uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Ronald Pringadi
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S. (Computer Science)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Service Oriented Architecture for Electronic Negotiation Servers

TITRE DE LA THÈSE / TITLE OF THESIS

Morad Benyoucef
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Azzedine Boukerche

Michael Weiss

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

A Service Oriented Architecture for Electronic Negotiation Servers

Ronald Pringadi

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
Faculty of Engineering

University of Ottawa
Ottawa, Ontario K1N6N5, Canada

Your file Votre référence
ISBN: 978-0-494-25823-1
Our file Notre référence
ISBN: 978-0-494-25823-1

Abstract

As the scope of e-business and e-commerce becomes broader, electronic negotiation (e-negotiation) practice requires more attention through rigorous research by the academic community. Most of the research conducted today in the field of e-negotiation focuses on automated negotiation through strategy-enabled software agents, and on studying new agent and multi-agent frameworks that incorporate artificial intelligence (AI) learning techniques.

One of the fundamental issues consists of devising a systematic design framework for e-negotiation platforms that support the interactions of these software agents. Ideally, such platforms should enable various negotiation protocols and offer high flexibility in specifying, implementing, and deploying them. The study of a properly defined e-negotiation platform or server is very critical, yet not enough research is being conducted in this field.

The focus of this thesis is on the design, implementation, and deployment of e-negotiation servers, including: (1) a thorough study and classification of existing research projects on e-negotiation servers based on their design characteristics; (2) the specification of requirements for a generic e-negotiation server; (3) the design, development, deployment and testing of a generic e-negotiation server using a service oriented architecture (SOA) approach. We use a web service orchestration language to model and enact negotiation processes. The choice of an SOA approach is justified by the need to facilitate inter-organizational integration, in which business negotiation plays an important role.

To validate our design framework we implemented an e-negotiation server using BPEL (Business Process Execution Language). We used the server to model and enact several negotiation scenarios. This enabled us to identify guidelines and templates that make the modeling task easier and more systematic.

Publications

Pringadi, R. and Benyoucef, M., *Web Service Orchestration of E-negotiation Interactions,* Working Paper #wp06-28, School of Management, University of Ottawa, 2006.

Benyoucef, M. and Pringadi, R., *A BPEL Based implementation of Online Auctions*, IC-SOC Conference, WESOA Workshop, Chicago, Dec4, 2006, to appear.

Dedication

I dedicate this thesis to my beloved parents, my father Setianto Saputra, my mother Catharina T. Herawati, and my dear girlfriend Catherine Millar for their love, encouragements and support.

Acknowledgement

This research could not have been conducted without the support of many people. Firstly, I would like to express my appreciation to my supervisor, Dr. Morad Benyoucef, for his continuous encouragement, helpful advice and considerable patience during the completion of this thesis. Secondly, I would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding this research. Thirdly, I would also like to thank my colleagues Dr. Stefanie Beate Rinderle, Parag Sharma, and Hui Li for their moral support. Special thanks to Emma Debonno - for technically editing this thesis.

Table of Content

List of Figures

List of Tables

Glossary

The following words was compiled form the following sources: http://www.dictionary.com, http://en.wikipedia.org, Microsoft® Encarta® Reference Library 2005 software, Oxford Dictionary 2004 software, and various academic papers listed in the references section.

Architecture

The design, structure, and behavior of a computer system, hardware, or software, including the characteristics of individual components and how they interact.

Alternatives

A set of possible choices that a negotiator may choose. Alternatives can be options of possible decisions, elements to choose.

Auctioneer

Somebody who is in charge of and announces the bids at an auction which can be the initiator or the mediator.

Compromise

A settlement of a dispute in which two or more sides agree to accept less than they originally wanted.

ENS

Electronic Negotiation System. Online system that facilitates and supports negotiation.

Framework

A supporting structure around which something can be built.

Initiator

A member of the negotiation participants that start the negotiation by giving proposal. In the English auction it can refer to the seller, while in the procurement auction it can refer to the buyer.

Issue

A topic of discussion that is of particular interest in a negotiation. Each issue has a range of alternatives or options, one of which must ultimately be agreed upon by the negotiators in order to achieve a compromise.

Marketplace

A place or facility where the sellers and the buyers meet to negotiate the value of products or services exchanged, and complete the transaction based on the agreed terms.

Methodology

A system of ways of doing, teaching, studying, or implementing something.

Negotiation

Negotiation is an iterative process that involves two or more agents who communicate for the purpose of exchanging goods or services and defining the terms of the exchange.

Negotiation Servers

Software systems that provide a platform for the participants' interactions and implement a negotiation protocol such as a marketplace.

NSA

Negotiation Software Agents. A software entity that automates negotiation process on behalf of its owner/user preferences.

NSS

Negotiation Support System. A software entity that helps the user to gather information and point out the available options.

Offer

An attempt to sale or buy something that is sent by one negotiator to the other. An offer may contain one option for each issue under considerations, or some issues may not be present in the offer.

