, no monitoring is done for the message. The CPMH correlates the requests and replies of a Web service by using the PID and other associated context information in the messages. Upon receipt of the reports, the PM validates the SLAs. If a violation of the SLA is detected it reports immediately, otherwise the PM sends a general report at the end of the process. We note that the ability to specify multiple receivers, for example Consumer_URL and Manager_URL in Figure 5.5 in line 5 and 6 makes the PM framework well-suited to distributed systems.

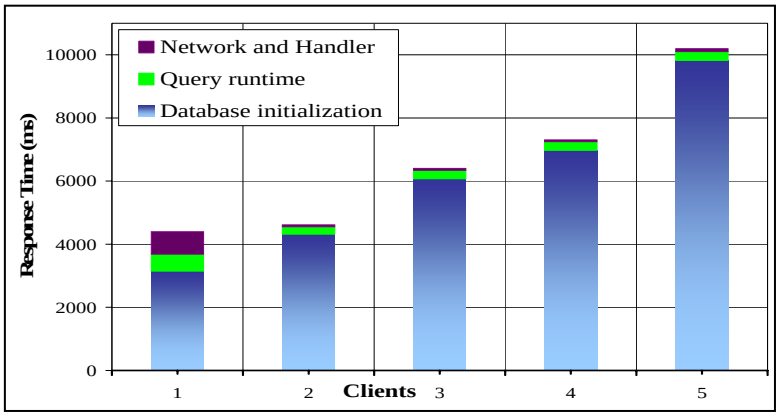

Figure 5.7 Network, database and system overhead

5.3.3. Observations

In all our experiments service performance is monitored at three different points: at the host Web server using basic code-level instrumentation; at the messaging framework using the CPMH and the PM, and at the consumer application. We analyze the performance overhead of our monitoring framework in terms of the service response time. We observed that a large time is required to connect to the DB compared to the actual query and network time, as shown in Figure 5.7, which has a significant impact on service performance as we increase the number of clients. To eliminate the connection time effect from the service

response time, we implemented connection pools for all the DBs used with the Web services. Connection pools create a specified number of connections beforehand to reduce the waiting time for the clients.

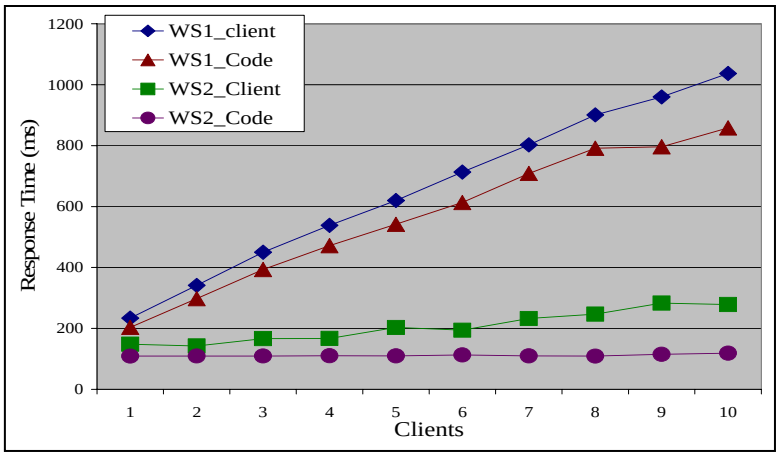

Figure 5.8 No monitoring

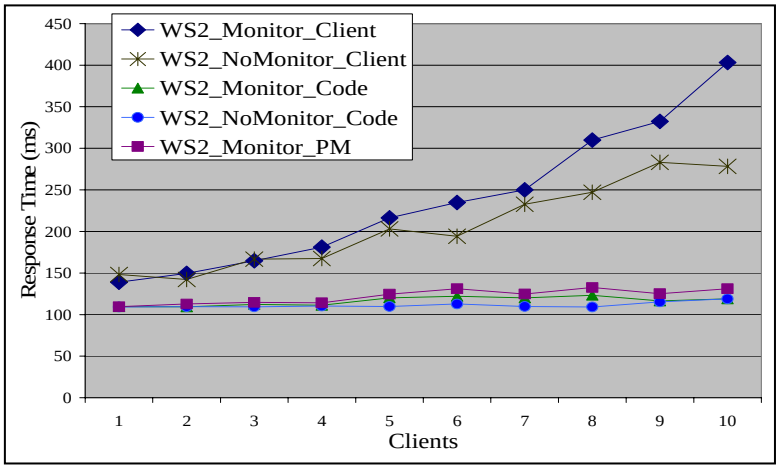

Figure 5.9 Monitoring overhead for WS2

First we show in Figure 5.8 the differences in response times measured at the code and at the client due to the network and associated software components, which are necessary to invoke a Web service. Figure 5.9 shows the performance of WS2 with and without monitoring at the three levels. Besides the network factor, one of the reasons for the overhead in response time measured at the client level is that all clients are executed on the same machine. For the same run,

the overhead measured at the code level is insignificant, which is more clearly shown in Figure 5.10. Also compared to the client-level monitoring data, the PM provides a much closer measurement to that of the code level, which justifies why the PM is a better tool for SLA verification.

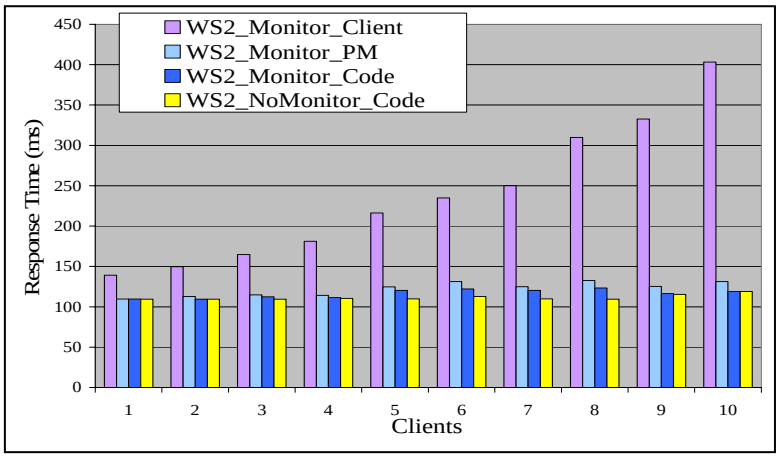

Figure 5.10 Monitoring overhead for WS2

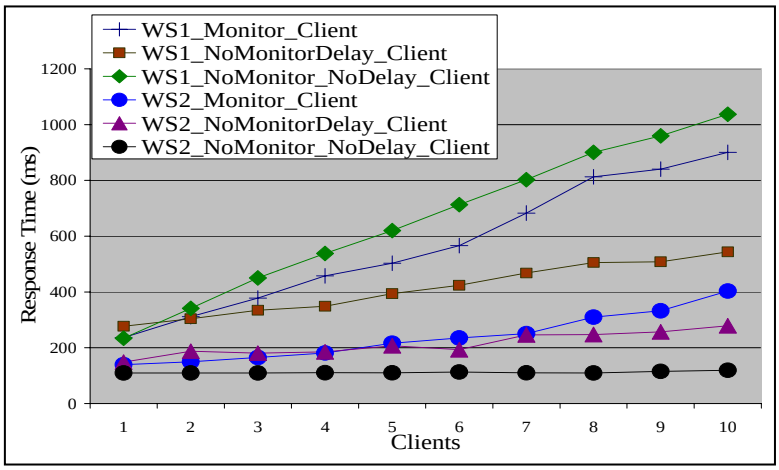

Figure 5.11 Effect of workload on WS1

Figure 5.11 shows that without monitoring, the response time of WS1 increases more linearly than with monitoring. The reason behind this is the absence of a workload adaptation technique on the server side causes performance degradation with sudden increase in workload for WS1. WS2 does not demonstrate such behavior because the prior call to WS1 inserts a queuing

delay for WS2. Due to the same reason, WS1 performs better with monitoring because then the first call is made to the PM. We verified this hypothesis by inserting a small delay of 1 second for every process. Figure 5.12 shows that when the PM is not used (WS1_NoMonitor) for monitoring, WS1 has lower average response time with delay inserted than without the delay. However, with the delay the performance of WS2 suffers.