Package

A set of options that has been selected from several or all the negotiated issues. An example of a package can be a set of price, payment method, and shipping and handling.

Proposal

The act of making a suggestion, an offer or stating an intention.

Rating

A measurement system which put one option over another based on a certain value for the negotiator.

Web Server

Software that serves web pages, either static or dynamic, for the client.

Web services

A software automation service provided through the web that is described via WSDL and is capable of being accessed via standard network protocols such as but not limited to SOAP over HTTP.

Chapter 1

Introduction

While conducting business deals, parties (businesses or individuals) usually see their own interest and needs as their priority. They prefer outcomes that benefit them most. While this situation can lead to a deadlock and a failure to reach business agreements, each party is typically willing to negotiate the proposed terms and conditions of the deal. Negotiation, as defined in [1], can be formalized as an iterative communication and decision making process between two or more parties, which can be represented by two or more agents, who cannot achieve their objectives through unilateral actions and who search for consensus.

Negotiation can be a very intensive, time consuming, and costly process for conducting business; therefore, there is a demand to automate it [2]. Any negotiation conducted using computers or various electronic devices is referred to as electronic negotiation (e-negotiation). In business, e-negotiation usually refers to Internet based auctions, competitive bids for procurement, online brokerages, exchanges, cartels, and two-party negotiations [3]. Because of the strong domination of auctions in e-negotiations, many might think that e-negotiations primarily consist of auctions [4]. In this research, although we concentrate on auctions, we believe that other negotiation protocols can and should be conducted electronically.

According to Bichler et al. (2003) [1], negotiation can be viewed from many different angles depending on the field of study. These fields are computer science, information systems, economic sciences, management, law, and social sciences. Since this thesis focuses on e-negotiation servers, then computer science and information systems will be the main perspective.

The study of e-negotiation from this perspective is divided into three main categories [1]. First, negotiation support systems (NSS), which are software tools for assisting users with communication and decision-making activities, eliciting preferences, evaluating and comparing offers based on the elicited preferences, and recommending strategies [5]. Second, negotiation software agents (NSA), which are intelligent software entities that communicate with other entities and make decisions on behalf of their owner, according to

pre-programmed strategies, with or without learning capabilities [5]. NSA's main concern is about software agent's strategies, performance, and observed behavior through competition [6]. The third main category is e-negotiation servers, which consist of software that provides a platform implementing a negotiation protocol such as an electronic marketplace. In the literature, the term negotiation server also refers to negotiation media, platform, or marketplace; therefore, these terms can be used interchangeably.

Most of the research conducted today in the field of e-negotiation systems (ENS) is centered on specific techniques for software agents, negotiation strategies, and negotiation automation. Few researchers assessed the problems of how to design a platform that supports these interactions. Ideally, such platform should support various negotiation protocols [7], offer high flexibility in design [5], and foster easy development and deployment [8].

Designing a platform which addresses all the requirements mentioned above is obviously not an easy task, knowing that it will likely involve business-to-business (B2B) and application-to-application (A2A) integration. According to Basu and Kumar (2002) [9], marketplace-based architecture is a good solution for managing inter-organizational processes such as negotiation processes because marketplace based links between companies are more efficient than point-to-point links between every buyer and every supplier. Other challenges in the design process include formalizing the shared protocol [10], extracting the business logic from the server process [8], and validating the rules of negotiation.

Many companies have recently expanded their presence online. Internet based business enables them to reach more potential customers and suppliers, provide around the clock service, and reduce their operational costs. This condition has shifted the trend of the object-oriented paradigm into a service oriented paradigm, where software modules are converted into services (called web services) and published over the Internet to be used as-is or to be integrated with other applications. Because web services use XML based protocols, we have the advantage of providing a platform independent service which fosters B2B and A2A integration.

Later in this thesis, we discuss and compare previous research in the field of ENS. We then make the case for a service oriented architecture (SOA) to implement e-negotiation systems, and propose a flexible e-negotiation server that is based on web service orchestration of e-negotiation interactions.

1.1 Motivation of the Research

Nearly all academic research on e-negotiation centers on the use of software agents to assist with the negotiation process. Only a small number of researchers have addressed the challenges of designing a proper e-negotiation server which would be able to accommodate various negotiation processes and support software agent interactions. Therefore, the first aim of this research was to study the existing research work on e-negotiation servers. We reviewed, classified, and pointed out the main characteristics of the proposed designs for these systems.

After we studied the existing research work, we realized that, should we ever propose a new design for e-negotiation servers, we must address the following problems:

(1) Support various negotiation protocols

(2) Provide flexibility in design

(3) Foster easiness in system development and deployment

(4) Facilitate negotiators' interactions, enable the participation of human and software agents

(5) Permit the seamless integration of the e-negotiation server with different applications within the organization and across various organizations.

These requirements represent a big challenge. One of the software engineering approaches that we consider appropriate for designing such a system is the SOA approach. The system and its interfaces should be deployed as a series of web services, and by orchestrating different web services we can implement our negotiation processes.

1.2 Research Objectives

The research objectives can be categorized in the following three points:

Firstly, we work within a new vision of designing e-negotiation systems (servers and clients) by using an SOA approach. We focus on one component, which is the server.

We study and classify the existing e-negotiation servers' designs and architectures from a software engineering perspective

Secondly, we use a web service orchestration tool to describe (model) and to enact (run or execute) negotiation processes such as the English auction, the Dutch auction, etc. We deploy a common interface for human participants and software agents. We provide Graphical User Interfaces (GUI) for human participants to interact directly with the server or to configure and control the software agents that act on their behalf.

And thirdly, we focus on the specific task of modeling e-negotiation processes using web service orchestration languages. Thus, we identify common components in the models, which might constitute a repository that a marketplace owner can use to build an appropriate e-negotiation model. We also give some guidelines and best practices for modeling e-negotiation processes.

1.3 Thesis Organization

The rest of this thesis is organized in the following manner. Chapter two discusses the fundamental concepts of negotiations, auctions, and e-negotiations. Chapter three focuses on related work through a review and classification of existing research on e-negotiation servers. Chapter four explains the research methodology of this thesis; from supporting technologies to web service orchestration. Our e-negotiation server design and implementation is detailed in Chapter five, along with a discussion of modeling guidelines. Chapter six covers the evaluation and testing of our server. Finally, in Chapter seven we draw our conclusions and discuss future research perspectives.

Chapter 2

E-Negotiation Concepts

2.1 Negotiations

Reaching a business agreement is a complicated process. The involved parties have their own needs and will strive to reach solutions that are the most beneficial to them. To avoid deadlock, each party is usually willing to *negotiate* and reconsider their terms and conditions in order to reach a business agreement that is acceptable for both sides. Negotiation, as proposed in [1], can be defined as an iterative communication and decision making process between two or more parties, which can be represented by two or more *agents*, who cannot achieve their objectives through unilateral actions and who search for a consensus. Note that in this definition, *agent* refers to a *human* or *software* entity.

Negotiation can be divided into *business negotiation* and *non-business negotiation.* Business negotiation includes two-party negotiation (bargaining), auctions, business procurement, online brokerages, exchanges, and cartels. Dispute resolution and voting are grouped within non-business negotiation.

The two-party negotiation is one of the most common negotiation formats that we come across in everyday life. It starts with an *offer* from the seller. The buyer will then ask for a better or more reasonable offer, and the seller might adjust his/her offer to this *demand.* This process of offer and counter-offer is often referred to as *exchanging proposals.* This exchange might repeat several times until either the buyer or the seller decides to submit their *final proposal.* From here, there will be a deal if the seller agrees to the final proposal submitted by the buyer and vice versa, but there will be no deal if one rejects the other's final proposal.

Auctions are another form of negotiation [4]. They are detailed below.

2.2 Auctions

An auction is the process of buying or selling an item by offering it up for bids, taking bids, and selling it to the highest bidder. In economic theory, an auction is a part of negotiation which is meant for determining the value of a commodity through a competition. This competitive mechanism leads to a *win-lose* situation, as opposed to the two-party negotiation described earlier which is supposed to be *collaborative* and to lead to a *win-win* situation.

There are various types of auctions, but as mentioned by Basu and Kumar (2002) [9] there are at least five basic key elements which can be extracted from all of them. These elements are: (1) a *deal* which can be in various states such as negotiable offer or final offer from the buyer or seller; (2) *participants* such as buyers, sellers, auctioneers, brokers, etc.; (3) *messages* sent by the participants to modify the deal, for example bids and offers to buy or sell, to change the price; (4) *process flow* describing how the state of the deal changes as a result of the messages sent by the participants; (5) *result* messages sent to the participants as the deal changes.

Auction *scenarios* specify the rules and objects of the negotiation [11], and different auction scenarios serve different purposes. The most well known auction scenarios are the English auction, Vickrey auction, sealed bid auction, Dutch auction, and procurement auction. The variables and attributes in the auction can be outlined in a classification schemes such as the *there key formats* classification [3]. Auction scenarios and auction classification formats will be discussed in the following section.

2.2.1 Auction Scenarios

One of the most well known auction scenarios is the English auction. This auction scenario's popularity has grown mainly as a result of its adoption by the world's biggest online marketplace - eBay.com[1]. There were at least 75.4 million active users in the first quarter of 2006 either buying or selling items using eBay's English auction format[2].