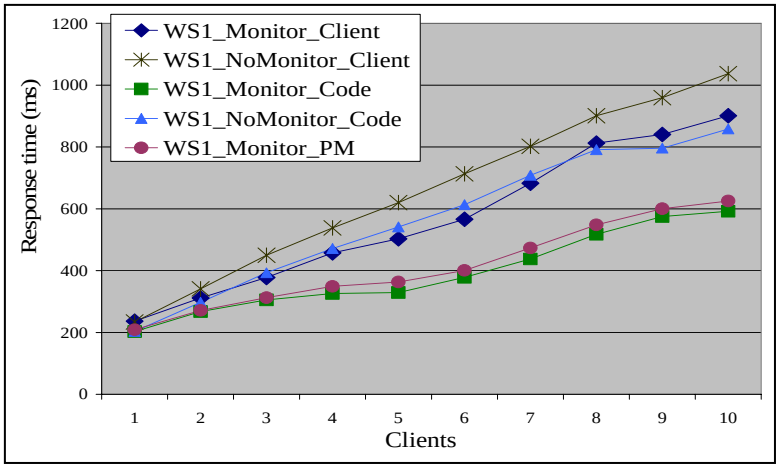

Figure 5.12 Effect of delay on WS1 and WS2

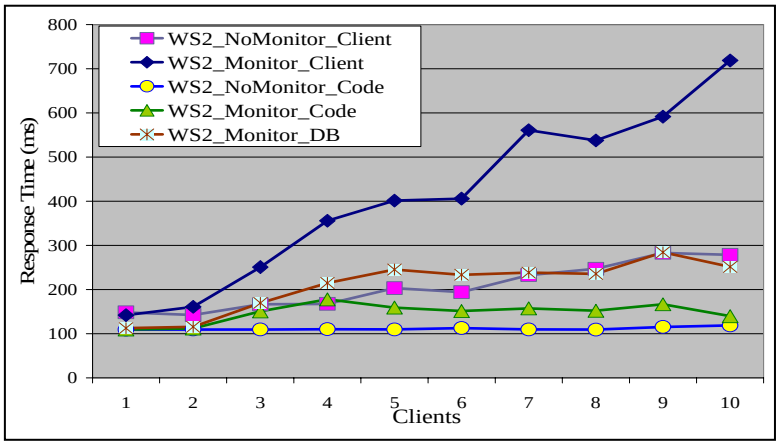

Figure 5.13 Monitoring overhead for WS2 with availability check

Checking availability: When the PM receives a *monitorRequest*, it invokes the component Web services to check their availability. The check allows the PM to report to the consumer about the status of the component services. If any of the

component services is unavailable, a corrective action may be taken before the process is executed. The added calls to the component Web services, however, incur considerable overhead as shown in Figure 5.13 for WS2. The extra calls degrade the performance of the component Web services for the added workload.

We also conducted experiments with multiple handlers, one for each monitored Web service hosted on a single server. Figure 5.14 shows the test results when two separate handlers are used for two Web services hosted on the same server. Our tests show that use of multiple handlers on one server increases the services' response times as the workload increases due to the computational and processing overhead. Therefore, assigning separate handlers for separate services that are hosted on the same server does not improve service performance when the PM is used.

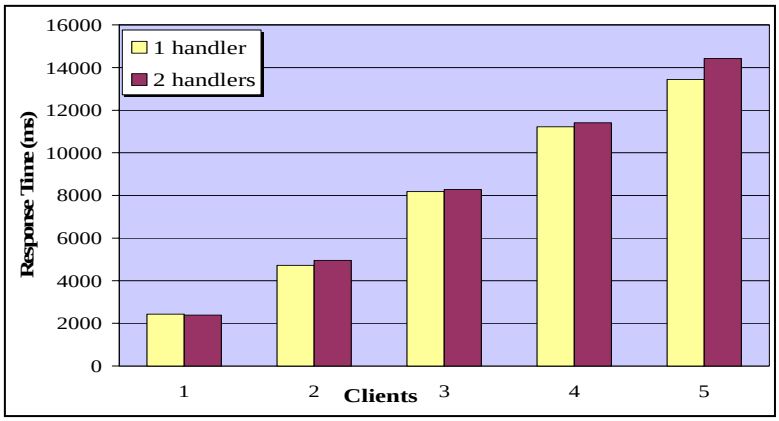

Figure 5.14 Effect of using multiple handlers

5.3.4. Discussion

We encountered several obstacles in implementing the prototype. We developed AXIS2 handlers as SSs for our prototype. In general, setup and use of the multi-server multi-component system for Web services is complex, challenging and time-consuming. Particularly, the use of the core Axiom APIs for developing the services and the CPMH required a considerable amount of time.

When the services are down, the handlers do not function, and therefore, it is difficult to monitor service availability using handlers. We implemented a timeout strategy to get around this problem where services are considered unavailable if a response is not received within a certain maximum time period. However, it does not reflect the correct status of the service because the problem can be in the network media. Ideally the handlers should function even if the services are down. Availability checks at the time of registration of the workflow with the PM were implemented to enable a quick remedy. However, the tests show that the overhead can be considerable, which is not cost-effective for small workflows consisting of very few services. The availability of the services should ideally be checked at regular intervals to maintain an up-to-date data of the service status.

The handlers intercept messages to and from the services to measure response time. For monitoring operations that follow the InOnly Message Exchange Pattern, i.e., Web service calls that do not generate any response, other reporting techniques will have to be used. Based on our observations, we can state that:

- for better and reliable monitoring, the handlers should be improved and functional at all times,
- service providers can provide additional functionality to make some of the service status information accessible through standard service management or enquiry interfaces, although the data may not be always be reliable considering that it is published by the service providers, and
- the PM can work best in a federated environment where a trust relationship can be established among the members of the federation and an agent application can be installed with the CPMH or in-code instrumentation to provide enhanced and more reliable monitoring data.

5.4 Contribution

We propose a trusted broker middleware framework for monitoring composite Web services-based processes. There are many monitoring software suites from large software vendors, such as CA and IBM, namely the CA Unicenter ® [111], IBM EWLM ® [23] and CA Wily SOA Manager ® [19], which are excellent for enterprise service and process management but do not support inter-organizational process monitoring because of inaccessibility to other systems.

Other monitoring tools are integrated with process execution tool as in Vaculín *et al.* [115] and Tröger *et al.* [114], and force the consumer to adopt the respective process management tool to avail monitoring services. We aimed at designing a standard-based middleware that can provide process monitoring services independent of the execution engine. The two part design of the PM enables the main sub-system to be used with any secondary sub-system, which is a powerful feature in the PM framework. The Web services-based design allows any existing management or monitoring tool to be used as secondary sub-system and to report monitoring data to the PM Web service. The design also allows the performance reports to be sent to multiple end points. In the proposed design, the PM is assumed to be a trusted broker middleware and the secondary sub-system is considered to be packaged as a standard message handler in the SOAP messaging layer for monitoring basic parameters.

The validity of our approach is illustrated by experiments conducted on a proof-of-concept prototype of the PM. The results of monitoring a composite process of two example Web services show the importance of taking the measurements at the service provider's location to avoid the overhead of network delay. Proper measurement is essential for SLA verification. We also propose the protocols for requesting monitoring services and reporting the

measurement data along with specification of the message formats based on common Web service standards.

The main limitations of the framework concern the installation of the secondary sub-system and the inability to monitor InOnly MEP. The first concern can be handled by establishing trust relationships between the PM and the monitored services, or by using the PM for monitoring services in a federated system. The second concern can be addressed by adopting a different type of secondary sub-system such as the popular in-code instrumentation, which enables better monitoring at the cost of maintenance of the code-level instrumentation.

5.5 Summary

Monitoring and verification of process SLAs is an essential part of composite process management in order to maintain the QoS of business processes. In this chapter, we propose a trusted monitoring middleware for monitoring and verification of SLAs for the component Web services of a multi-organizational composite process. We begin with a reference to the related work presented in Chapter 2, which motivated the design of the framework. The various monitoring techniques are described next with respect to their corresponding pros and cons. We then present our PM framework, its components and functionality. The PM is validated by a prototype implementation that is used to monitor a composite process made of two example Web services. The results of the experiments are provided and discussed with regards to network and monitoring overhead. Our study reveals the importance and effectiveness of distributed monitoring compared to monitoring at the point of process execution. The PM reduces monitoring overhead on the clients, which can be very effective for embedded or limited power mobile devices. It also enables computation of group statistics for multiple processes such as, 98% availability, 5

seconds average response time, or 99% reliability for all transactions of an organization. The limitations of the PM in monitoring one-way SOAP messages or providing more detailed monitoring data can be resolved by federated service monitoring or using a trust relationship with the service providers to install a more powerful secondary sub-system.