---

[1] http://pages.ebay.com/aboutebay/thecompany/companyoverview.html

[2] http://investor.ebay.com/releases.cfm

The English auction, or the open cry auction as it is sometimes called, is an auction where a seller tries to sell an item. Potential buyers compete with one another to win the item. The price (i.e., bid) offered by potential buyers increases over time. At the end of the process, which can be triggered when no bids have been received for a certain period of time, or the auction process has reached its time limit, the potential buyer with the highest bid wins the item. The English auction behaves as a mechanism to ensure maximum profit for the seller. An English auction can have a *starting price* to ensure that the potential buyers bid close to the estimated value of the item. It can also have a *reserve price*, which is a mechanism that protects the seller from committing to unprofitable deals by making sure the final bid is higher or equal to the reserve price.

A variation of the English auction is called the Vickrey auction. It was proposed between 1961 and 1962 by an economist named William Vickrey. In the Vickrey auction the winning bidder pays the second highest bid. This situation eventually encourages potential buyers to outbid one another more frequently because the winner will pay less then the amount that he/she bid for.

Sealed bid auctions are held when it is impractical for bidders to prepare bids and counter-bids instantaneously and the confidentiality of the bids is important. For example, counter-bids take time to be prepared, new bids cannot be disseminated to other bidders instantaneously, or bidders are not available to participate in the auction at the same time. This approach makes most sense for Internet based auctions, especially if bidders are from around the globe. In single round sealed bid auctions, all bidders submit their bids by a deadline, and then the bids are evaluated at this deadline. In multi round sealed bid auctions, there is a deadline for each round of bids. At that deadline, either the auction is closed or a fresh round of bids is required with a new deadline. Auction *collusions* may happen if bidders are aware of each other's identity. A group of conspiring bidders can exploit this situation in order to lower the final price of an item in an auction. In order for this collusion to work, the group members must be the dominant presence in the auction. They effectively lower the price by limiting the number of bids made by the total number of participants. Instead of bidding against each other, they work with each other to keep prices low. The members of this group decide who amongst them will be the winner of the auction. Once the auction has been won, the members can divide the profits amongst each

other. Because sealed bid auctions hide the bidders' identification and allow a greater number of bidders to participate, they serve as a mechanism for minimizing collusions during the bidding competition.

The Dutch auction is a mechanism where the seller starts the auction by setting the price of the item very high. As the auction progresses, the seller lowers the price gradually until a buyer is willing to bid on the item. If there is only one item, the auction ends immediately with the bid. Otherwise, the auction continues, and the seller will keep lowering the price and buyers making bids until the last item is sold. The main purpose of the Dutch auction is to guarantee the sale of the item in case the item is perishable (e.g., flowers, fish, etc.).

The procurement auction is quite similar to the Dutch auction, except that the roles of buyer and seller are reversed. In the procurement auction, also known as B2B auction, a buyer will make a proposal to a group of sellers to provide him/her with an item. The auction starts from the lowest price possible. The buyer will then gradually increase the amount that he/she is willing to pay for the item, until one of the sellers is willing to bid for it. By letting the sellers compete with one another, this auction scenario acts as a mechanism to lower the price of procured goods and services. The procurement auction can also be varied as *open procurement* where the identity of the sellers is published or *closed procurement* where the sellers' identity remains private. The purposes for each of the auctions above can be summarized in the following table.

Auction Scenarios	Purpose	Initiator	Price
English	Ensures maximum profit for the seller	Seller	Increase
Vickrey	Encourages potential buyers to bid more frequently Enables the winner to pay less	Seller	Increase
Sealed bid	Facilitates instantaneous auction processes Minimizes the collusion among participants during the bidding process	Seller	Increase
Dutch	Guarantees the sale of the item	Seller	Decrease
Procurement	Ensures the lowest price for buying items	Buyer	Increase

Table 1: Summary of Auction Scenarios

2.2.2 Classification of Auctions: The Three-Key Format

The three-key format classification of auctions introduced by Kumar and Feldman (1998) [3] are: the interaction format, the control of bids, and the additional policies.

The interaction format refers to the interaction level of negotiation such as whether the interaction format is *direct* in a public meeting space, where the English auction is usually held or *indirect* by using a bulletin board forum, where the update for a sealed bid auction is usually being posted. In the English auction, the participants are usually able to attend the same location at the same time, and they feel comfortable with making counter bids within a few seconds. Depending on the agreement before the auction begins, some participants who are not able to attend at the same time may still be allowed to bid by a proxy. For example, a participant may ask someone else to represent him/her or does the bidding remotely such as over the phone.

The control of bids explains who holds control of the bids, such as the initiator or the rest of the participants. In English auctions, the potential buyers control the bids. Each successive bid is higher than the previous one. In a Dutch auction, when the *auctioneer* (initiator or mediator of the auction) puts up the bids, he can start at either a high or low price. At very high prices, no bidders are willing to trade whereas at very low prices, there are more buyers than inventory. This is when the auctioneer will decrease or increase the bid until the number of buyers matches the number of items in the inventory.

Additional policies such as *anonymity, restrictions on bid amount, rules for closing the auction, rules for breaking tied bids, and auction services* aim to fine-tune the auction process. *Anonymity* can vary depending on whether or not the bidder is identified, or the number of available items during the bidding process is published. For example, in Dutch auctions if the bidders know that a large inventory exists, they will be less enthusiastic to bid because they will anticipate that the prices will decrease.

Restrictions on bid amount specify the opening bid value (for example, the opening bid can be 70% of the value of competing merchandise) or the subsequent bid values (for example, the subsequent bids must augment by 5% increments). Rules for closing the auction may also vary the interactions. For instance, English auctions can finish by the agreed closing time, or after a certain interval of idle time. This interval can be several minutes in an Internet auction or a few seconds in a meeting room auction. Dutch auctions could close

at a specific, pre-determined time; when the entire inventory has been sold; when the price has fallen to a pre-specified level; or at some combination of these three conditions.

Rules for breaking tied bids are required whenever multiple bids arrive at the same time and are of the same value, but the available inventory can only satisfy some of the bids. Preference may be given to bids for a larger quantity. If the seller maintains a history of his/her auctions, the seller can decide to give preference to the bidder with whom he/she had better business relations in the past.

Auction services are services provided by the auctioneer to the sellers or buyers; for example, services that are available for the sellers are enforcing the reserve price, credit checks on the bidders, certification of the quality of the product, collection of the payment on behalf of the seller, etc. Services for the buyers can be realized with facilities such as proxy bidding and alerting registered bidders about upcoming auction events.

2.3 E-Negotiations

Business activity on the Internet is currently dominated by publicizing the business opportunity and catalog-based sales [3], but it will rapidly expand to include negotiations to settle the price of the goods or the commodities being traded. These negotiations are currently conducted by human intermediaries through various forms of auctions, bidding systems for awarding contracts, and brokerages.

The term e-negotiation has been used to describe software that employs Internet technologies and capable of supporting, aiding or replacing one or more negotiators, mediators or facilitators [12]. By using an electronic medium such as the Internet, e-negotiation can reduce business expenses, quicken the time to conduct business deals, reach more potential business partners and customers, and provide service *24 hours a day, 7 days a week.*

Besides the fact that it takes place in an electronic medium, an ENS is also characterized by its process automation using *software agents*. Software agents are "computational entities that participate in the negotiation process, each one is assumed to be capable of rating its preferences, so that it can evaluate and choose between different deals" [13].

The study of e-negotiations involves many disciplines such as computer science, information systems, economic sciences, management, law, and social sciences [1]. The

computer science field studies e-negotiation regarding its algorithm, data mining, artificial intelligence, automated distributed systems, and negotiation software platforms. The information systems field covers issues pertaining to decision and negotiation support systems, workflow models, electronic support agents, and electronic markets. Topics such as econometrics, experimental economics, management and decision science, game and auction theory, and negotiation analysis are discussed in economic sciences and management. The legal, psychological, sociological, and cultural influences of negotiation are things that interest the law and social sciences fields of study. The interaction of these diverse fields of study can be seen in the following figure.

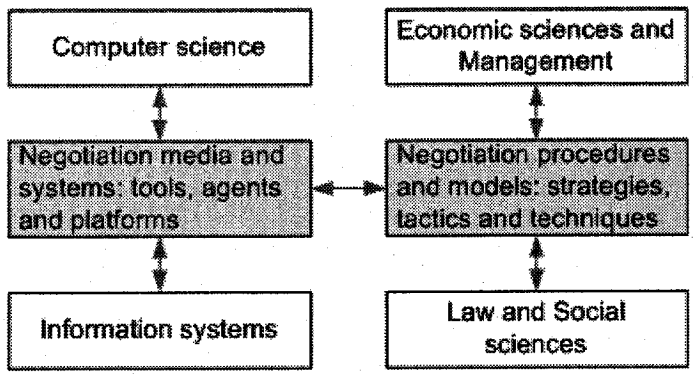

Figure 1: Negotiation research areas, their results and key influences - modified [1]

The variety of the involved disciplines and perspectives has created different terminologies, definitions, notations, concepts and formulations. As a result, interdisciplinary cooperation among concerned fields of study suffers from inconsistencies and contradictions [14]. Therefore, classifications are needed to provide a conceptual framework where protocols and strategies for negotiation can be organized.

2.3.1 Classification of E-Negotiations

Several academic papers discussed e-negotiation classifications. These classifications are useful for the full comprehension of the concepts and issues of e-negotiations, and for structuring and simplifying the design of ENS.

2.3.1.1 Parameters of the Negotiation Space

In 2003, Lomuscio et al. [13] proposed a classification scheme based on the parameters of the negotiation space, which consist of: the *cardinality of the negotiation*, *agents' characteristics*, *environmental and goods' characteristics*, *event parameters*, *information parameters*, and *allocation parameters*.

Cardinality of the negotiation indicates whether the negotiation involves a single or multiple issues. A single issue means there is only one issue on which an agreement is to be made, such as price, while multiple issues means there are other issues such as quality, warranty, delivery, etc. Cardinality of the negotiation also refers to the interaction between agents which can result in one-to-one, one-to-many, or many-to-many relationship.