Chapter 6

Conclusion

Web services technology has led the way to a new era of Service Oriented Computing. The agility of building multi-organizational business processes with services has leveraged business collaboration. At the same time, complexity of the management of these business processes has increased greatly and offers many new challenges for the researchers and software vendors. We deem autonomic computing to be a feasible solution to managing the increasing complexity in software systems by making systems self-manageable. In this dissertation we present an approach to autonomic client-side management of Web services-based processes and present our frameworks for broker-based process monitoring and negotiation of Service Level Agreement (SLA). We conclude our dissertation in this chapter by providing a summary of the previous chapters and discussing some of the future work directions.

6.1 Summary

We address the problem of Web services-based composite process management in our research with a view to facilitating seamless execution while maintaining the QoS of the component services as described in the SLAs. A SLA is negotiated between the service provider and the service consumer prior to service invocation, which outlines the liabilities of both parties. SLAs are used to extract the Service Level Objectives (SLOs) for server-side management of Web services. Our research on client-side process management originated from our group research on the server-side Autonomic Web Services Environment (AWSE) [112] [142] framework.

We divide the complete job of building and executing a composite service into four main tasks: service selection, SLA negotiation, workflow composition and execution, and process monitoring. Based on these four main tasks and their complexity, we outline our research objectives and state our research hypothesis of the benefit of outsourcing the management tasks to trusted service providers. In the dissertation we present the Comprehensive Service Management Middleware (CSMM) for both partial and complete management of all four client-side management tasks. We also propose two of the four modules of the CSMM, the Negotiation Broker (NB) and the Performance Monitor (PM) that we illustrate within the scope of this research. We conclude Chapter 1 with the list of contributions of our research in the area of Web services-based composite process management, which include the conceptual CSMM framework for autonomic process management, the NB framework for automated intelligent agent-based adaptive SLA negotiation and the PM framework for inter-organizational SLA monitoring and verification.

We define some of the common concepts, and lay out the background for Web services research in Chapter 2. We describe the Web service life cycle, dynamic and static compositions, complex composite services and some of the

commonly used standards. Standards are very important to maintain interoperability for the benefit of composition of Web services. We then present the state-of-the-art research on Web services-based composite process management, SLA negotiation and process monitoring. We discuss the pros and cons of the different approaches to identify their limitations, and thereby, provide arguments in support of the objectives and motivations behind our research.

In SOC, outsourcing is an efficient means to reduce client-side overhead. We present our CSMM framework in Chapter 3, which is based on this concept and enables outsourcing of the different management tasks to the different modules within the framework. We discuss the complexity and challenges in each of the four tasks of process management and argue that each task presents an important research topic in the area. We provide an overview and design ideas for implementing each of the four main modules and the sub-modules in the CSMM. The functionality of the CSMM is explained using an example scenario. We describe our contribution with regards to the state-of-the-art research in the area of process management.

Chapters 4 and 5 present the NB and the PM frameworks respectively. We provide definitions of negotiation tactics, strategy, negotiation support systems (NSS), and decision support systems (DSS) in Chapter 4. We also provide some background information on the evolution of negotiation theory and decision support systems. We present the time-based exponential, polynomial and sigmoid decision functions and utility functions for cost-benefit decision strategy, which are applied in the NB framework. Our contributions in this area include the definition of a flexible, adaptive and intelligent NB framework; a WS-Policy based negotiation policy specification model and a policy mapping model for automatic translation of high level consumer policy specification to low level negotiation strategy and decision model. We propose an adaptive negotiation decision algorithm to adapt the decision function with the opponent's offer and

the current status of the negotiation process during an ongoing negotiation. The algorithm enables consumers to intercept the automated negotiation process in the NB to provide last minute feedback and then allow the negotiation to resume with an updated policy. We also propose a strategy selection algorithm that allows the NB to select the most appropriate negotiation strategy based on the parties' policy specifications. We demonstrate the improvements observed in the combined utility value (CUV) of the service provider and the service consumer as a result of the application of the adaptive algorithm and the automatic strategy selection algorithm.

Chapter 5 in the dissertation presents our PM framework. We emphasize the importance of monitoring with regards to ensuring QoS of business processes. We describe the problems in monitoring network-based distributed composite processes on the client-side, and the common approaches used for monitoring Web services. We apply the SOAP message interception technique to monitor response time, availability, and reliability of Web services. The PM framework consists of two sub-systems: the primary sub-system accepts monitor requests and performance reports to verify SLAs, while multiple secondary sub-systems reside on the servers that provide Web services as part of their message processing layer. We propose the secondary sub-system to be integrated with the standard message processing software, otherwise trust models could be used or federated service systems can be established to facilitate trusted monitoring by the PM broker service. Most of the existing management software provides monitoring within organizational boundaries. We propose a standard-based simple inter-organizational process monitoring framework that can provide trusted process monitoring services. The PM framework also allows the data collected to be used to build a reputation knowledge base for reputation-based service discovery.

In the following section we present our future work directions before concluding the dissertation.

6.2 Future Work

We describe our future work directions based on some of the limitations that we observed in our current approaches, possible enhancements, and potential new problem areas that we need to address to implement the CSMM.

6.2.1. The CSMM

Define and implement the SRH, WM and other components: We have not implemented two of the four main modules, the SRH and the WM, and some of the other components in the CSMM, such as the ETR and the Reputation KB. We plan to implement these components either based on new approaches or some existing approaches with necessary modifications. We provide a brief literature study of the service selection and workflow composition and execution approaches in Chapter 2. Significant research has already been done in these areas and we can customize an existing approach to implement the SRH and WM modules. The ETR needs to be designed to act on rules and policies for error recovery.

Extension of the CSMM and its modules to other application areas: Different modules of the CSMM can be used in the areas of wireless or pervasive computing and in small industrial devices, where clients have limited processing power. Due to the use of Web service technology, CSMM can also provide ubiquitous access to a wide range of service consumers. The independent service provisioning infrastructure allows each module in the CSMM to be used as an independent service provider in specific aspects such as, more general automated policy based negotiations, Web-based reputation systems, and distributed resource monitoring with WSDM (Web Services Distributed Management) interfaces. We plan to extend the application of the CSMM modules in other areas.

6.2.2. The NB

Automation of the detection of constant values in the parameter mapping equations: In our current research, we performed exhaustive experimental study to determine the optimum values of the constants used in the parameter mapping model. In our future work, we plan to implement an algorithm to automatically execute offline experiments to detect these values apriori for different number of issues.

Application of machine learning approaches: The policy mapping model, the decision model and the strategy selection algorithm can be further improved by the application of machine learning theory. The negotiation knowledge base contains negotiation history, which can be used with the policy database and machine learning theory to derive efficient goal and context mapping rules. Previous negotiation results can also be very effective for analyzing the performance of the time-based functions for various DF and preference values to improve the automatic strategy selection algorithm for different numbers of issues. The learning approach can be useful particularly when all the preferences and DF values are not explicitly specified in the policy, which puts forth a very interesting research problem to explore in future.

Trade-off algorithms for negotiation: In the NB framework, we can enhance the adaptive decision model to apply trade-off techniques. By observing the conceding pattern of the different issues in the opponent's offers, we can try to determine the other party's preference weights for the different issues. Hou *et al.* [54] apply regression analysis to determine an opponent's decision function and Brzostowski *et al.* [18] apply difference analysis to detect concession patterns. If we can determine an opponent's preferences, we can devise a trade-off algorithm to compromise with worse values of our less important issues that are more important to the opponent in order to achieve better values of own more important issues.

Multi-lateral negotiation: The NB framework currently supports only bilateral bargaining. If multiple service providers offer similar services, a multi-lateral negotiation can be very effective in finding the best service in the shortest time. We believe that the NB can be easily extended to support multi-lateral negotiation, and we would like to explore this in our future research.