Agents' characteristics specify the agents' *role*, *rationality*, *knowledge*, *commitment*, *social behavior*, and *bidding strategy*. An agent can have a *role* as a seller, a buyer, a host, a support system, or an expert system. Agents' *rationality* refers to the computational ability of an agent, which can be perfect if it is able to perform large computations in constant time, or bounded if it uses approximation. Agents' *knowledge* can also vary depending on their own valuation or on how other agents will valuate the item. Various levels of the agent *commitment* also affect the negotiation, such as whether an agent should stop bidding for similar items and wait, or retract the bids it made. Some agents have *social behavi*ors, therefore they can act individually or in a group, which influences the result of the negotiation. Agents can also vary in their *bidding strategy* which is based on combining the aspects above.

In addition to the role of an agent, Bartolini et al. (2002) [10] consider that if an agent plays the role of a negotiation host, then it has several sub-roles which are: (1) the gatekeeper, the entity that enforces policy governing admission to the negotiation; (2) the proposal validator, the entity that ensures that a proposal is well formed, and acceptable to the required format; (3) the protocol enforcer, the one which ensures participants' proposals are posted and withdrawn according to the rules; (4) the agreement maker which ensures that agreements are well formed; (5) the information updater, an entity that notifies the participants of the current states or changes; and (6) the negotiation terminator, the one that has the authority to terminate the negotiation according to the specified rule.

The *environment* also influences the agent in evaluating the *private* value of the goods, such as vintage items and art works, or *public* value such as bonds and shares. The *nature of the good* can be *discrete* or *continuous*; for example, negotiation over quantity is discrete, but negotiation over time of delivery is something continuous because it involves the date, month, year, hours, minutes, and seconds.

Event parameters consist of *bid validity*, *bid visibility*, *clearing schedule and time-outs*, and *quotes schedule*. *Bid validity* specifies whether a bid is acceptable or not according to the procedure; in an English auction each successive bid must be higher than the previous one. *Bid visibility* rules whether or not to keep bidders' identification confidential (to minimize collusion). *Clearing schedule* specifies the allocation of the traded goods or services between the seller and the buyer. *Timeout* specifies the rules for the final clearing schedule, which can be triggered by the agreed negotiation time limit, or a certain idle time from participants. The *quotes schedule* needs to follow a certain rule so that it is not too frequent and does not slow down the mechanism.

Information parameters govern the information passed among the agents such as *price quotes*, *transaction history*, and *arguments*. *Price quotes* deal with sellers or buyers request of the current price during the negotiation. *Transaction history* provides credibility and audit control for the host of the negotiation, and the *arguments* give the process clarity.

Finally, the *allocation parameters* specify the winner of an auction when more than one agent have shown an interest in the good. This can only apply in one-to-many, and many-to-many relationships.

2.3.1.2 Classification of Auctions – The Montreal Taxonomy

The Montreal Taxonomy allows for the exact characterization and comparison of a broad variety of electronic negotiation designs and systems, ranging from auctions to bilateral bargaining tables [15]. Essentially, the *Montreal Taxonomy* classifies e-negotiation from the perspective of a system designer. This classification is based on various criteria which can be selected by a system designer while he/she is designing the negotiation protocols; also referred to as *explicit-endogenous* criteria. These criteria can be grouped into six

negotiation phases: *offer specification, offer submission, offer analysis, offer matching, offer allocation, and offer acceptance.*

During the *"offer specification"* phase, the agent specifies details on the transaction object. This phase can be influenced by: the value of an object (exact or relative such as: bigger-than or less-than); whether the structure of the transacted object is fixed or flexible; whether or not an agent can neglect some of the negotiation attributes; and whether an agent can submit several offers at the same time.

During the *"offer submission"* phase, an offer is being submitted to the counterpart agent. In order to be valid, an offer must be submitted by a party or multiple parties, the agent that submits the first offer must behave as a buyer, a seller, or both (e.g. double-auction or stock exchange), and the submission phase must be executed following a predetermined event, a schedule, or at anytime.

During the *"offer analysis"* phase, the validity of incoming offers is evaluated. For example, the system analyses whether each incoming offer has a higher (e.g. English auction) or lower (e.g. procurement auction) value than the previous offer. Alternatively, the system can use a certain threshold value to compare each incoming offer.

During the *"offer matching"* stage, potential offers are matched with the agent's ratings and preferences. This process depends on: whether this stage happens according to a schedule or after a certain time delay; whether or not the offers are rated and sorted to a ranking scheme; whether or not there is a preference for the partners that submit the offer; and whether or not there is a tie-braking rule to solve conflicts of offers.

During the "offer allocation" phase, the system determines "who gets what". This process can be influenced by: whether or not the winning price is fixed or follows certain rules (e.g. Vickrey auction, discounts, etc.); and whether agents can select a final configuration freely among the possible options, or a third party decides.