Applicability to other application areas: Although the NB is designed for SLA negotiation, the flexible domain schema-based policy specification model allows different domain specific schemas to be used. We believe that the NB can be extended to be used in other application domains because of its powerful negotiation service provisioning infrastructure, Web service front-end, policy database and the negotiation knowledge base. One of these areas is cloud computing where software is made accessible to consumers as a service from the controlled environment of the different service providers. Another area is governance in SOA where service requirements propagate down from higher levels to the lower application levels, from where responses need to propagate to the upper levels. Industrial supply-chain management may be yet another prospective application area where product requirement specifications at the management level have to be negotiated with the manufacturing level.

Application of trust models: Negotiation service consumers have to entrust the NB with their company policies. Some trust model can, therefore, be implemented with the NB to establish a trust relationship between the NB and its service consumers. The current implementation of the NB assumes it to be a trusted broker framework.

6.2.3. The PM

Application of trust model: As for the NB, we assume that the PM is a trusted broker framework. The requirement to install the secondary sub-system (SS) on the service provider's side stands as a limitation of this approach. Although we propose making the SS a part of the message processing layer,

there can be arguments about its practicality. To overcome this limitation, trust models can be used to establish a trust relationship between the PM and the service providers for the installation of the SS.

Add support for monitoring other SLA parameters: We intend to explore other QoS attributes that can be monitored using the PM framework. With the current secondary sub-system based on the Custom Performance Monitor Handler (CPMH), the types of parameters that can be monitored are very limited. Applying other secondary sub-systems (SSs) is one of the ways to overcome this limitation.

Apply other monitoring techniques or secondary sub-systems: Another limitation of this approach is that the PM cannot monitor the InOnly Message Exchange Protocol (MEP), where the incoming requests do not generate any reply. In-code instrumentation can provide extensive monitoring data at the cost of maintenance and can be used as a remedy to this problem. However, the PM then needs to be implemented in a federated service environment.

Alternatively, other SS such as the server-side AWSE framework, or an existing Enterprise Management Software that the provider may have in place can be used with the same primary sub-system (PS) of the PM. In AWSE performance data about the service can be obtained by querying a publicly exposed management endpoint. However, the correctness of that data may be questionable. The grave problem of getting access to the service provider's site is the main reason behind the unavailability of good commercial software for monitoring cross-organizational Web processes. Other server-side monitoring tools and applications can be used as SSs to extend the applicability of the PM to more general Web processes.

Creation of reputation knowledgebase for service discovery: The monitoring data can be used to create an efficient reputation knowledge base since the performance data is collected automatically, and is therefore, correct

and unbiased. The reputation knowledge base can be used for service discovery in the presence of multiple service providers offering similar services.

References

[1] Abdelzaher, T., Stankovic, J., Lu, C., Zhang, R., and Lu, Y., 2003. Feedback Performance Control in Software Services, *IEEE Control Systems Magazine*, vol. 23(3).

[2] Aggarwal, R., Verma, K., Miller, J., and Milnor, W., 2004. Constraint Driven Web Service Composition in METEOR-S. In *Proc. of the IEEE Conf. on Services Computing (SCC'04)*, Washington, DC, IEEE CS Press, pp. 23-30.

[3] Akamai Technologies 2004. The Impact of Web Performance On E-Retail Success, *White Paper*, Available at: Http://www.Akamai.com/en/resources/pdf/whitepapers/Akamai_Eretail_Success_Whitepaper.Pdf.

[4] Anderson A., 2004. An Introduction to the Web Services Policy Language (WSPL). In *Proc. of IEEE Int. Workshop on Policies for Distributed Systems and Networks*, Yorktown Heights, NY, IEEE CS Press, pp. 189-190.

[5] Andonoff, E., Bouzguenda, L., and Hanachi, C., 2005. Specifying Workflow Web Services using Petri Nets with Objects and Generating of Their OWL-S Specifications. *LNCS*, Springer, vol. 3590, pp. 41-52.

[6] APACHE ANT. Retrieved Feb.5, 2009 from http://ant.apache.org/.

[7] APACHE AXIS2. Retrieved Feb.5, 2009 from http://ws.apache.org/axis2/.

[8] APACHE HTTP Server. Retrieved June 26, 2007 from: http://httpd.apache.org/.

[9] APACHE TOMCAT. Retrieved Feb. 5, 2009 from http://tomcat.apache.org/.

[10] Apshankar, K., Hanson, J., Mittal, K., Myerson, J., and Clark, M., 2002. Web Services Business Strategies and Architectures, Expert Press.

[11] Bansal, A., Kona, S., Simon, L., and Hite, T., 2005. A Universal Service-Semantics Description Language. In *Proc. of the 3rd IEEE European Conf. on Web Services (ECOWS'05)*, pp.214-225.

[12] Beam, C., and Segev, A., 1997. Automated Negotiations: A Survey of the State of the Art. *Wirtschaftsinformatik*, vol. 39(3), pp.263-268.

[13] Bennani, M., and Menascé, D., 2004. Assessing the Robustness of Self-Managing Computer Systems under Highly Variable Workloads. In *Proc. of the Int. Conf. on Autonomic Computing (ICAC'04)*, New York, NY, USA, pp.62-69.

[14] Birman, K., Renesse, R., and Vogels, W., 2004. Adding High Availability and Autonomic Behavior to Web Services, In *Proc. of the Int. Conf. on Software Engineering (ICSE'04)*, Scotland, UK, pp.17-26.

[15] Blum, A., 2004. UDDI as an Extended Web Services Registry: Versioning, quality of service, and more. White paper, SOA World magazine, vol. 4(6).

[16] Bouchenak, S, De Palma, N. Hagimont, D. Krakowiak, S. and Taton, C., 2006. Autonomic Management of Internet Services: Experience with Self-Optimization. In *Proc. of IEEE Int. Conf. on Autonomic Computing (ICAC)*, Dublin, Ireland, pp.309- 310.

[17] Brohman, M.K., Martin, P., Zulkernine, F., Piccoli, G., Parasuraman, A., and Watson, R., (submitted) 2008. A Design Theory Approach to Building Strategic Net-based Customer Service Systems, *Decision Sciences Journal*, Special Topic Forum: Advancing Decision Making in Service Innovation, *Blackwell Publishing*.

[18] Brzostowski, J. and Kowalczyk, R., 2006. Predicting partner's behaviour in agent negotiation. In *Proc. of Int. Joint Conf. on Autonomous Agents and Multiagent Systems, (AAMAS '06)*, Hakodate, Japan. ACM, NY, pp. 355-361.

[19] CA, 2007. CA Wily SOA Manager – Monitor and Manage Web Services Performance. *White paper*. At: http://ca.com/files/IndustryAnalystReports/technology_audit_ca_wily_application.pdf.

[20] Cappiello, C., Comuzzi, M., and Plebani, P., 2007. On Automated Generation of Web Service Level Agreements. In *Proc. of IEEE Int. Conf. on Advanced Information Systems Engineering (CAiSE'07)*, Trondheim, Norway, pp. 264-278.

[21] Cardoso, J., Miller, J., Sheth, A., and Arnold, J., 2004. Quality of Service for Workflows and Web Service Processes, *Journal of Web Semantics*, CiteSeer, vol. 1, pp. 281-308.

[22] Casati, F., Shan, E., Dayal, U., and Shan, M., 2003. Business-oriented management of Web services, *Communications of the ACM* Special Section on *Service-oriented computing*, vol. 46(10), pp.55-60.

[23] Chan, Y., Min, H., and Winkelbauer, L., 2006. Hardening the EWLM Performance Data, *IBM Red Paper*, at: http://www.redbooks.ibm.com/redpapers/pdfs/redp4018.pdf.

[24] Cheng, Y., Farha, R., Kim, M. S., Leon-Garcia, A., and Won-Ki Hong, J., 2006. A generic architecture for autonomic service and network management. *Computer Communications*, Elsevier, vol. 29(18), pp. 3691-3709.

[25] Chhetri, M., Lin, J., Goh, S., Zhang, J., Kowalczyk, R., and Yan, J., 2006. A Coordinated Architecture for the Agent-based Service Level Agreement Negotiation of Web Service Composition. In *Proc. of Australian Software Engineering Conf. (ASWEC'06)*, Sydney, Australia, IEEE CS Press, pp. 90-99.

[26] Chiu D., Cheung, S., Hung, P., and Leung, H., 2005. Facilitating e-Negotiation Processes with Semantic Web Technologies. In *Proc. of Hawaii Int. Conf. on System Sciences (HICSS'05)*, Big Island, Hawaii, pp. 36-45.

[27] Chung I., and Hollingsworth, J., 2004. Automated Cluster-Based Web Service Performance Tuning. In *Proc. of IEEE Conf. on High Performance Distributed Computing (HPDC'04)*, Honolulu, Hawaii, IEEE CS Press, pp. 36-44.

[28] Cibrán, M., Verheecke, B., Suvee, D., Vanderperren, W., and Jonckers V., 2004. Automatic Service Discovery and Integration using Semantic Descriptions in the Web Services Management Layer. In *Proc. of 3rd Nordic Conference on Web Services* and *Journal of Mathematical Modeling in Physics, Engineering and Cognitive Sciences*, vol. 11, pp.79-89.

[29] Coetzee, M., and Eloff, J., 2007. A trust and context aware access control model for web services conversations. In *Proc. of Int. Conf. on Trust, Privacy and Security in Digital Business (TrustBus'07)*, Regensburg, Germany, LNCS, Springer, pp. 115-124.

[30] Comuzzi, M., and Pernici, B., 2005. An Architecture for Flexible Web Service QoS Negotiation. In *Proc. of IEEE Int. EDOC Conf.*, Enschede, The Netherlands, pp. 70-82.

[31] Curbera, F., Khalaf, R., Mukhi, N., Tai, S., and Weerawarana, S., 2003. Service-oriented computing: The next step in Web services, *Communications of the ACM* Special Section on *Service-oriented computing*, ACM, New York, NY, USA, vol. 46(10), pp. 29-34.

[32] Dahlem, D. Nickel, L., Sacha, J., Biskupski, B., Dowling, J., and Meier, R., 2007. Towards Improving the Availability of Service Compositions. In *Proc. of the IEEE Int. Conf. on Digital Ecosystems and Technologies (DEST)*, Cairns, Australia, IEEE CS Press, pp. 67-70.

[33] Dai, Y., Hinchey, M., Qi, M., and Zou, X., 2006. Autonomic Security and Self-Protection based on Feature-Recognition with Virtual Neurons. In *Proc. of IEEE Int. Symposium o Dependable, Autonomic and Secure Computing (DASC)*, Washington, DC, USA, pp. 227-234.

[34] Dan, A., Davis, D., Kearney, R. Keller, A., King, R., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M., and Youssef, A., 2004. Web services on demand: WSLA-driven automated management, *IBM Systems Journal*, vol. 43(1), pp.136-158.

[35] Dearle, A., Kirby, G., and McCarthy, A., 2004. A Framework for Constraint-Based Deployment and Autonomic Management of Distributed Applications. In *Proc. of the Int. Conf. on Autonomic Computing (ICAC'04)*, New York, NY, pp. 300-301.

[36] Delaney, M., Foroughi, A., and Perkins, W., 1997. An empirical study of the efficacy of a computerized negotiation support system (NSS). *Decision Support System*, Elsevier, vol. 20(3), pp. 185-197.

[37] DTMF Distributed Management Task Force. List of standards (WBEM, WS Management). At: http://www.dmtf.org/standards.

[38] Dustdar, S., and Schreiner, W., 2005. A Survey on web services composition. *Int. Journal on Web and Grid Services*, InderScience, vol. 1(1), pp. 1-30.

[39] Dustdar, S., 2007. Towards Autonomic Processes and Services. In *Proc. of the Int. working conf. on Business Process and Services Computing (BPSC)*, Leipzig, Germany, pp. 13-19.

[40] Eclipse. Retrieved Feb. 5, 2009 from http://www.eclipse.org/.

[41] Faratin, P., Sierra, C., and Jennings, N., 1998. Negotiation Decision Functions for Autonomous Agents. *Int. Journal of Robotics and Autonomous Systems*, vol. 24(3-4), pp.159-182.

[42] Farrell, J., and Kreger, H., 2002. Web Services Management Approaches. *IBM Systems Journal*, vol. 41(2), pp.212-227.

[43] Fatima, S., Wooldridge, M., and Jennings, N., 2005. A Comparative Study of Game Theoretic and Evolutionary Models of Bargaining for Software Agents. *Artificial Intelligence Review*, vol. 23(2), pp.187-205.

[44] FIPA Contract Net Interaction Protocol Specification. At: http://www.fipa.org/specs/fipa00029/SC00029H.pdf.

[45] Foster, H., Uchitel, S., Magee, J., and Kramer, J., 2003. Model-based verification of Web service compositions. In *Proc. of 18th IEEE Int. Conf. on Automated Software Engineering*, Montreal, Canada, pp.152- 161.

[46] Ganek, A., and Corbi, T., 2003. The Dawning of the Autonomic Computing Era, *IBM Systems Journal*, vol. 42 (1), pp. 5-18.

[47] Garfinkel, S., and Spafford, G., 1997. *Web Security & Commerce*. O'Reilly, ISBN 10: 1-56592-269-7.

[48] Gimpel, H., Ludwig, H., Dan, A., and Kearney, B., 2003. PANDA: Specifying Policies for Automated Negotiations of Service Contracts. In *Proc. of Int. Conf. on Service Oriented Computing (ICSOC'03)*, Trento, Italy. Orlowska, M., Weerawarana, S., Papazoglou, M., and Yang, J. (Eds.), LNCS 2910, Springer, pp. 287–302.

[49] Gurguis, S., and Zeid, A., 2005. Towards autonomic web services: achieving self-healing using web services. *SIGSOFT Software Eng. Notes*, vol. 30(4), pp. 1-5.

[50] Gutiérrez, C., Fernández-Medina, E., and Piattini, M., 2004. A Survey of Web Services Security. In *Proc. of Int. Conf. on Computational Science and its Applications (ICCSA'04)*, Assisi, Italy, A. Laganà et al. (Eds.), Springer, LNCS 3043, pp. 968–977, 2004.

[51] Hanson, J., 2003. Coarse-Grained Interfaces Enable Service Composition in SOA, *White Paper, JavaOne*, Available at: Http://Builder.com/5100-6386-5064520.Html.

[52] Helft, M. and Peake, C., 2007. An Automated Approach to Legacy Modernization, *White Paper*, Computer Associates, Available at: Http://Viewer.Bitpipe.Com/Viewer/Viewdocument.Do?Accessid=5944652.

[53] Hogg, K. Chilcott, P., Nolan M., and Srinivasan, B., 2004. An evaluation of Web services in the design of a B2B application. In *Proc. of the conference on Australasian Computer Science*, Dunedin, New Zealand, Australian CS press, vol. 26, Estivill-Castro (Ed.) *ACM Int. Conf. Proc. Series*, vol. 56, pp.331-340.

[54] Hou, C., 2004. Predicting agents' tactics in automated negotiation. In *Proc. of IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology (IAT'04)*, Beijing, China, pp. 127-133.

[55] Hung, P., Li, H., and Jeng, J., 2004. WS-Negotiation: An Overview of Research Issues. In Proc. of Hawaii Int. Conf. on System Sciences (HICSS), Hawaii, IEEE CS Press, vol. 1, pp.10033b.

[56] IBM DB2 SQL and XQUERY Tutorial, Part 1. Retrieved from: http://www.ibm.com/developerworks/edu/dm-dw-dm-0607cao-i.html June 26, 2007.

[57] IBM Tivoli ® Composite Application Manager Suite. At: http://www-01.ibm.com/software/tivoli/solutions/application-management/.

[58] Issa, H., Assi, C., and Debbabi, M., 2006. QoS-Aware Middleware for Web Services Composition - A Qualitative Approach. In *Proc. of IEEE Symposium on Computers and Communications (ISCC'06)*, Cagliari, Sardinia, Italy, IEEE CS Press, pp.359-364.

[59] Iyengar, A., King, R., Ludwig, H., and Rouvellou, I., 2003. Performance and Service Level Considerations for Distributed Web Applications. In *Proc. of the 7th World Multi-conference on Systems, Cybernetics, and Informatics (SCI)*, Orlando, Florida, pp.1-6.