The "offer acceptance" phase is the last of the six phases of negotiation as defined in [15]. An offer can be accepted or rejected depending on whether the agents are committed and bounded to their offers, and whether they are allowed to retract their offers.

2.3.1.3 Taxonomy of E-Negotiation Rules

There is a classification based on e-negotiation rules which helps govern the validity and the accuracy of the negotiation process. There are six rules in the taxonomy proposed by Bartolini et al. (2002) [16] including *participant admission rules*, *proposal validity rules*, *protocol enforcement rules*, *status and participants' update rules*, *agreement formation rules*, and *negotiation lifecycle rules*.

The admission rules govern the participant's admission to the negotiation process. The proposal validity rules enforce the fact that any submitted proposal is compliant with the negotiation template. The protocol enforcement rules determine when a participant may post a proposal, what new proposals may be posted or withdrawn, and govern proposals' expiration time. The status and participants' update rules specify how the parameters of the negotiation change on occurrence of certain events; select which participants can view a given proposal; and decide the content of the notification message (such as: accepted, submitted, agreed, or show summary). The agreement formation rules determine which proposals are compatible; which agreements should be formed; and which tie-breaking rules should be applied. Lastly, there are the negotiation lifecycle rules which specify the negotiation termination rule where no more proposals may be posted.

2.3.2 General Design of ENS

According to Lomuscio et al. (2003) there are two important basic components when designing an automated ENS, which are *the negotiation protocol* and *the negotiation strategies* [13].

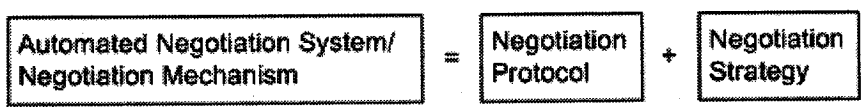

Figure 2: Automated negotiation system

The negotiation protocol defines the rules of interaction between the negotiation participants (agents) and sequences of the allowed offers. In general, agents must reach an agreement on the negotiation protocol before negotiation begins. In addition to this, Bichler

et al. (2003) distinguished three different levels of structure in a negotiation protocol which are: *unstructured*, *semi-structured* and *structured* [1]. In *unstructured* negotiations, participants do not follow any protocol or rules during the process (for example: face to face negotiations). Fully unstructured negotiations are rare, considering that participants always use implicit rules for guidance. In *semi-structured* negotiations, participants use certain rules that are not fully defined so that they can have some flexibility (for example: negotiations supported by NSS). *Structured* negotiations follow a set of rules which fully define the process and allowable activities (for example: auctions).

ENS are further divided to *unsupported*, *supported* and *automated* [1]. In *unsupported* e-negotiations, the participants control and manage all tasks without any support or advice from the information system. The *supported* e-negotiations involve the help of information systems for decision making. The *automated* e-negotiations are e-negotiations that include software agents to make decisions based on the assigned strategy and control the entire process.

The negotiation strategy is a sequence of action plans for the agents to behave properly whenever they receive offers or have to provide responses during the negotiation. Usually there are many strategies for a particular protocol, each of which may produce a very different outcome.

Bichler, Kersten et al. (2003) pointed out that an ENS can be divided into three components: *Negotiation Support Systems* (NSS), *Negotiation Software Agents* (NSA), and *the negotiation servers*.

Negotiation Support Systems (NSS) are software tools that support negotiation activities such as eliciting preferences, evaluating and comparing offers based on the elicited preferences, and recommending strategies. NSS's main functions are to assist users with information gathering, problem structuring and generating alternatives for decision-making activities; therefore, NSS are also referred to as Decision Support Systems (DSS) of negotiation.

Negotiation software agents (NSA) are intelligent software entities which communicate with other entities and make decisions on behalf of the owner, with or without learning capabilities [5]. NSA's main concerns are about the agent strategies, performance and observing behaviors through competition [6].

E-negotiation servers are software systems that provide a platform for the participants' interactions and implement a negotiation protocol such as a marketplace. In the literature, the term *negotiation server* also refers to *negotiation media*, *platform*, or *marketplace*; therefore, these terms can be used interchangeably.

The relation between the negotiation medium (the server) and the negotiation support system can be illustrated in Figure 3. The negotiators have to use the negotiation server but can choose whether or not to use the additional service (such as: advisors, experts, DSS, or NSS). Usually the negotiation server is tightly coupled with the NSS, which makes some confusion when differentiating them. In this thesis, we focused on e-negotiation servers; therefore, we will review and study the existing e-negotiation servers that can be found in the literature in the next chapter.