[60] Jakob, M., Healing, A., and Saffre F., 2007. Mercury: Multi-Agent Adaptive Service Selection Based on Non-Functional Attributes. In *Proc. of Int. Workshop on Engineering Emergence in Decentralized Autonomic Systems (EEDAS)*, Jacksonville, Florida, USA.

[61] Jammes, F. and Smit, H., 2005. Service-Oriented Architectures for Devices- the SIRENA View. In *Proc. of the 3rd IEEE Int. Conf. on Industrial Informatics (INDIN'05)*, Perth, Australia, pp.140-147.

[62] Java. Retrieved Feb. 5, 2009 from http://java.com/en/.

[63] Jelassi, T., and Foroughi, A., 1989. Negotiation Support Systems: an Overview of Design Issues and Existing Software. Decision Support Systems, vol. 5(2), 167-181.

[64] Johnson, M., 2005. Monitoring and Diagnosing Applications with ARM 4.0. MeasureIT, Issue 3.3, Computer Measurement Group (CMG).

[65] Kephart, J. O. and Chess, D. M., 2003. The Vision of Autonomic Computing, *Computer*, vol. 36 (1), pp. 41-50.

[66] Kersten, G., and Cray, D., 1997. Perspectives on Representation and Analysis of Negotiation: Towards Cognitive Support Systems. *Centre for Computer Assisted Management*, Carleton University, Ottawa, Canada.

[67] Koutsomitropoulos, D., Meidanis, D., Kandili, A., and Papatheodorou, T., 2006. Establishing the Semantic Web Reasoning Infrastructure on Description Logic Inference Engines. In *Proc. of Int. Conf. on Enterprise Information Systems (ICEIS)*, Paphos, Cyprus, pp.351-362.

[68] Kranenburg, H. Van, and Eertink, H., 2005. Processing Heterogeneous Context Information, In *Proc. of the IEEE Saint-Workshop*, Trento, Italy, IEEE CS Press, pp.140-143.

[69] Lau, R., Li, Y., Song, D., and Kwok, R., 2008. Knowledge discovery for adaptive negotiation agents in e-marketplaces. *Decision Support Systems*, Elsevier Science, Amsterdam, The Netherlands, vol. 45(2), pp. 310-323.

[70] Levy, R., Nagarajarao, J., Pacifici, G., Spreitzer, M., Tantawi, A., and Youssef, A., 2003. Performance Management for Cluster Based Web Services. In *Proc. of IEEE/IFIP Int. Symposium of Integrated Network Management*, Colorado Springs, USA, pp. 247-261.

[71] Li, H., Su, S., and Lam, H., 2006. On Automated e-Business Negotiations: Goal, Policy, Strategy, and Plans of Decision and Action. *Journal of Organizational Computing and Electronic Commerce*, Lawrence Erlbaum Associates, vol. 13(1), pp. 1-29.

[72] Liao, B., Gao, J., Hu, J., Chen, J., 2004. A Federated Multi-agent System: Autonomic Control of Web Services. In *Proc. of Int. Conf. on Machine Learning and Cybernetics (ICMLC'04)*, Shanghai, China, IEEE Press, vol. 1, pp. 1- 6.

[73] Lipton, P., 2004. Composition and Management of Web Services, *White Paper*, available at: http://www.sys-con.com/story/print.cfm?storyid=43567.

[74] Ludwig, S., Kersten, G., and Huang, X., 2006. Towards a Behavioral Agent-based Assistant for e-Negotiations. In *Proc. of Montreal Conf. on E-Technologies (MCETECH)*, InterNeg, Montreal, Canada, pp.1-14.

[75] Machiraju, V., Sahai, A., and van Moorsel, A., 2003. Web Services Management Network: An Overlay Network for Federated Service Management. *Technical Report*, HP lab, Palo Alto, CA, USA. In *IFIP/IEEE Int. Symposium on Integrated Network Management*, Colorado Springs, USA, pp. 351-364.

[76] Majithia, S., Ali, A., Rana, O., and Walker, D., 2004. Reputation-based Semantic Service Discovery. In *Proc. of the IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE'04), Modena, Italy, pp.297-302.

[77] Martin, D., 2006. Putting Web Services in Context, In *Elec. Notes in Theoretical Comp. Science*, vol. 146(1), pp. 3-16.

[78] Martin, P., Powley, W., and Benoit, D., 2004. Using Reflection to Introduce Self-Tuning Technology into DBMSs. In *Proc. of IDEAS'04*, Coimbra, Portugal, pp. 429-438.

[79] Maximilien, E. and Singh, M., 2005. Self-Adjusting Trust and Selection for Web Services. In *extended Proc. of IEEE Int. conf. on Autonomic Computing (ICAC'05)*, Seatle, Washington, USA, IEEE CS Press, pp.385-386.

[80] Maximilien, E., and Singh, M., 2004. A Framework and Ontology for Dynamic Web Services Selection, *IEEE Internet Computing*, vol. 8(5), pp.84-93.

[81] Mecella, M., Ouzzani, M., Paci, F., and Bertino, E., 2006. Access control enforcement for conversation-based web services. In *Proc. of the Int. Conf. on World Wide Web (WWW '06)*, Edinburgh, Scotland, ACM, New York, NY, pp. 257-266.

[82] Momm, C., Mayerl, C., Rathfelder, C., and Abeck, S., 2007. A Manageability Infrastructure for the Monitoring of Web Service Compositions. In *Proc. of the 14th Hewlett Packard – Software University Association (HP-SUA) Workshop*, München, Germany.

[83] Momotko, M., Gajewski, M., Ludwig, A., Kowalczyk, R., Kowalkiewicz, M., and Zhang, J., 2007. Towards adaptive management of QoS-aware service compositions. Multiagent Grid System, *Special Issue on "Advances in Grid services Engineering and Management"*, IOS Press, vol. 3(3), pp.299-312.

[84] Monge, H., and Martinez, M., 2005. AWS-Net Traveler: Autonomic Web Services Framework for Autonomic Business Processes. In *Proc. of the IEEE Int. Conf. on Services Computing (SCC'05)*, Orlando, FL, USA, vol. 2, pp. 270-272.

[85] Narayanan, V. and Jennings, N., 2006. Learning to negotiate optimally in non-stationary environments. In *Proc. of ACM Int. Workshop on Cooperative Information Agents (CIA)*, Edinburgh, UK and *LNCS*, Springer, vol. 4149, pp.288-300.

[86] Ngai, L., Mak, P., Ni, W., Liu, L., and Wu, C. 2007. A Semi-automated Negotiation Process to Improve the Usability for Online Marketplaces. In *Proc. of IEEE Int. Conf. on Computer and information Technology (CIT)*, Washington, DC, Oct 2007, IEEE CS Press, pp. 253-258.

[87] OASIS UDDI, 2005. Technical Committee Specification, v 3.0.2.

[88] OASIS WS-BPEL (Web Services Business Process Execution Language) 2.0 Draft, 2006. At: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.

[89] OASIS WSDM, 2005. Web Services Distributed Management: Management Using Web Services, (MUWS 1.0) part1, vol.1.1, OASIS Standard.

[90] OASIS WS-Reliability, 2004. Web Services Reliability, v1.1. At: http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf.

[91] OASIS WSRF, 2006. Web Services Resource Framework, v 1.2, OASIS Standard. At: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.

[92] OASIS WS-Security, 2006. Web Services Security, v 1.1. At: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[93] OASIS WS-Transaction, 2007. Web Services Transactions, v.1.1. At: http://www.oasis-open.org/news/oasis-news-2007-05-08.php.

[94] OASIS WS-Trust, 2007. Web Services Trust Framework, v 1.3. At: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html.

[95] Object Management Group (OMG) Unified Modeling Language (UML) v 2.1.2. Retrieved Feb. 6, 2009 from http://www.omg.org/technology/documents/formal/uml.htm.

[96] Olson, L., Winslett, M., Tonti, G., Seeley, N., Uszok, A., and Bradshaw, J., 2006. Trust Negotiation as an Authorization Service for Web Services. In *Proc. of IEEE ICDE Workshops*, Atlanta, GA, USA, IEEE CS Press, vol. 21, pp.1-10.

[97] Ouelhadj, D., Garibaldi, J., MacLaren, J., Sakellariou, R., and Krishnakumar, K., 2005. A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in Grid Computing. *Advances in Grid Computing-EGC*, Sloot *et al.*, (eds.), *LNCS*, Springer, vol. 3470, pp. 651-660.

[98] OWL-S 1.2 2006. (http://www.daml.org/services/owl-s/)

[99] Papazoglou, M., and van den Heuvel, W., 2005. Web Services Management: A Survey. *IEEE Internet Computing*, IEEE CS Press, pp.58-64.

[100] Park, S., Kim, W., and Kim, D., 2004. Autonomic Protection System Using Adaptive Security Policy. In *Proc. of Int. Conf. on Computational Science and Its Applications – ICCSA*, Assisi, Italy, A. Laganà et al. (Eds.), *LNCS*, vol. 3045, pp. 896–905.

[101] Pautasso, C., Heinis, T., Alonso, G., 2005. Autonomic Execution of Web Service Compositions. In *Proc. of Int. Conf. on Web Services (ICWS'05)*, Orlando, FL, USA, pp. 435-442.

[102] Raiffa H., 1982. The Art and Science of Negotiation. *Harvard University Press*, Cambridge, USA.

[103] Rao, J., and Su, X., 2005. A Survey of Automated Web Service Composition Methods. In *Proc. of Workshop on Semantic Web Services and Web Process Composition* (*SWSWPC'04) with ICWS'04*, San Diego, CA, USA. Cardoso, J., and Sheth, A. (Eds.): *LNCS 3387*, Springer, pp. 43–54.

[104] Sahai, A., Machiraju, V., Ouyang, J., and Wurster, K., 2002. Message Tracking in SOAP-based Web Services. In *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS'02)*, Florence, Italy, pp.33-47.

[105] Sahai, A., Machiraju, V., and Wurster, K., 2001. Monitoring and Controlling Internet based E-Services. In *Proc. of the IEEE Workshop on Internet Applications (WIAPP'01)*, San Jose, CA, pp.41-48.

[106] Seth, M., 2002. Web Services - A Fit for EAI, *White Paper*. Retrieved from http://www.developer.com/tech/article.php/10923_1489501_2.

[107] Silva, A., Neto, J., and Ibert, I., 2007. A computation environment for automated negotiation: a case study in electronic tourism. In *Proc. of ACM Symposium on Applied Computing (ACM SAC)*, Seoul, Korea, ACM, NY, USA, pp.654-658.

[108] Simon, L., Mallya, A., Bansal, A., Gupta, G., and Hite, T., 2005. A Universal Service Description Language. In *Proc. of the IEEE Int. Conf. on Web Services (ICWS'05)*, Orlando, Florida, USA, IEEE CS Press, pp.823-824.

[109] Su, S., Huang, C., Hammer, J., Huang, Y., Li, H., LiuWang, Liu, Y., Pluempitiwiriyawej, C., Lee, M., and Lam, H., 2001. An Internet-based Negotiation Server for E-Commerce, *VLDB Journal*, vol. 10(1), pp. 72-90.

[110] Ta, X., and Mao, G., 2006. Online End-to-End Quality of Service Monitoring for Service Level Agreement Verification. In *Proc. of the IEEE Int. Conf. on Networks (ICON)*, Singapore, vol. 2, pp: 1-6.

[111] Tcherevik, D., 2004. Managing Web Services with Unicenter® Web Services, *White Paper*, Office of the CTO, retrieved June 4, 2007, at: http://www.ca.com/files/WhitePapers/uni_wsdm_cto_wp.pdf.

[112] Tian, W., Zulkernine, F., Zebedee, J., Powley, W., and Martin, P., 2005. An Architecture for an Autonomic Web Services Environment. In *Proc. of the Joint Workshop on Web Services and Model-Driven Enterprise Information Systems (WSMDEIS) in conjunction with (ICEIS)*, Miami, Fl, pp. 54-66.

[113] Tosic, V., Pagurek, B., Patel, K., Esfandiari, B., and Ma, W., 2005. Management applications of the Web Service Offerings Language (WSOL). *Information Systems*, vol. 30 (7), pp. 564-586.

[114] Tröger, P., Meyer, H., Melzer, I., and Flehmig, M., 2007. Dynamic Provisioning and Monitoring of Stateful Services. In *Proc. of the 3rd Int. Conf. on Web Information Systems and Technology (WEBIST'07)*, INSTICC, Madeira, Portugal, pp. 434-438.

[115] Vaculín, R., and Sycara, K., 2008. Semantic Web Services Monitoring: An OWL-S based Approach. In *Proc. of Hawaii Int. Conf. on System Sciences (HICSS'08)*, Big Island, Hawaii, IEEE CS Press.

[116] Vidal, J.M., Buhler, P., and Stahl, C., 2004. Multi-agent systems with workflows. *IEEE Internet Computing*, 8(1):76–82, January/February 2004.

[117] W3C Message Exchange Patterns, 2002. Available at: http://www.w3.org/2002/ws/cg/2/07/meps.html.

[118] W3C PLING, 2007. Policy Language Interest Group. At: http://www.w3.org/Policy/pling/.

[119] W3C SOAP (Simple Object Access Protocol), 2004. Version 1.2 Part 1: Messaging Framework. Retrieved from http://www.w3.org/TR/soap12-part1/.

[120] W3C WSDL (Web Services Description Language), 2005.Version 2.0 (Working Draft). Retrieved from http://www.w3.org/2002/ws/desc/.

[121] W3C WS-Policy, 2006. Web Services Policy Framework. V 1.2. At: http://www.w3.org/Submission/WS-Policy/.

[122] W3C XML (eXtensible Markup Language), 2004. Retrieved from http://www.w3.org/XML/.

[123] Wilkes, J., 2008. Utility Functions, Prices and Negotiation, *Technical Report*, HP Labs Palo Alto, HPL-2008-81.

[124] Wong, S., 2001. Web Services: The Next Evolution of Application Integration, In Web Services and XML, *White Paper*, Patkai Networks, Available at: Http://www.Ebizq.Net/Topics/Web_Services/Features/1526.Html

[125] WS-Agreement, 2005. GGF (Global Grid Forum) developed by the Grid Resource Allocation and Agreement Protocol (GRAAP) work group (WG). At: http://www.ggf.org/Public_Comment_Docs/Documents/Oct-2005/WS-AgreementSpecificationDraft050920.pdf.

[126] Xu, Z., Martin, P., Powley, W., and Zulkernine, F., 2007. Reputation-Enhanced QoS-based Web Services Discovery. In *Proc. of the IEEE Int. Conf. on Web Services (ICWS'07)*, Salt Lake City, Utah, USA, IEEE CS Press, pp. 249-256.

[127] Yan, J., Zhang, J., Lin, J., Chhetri, M., Goh, S., and Kowalczyk, R., 2006. Towards Autonomous Service Level Agreement Negotiation for Adaptive Service Composition. In *Proc. of the 10th Int. Conf. on Computer Supported Cooperative Work in Design*, Nanjing, China, pp.1-6.

[128] Yan, J., Pidgeon, P., Krishna, A., and Yong, J., 2007. An agent-based decentralised process management framework for web service composition. In *Proc. of the Int. Conf. on Scalable information Systems*, Suzhou, China, *ACM Int. Conf. Proc. Series*, vol. 304, ICST (Institute for Computer Sciences Social-Informatics and Telecommunications Engineering), Brussels, Belgium, pp.1-4.

[129] Yang, Y., Tan, Q., Xiao, Y., Yu, J., and Liu, F., 2006. Exploiting Hierarchical CP-Nets to Increase the Reliability of Web Services Workflow. In *Proc. of the Int. Symposium on Applications on internet (SAINT)*, Phoenix, Arizona, USA, IEEE CS Press, pp. 116–122.

[130] Yee, G., and Korba, L., 2003. Bilateral E-Services Negotiation under Uncertainty. In *Proc. of the Int. Symposium on Applications and the Internet (SAINT '03)*, Orlando, Florida, pp.352.

[131] Yu, T., Zhang, Y., and Lin, K., 2007. Efficient algorithms for Web services selection with end-to-end QoS constraints. *ACM Transactions on the Web*, vol. 1(1), Art. 6, pp.1-26.

[132] Zeid, A., and Gurguis, S., 2005. Towards autonomic Web services. In *Proc. of the ACS/IEEE Int. Conf. on Computer Systems and Application (ICCSA)*, Kairo, Egypt, pp. 69-73.

[133] Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., and Chang, H., 2004. QoS-Aware Middleware for Web Services Composition. In *IEEE Transactions on Software Engineering*, vol. 30(5), IEEE CS Press.

[134] Zhang, F., Ji, G., Guo, H., Zhu, P., and Liao, B., 2006. Autonomic Management of Web Services Based on Federated Multi-agent System. In *Proc. of the World Congress on Intelligent Control and Automation*, Dalian, China, vol. 2, pp. 6949-6953.

[135] Zhou, J., Koivisto, J., and Niemela, E., 2006. A Survey on Semantic Web Services and a Case Study. In *Proc. of the Int. Conf. on Computer Supported Cooperative Work in Design (CSCWD''06)*, Nanjing, China, IEEE CS Press, pp.1-7.

[136] Zhu, Y., Ma, D., Sun, H., Zhang, S., and Li, J. 2007. A QoS-aware middleware for ensuring web services reliability. In *Proc. of the IASTED Int. Multi-Conf. on Parallel and Distributed Computing and Networks,* Innsbruck, Austria, ACTA Press, Anaheim, CA, pp. 338-344.

[137] Zulkernine, F., and Martin, P., 2006. Web Services Management: Towards Efficient Web Data Access, pp. 266-288, Vakali, A., and Pallis, G. (Eds.), *Web Data Management Practices: Emerging Techniques and Technologies*, Idea Group of Publishing, PA, USA, and *Selected Readings on Information Technology Management: Contemporary Issues*, Kelley G. (Eds.), 2009, Information Science Reference, IGI Global, PA, USA.

[138] Zulkernine, F., and Martin, P., 2007. Conceptual Framework for a Comprehensive Service Management Middleware. In Proc. of the 2nd Int. IEEE Workshop on *Service Oriented Architectures in Converging Networked Environments (SOCNE'07)* in conjunction with AINA 2007, Niagara Falls, Canada, IEEE CS Press, pp. 995-1000.

[139] Zulkernine, F., Martin, P., and Wilson, K., 2008. A Middleware Solution to Monitoring Composite Web Services-based Processes. In *Proc. of IEEE Congress on Services*

(SERVICES'08) Part II at the Workshop on Service Intelligence and Computing (SIC) of IEEE Int. Conf. on Web Services (ICWS'08), Beijing, China, IEEE CS Press, pp.149-156.

[140] Zulkernine, F., Martin, P., Craddock, C., and Wilson, K., 2008. A Policy-based Middleware for Web Services SLA Negotiation. In *Proc. of the IEEE Int. Conf. on Web Services (ICWS'08)*, Beijing, China, IEEE CS Press.

[141] Zulkernine, F., Powley, W., and Martin, P., 2009. Autonomic Management of Networked Web Services-based Processes. Accepted for publication in *Autonomic Computing and Networking*, Springer, USA.

[142] Zulkernine, F., Tian, W., Powley, W., Martin, P., Xu, T., and Zebedee, J., 2008. Autonomic Web Services Environment using a Reflective Database-Oriented Approach, *Ubiquitous Computing and Communication Journal* special issue on Autonomic Computing and Communications, UBICC, vol. ACSA, pp.1-12.

[143] Zulkernine, F., and Martin, P., (submitted in Feb. 2009). An Adaptive Negotiation Decision Support System for Service Level Agreements. In *IEEE Transactions on Services Computing*, IEEE CS Press.

Appendix A

Service.xml for WSCompany (WS2) Web service

```
<service name="WSCompany" scope="application" targetNamespace ="http://WSOMCompany.server/">
   <description>
      Company Service with Axiom
   </description>
   <module ref="pmreporter"/>
   <operation name="companyQuery0">
      <messageReceiver class="org.apache.axis2.receivers.RawXMLINOutMessageReceiver"/>
   </operation>
   <operation name="companyQuery1">
      <messageReceiver class="org.apache.axis2.receivers.RawXMLINOutMessageReceiver"/>
   </operation>
   <schema schemaNamespace="http://WSOMCompany.server/xsd"/>
   <parameter name="ServiceClass" locked="false">WSOMCompany.server.WSCompany</parameter>
</service>
```

WSCompany.wsdl (for WS2)

```
<wsdl:definitions xmlns:axis2="http://WSOMCompany.server/"
          xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
          xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
          xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
          xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
          xmlns:ns="http://WSOMCompany.server/xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
          targetNamespace="http://WSOMCompany.server/">
     <wsdl:types>
       <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="qualified"
                  elementFormDefault="qualified" targetNamespace="http://WSOMCompany.server/xsd">
          <xs:complexType name="Query0Struct">
               <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="employee">
                         <xs:complexType>
                              <xs:sequence>
                                   <xs:element name="fname" nillable="true" type="xs:string" />
                                   <xs:element name="lname" nillable="true" type="xs:string" />
                                   <xs:element name="dept_name" nillable="true" type="xs:string" />
                                   <xs:element name="dept_loc" nillable="true" type="xs:string" />
                              </xs:sequence>
                         </xs:complexType>
                    </xs:element>
               </xs:sequence>
          </xs:complexType>
          <xs:complexType name="StringArray">
               <xs:sequence>
                    <xs:element maxOccurs="unbounded" minOccurs="0" name="storenames" nillable="true"
                    type="xs:string" />
               </xs:sequence>
          </xs:complexType>
          <xs:complexType name="QueryFault">
               <xs:sequence>
                    <xs:element name="companyQueryFault" type="xs:string" />
               </xs:sequence>
          </xs:complexType>
```

Continued

```
            <xs:element name="query0param">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="PID" nillable="false" type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="companyQuery0Fault" type="ns:QueryFault" />
            <xs:element name="companyQuery0Response" type="ns:Query0Struct" />
            <xs:element name="companyQuery1Fault" type="ns:QueryFault" />
            <xs:element name="companyQuery1Response" type="ns:StringArray" />
        </xs:schema>
    </wsdl:types>
    <wsdl:message name="companyQuery0Message">
        <wsdl:part name="part1" element="ns:query0param" />
    </wsdl:message>
    <wsdl:message name="companyQuery0ResponseMessage">
        <wsdl:part name="part1" element="ns:companyQuery0Response" />
    </wsdl:message>
    <wsdl:message name="companyQuery0Fault">
        <wsdl:part name="part1" element="ns:companyQuery0Fault" />
    </wsdl:message>
    <wsdl:portType name="WSCompanyPortType">
        <wsdl:operation name="companyQuery0">
            <wsdl:input message="axis2:companyQuery0Message" wsaw:Action="urn:companyQuery0" />
            <wsdl:output message="axis2:companyQuery0ResponseMessage"
                wsaw:Action="//WSOMCompany.server/WSCompanyPortType/companyQuery0Response" />
            <wsdl:fault message="axis2:companyQuery0Fault" name="companyQuery0Fault"
                wsaw:Action="//WSOMCompany.server/WSCompanyPortType/companyQuery0/Fault/companyQ
                uery0Fault" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="WSCompanySOAP11Binding" type="axis2:WSCompanyPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
        <wsdl:operation name="companyQuery0">
            <soap:operation soapAction="urn:companyQuery0" style="document" />
            <wsdl:input><soap:body use="literal" /></wsdl:input>
            <wsdl:output><soap:body use="literal" /></wsdl:output>
            <wsdl:fault name="companyQuery0Fault"><soap:body use="literal" /></wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="WSCompanySOAP12Binding" type="axis2:WSCompanyPortType">
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
        <wsdl:operation name="companyQuery0">
            <soap12:operation soapAction="urn:companyQuery0" style="document" />
            <wsdl:input><soap12:body use="literal" /></wsdl:input>
            <wsdl:output><soap12:body use="literal" /></wsdl:output>
            <wsdl:fault name="companyQuery0Fault">
                <soap12:fault use="literal" name="companyQuery0Fault" />
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="WSCompany">
        <wsdl:port name="WSCompanySOAP11port_http" binding="axis2:WSCompanySOAP11Binding">
            <soap:address location="http://localhost:8080/axis2/services/WSCompany" />
        </wsdl:port>
        <wsdl:port name="WSCompanySOAP12port_http" binding="axis2:WSCompanySOAP12Binding">
            <soap12:address location="http://localhost:8080/axis2/services/WSCompany" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```