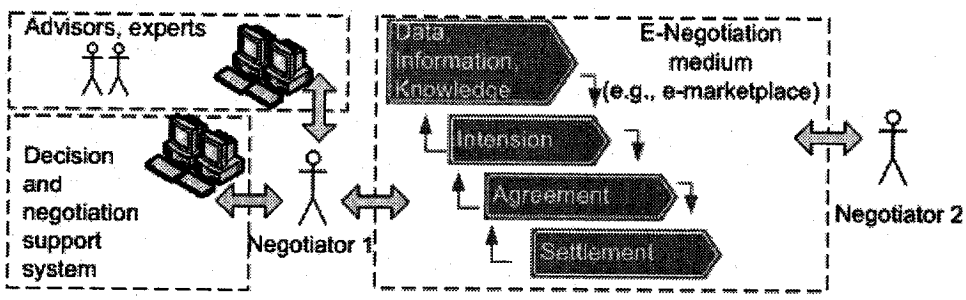

Figure 3: E-negotiations and two categories of the systems: media and support [1]

Chapter 3

E-Negotiation Servers:

A Classification of the Literature

This chapter covers some of the leading research that has been done on e-negotiation servers from a software engineering perspective and discuses and classifies the characteristics and best practices that can be used to improve future design.

3.1 Related Work

The subsequent sections will review and comment on the existing designs approaches and frameworks for e-negotiation servers that can be found in academic papers. Their classification will be given in Section 3.2 and will be summarized in Table 4.

3.1.1 GEE

Generic Experimentation Engine (GEE) [7] is a web-based application that is designed to support various scenarios in game theory. In order to run GEE the user only requires a java applet enabled browser, but the *game designer* is required to be familiar with the JPython[3] programming language.

The authors presented an example of a continuous double auction running at the core of GEE engine. At the end of each round, the players receive a set of public and private information. The game ends when it reaches a certain number of rounds or reaches its time limit. At the end of the game, players receive scores as monetary gain. The methods that can be called on GEE include ask, RFQ, bid, final allocation, and price transfer.

In terms of architecture, GEE was designed using server-side Java technologies for easy deployment. The architecture of GEE supports three different modes, which are the pull mode, push mode, and hybrid mode (Figure 4). The pull mode is a method where the

[3] www.python.org

information is sent to the players' web browser only upon request. This mode is meant to keep the information up-to-date, which relies on web pages to reload the player's browser. The push mode is the method where the information is sent from the server to the client using a Java applet. Finally, the hybrid mode lies between the other two modes. It uses a small Java applet to listen for events on the server (through a TCP/IP socket) and performs a browser reload only when needed. The JPython script (see Figure 4) represents a high level description of the economic game to be implemented. This represents one of the earlier attempts to isolate a protocol from an e-negotiation server.

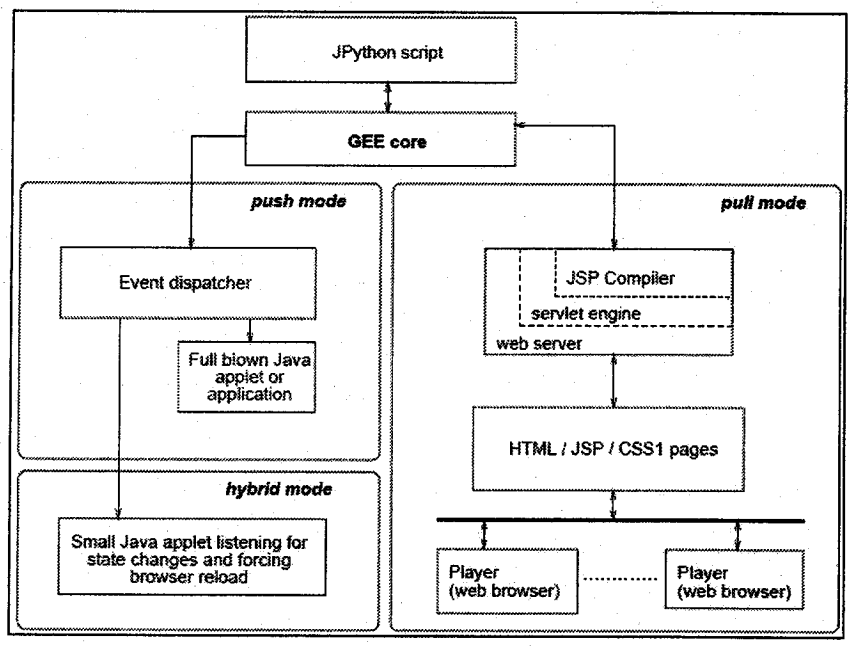

Figure 4: GEE architecture [7]

3.1.2 GNP

A successor to GEE, called *Generic Negotiation Platform* (GNP) [7] is aimed at overcoming GEE's limitations. GNP provides pervasive error handling; a registration tool to manage access and customization; a persistent management tool; and fine-grained time management. GNP utilizes reusable classes, persistent objects, and files assembled within a negotiation toolkit. The negotiation designer in GNP does not need any programming knowledge except for the JPython scripting language. The main difference between GEE

and GNP is that GEE is a game-oriented engine, while GNP is an auction-oriented platform. Differences between GEE and GNP are summarized in Table 2.

Category	GEE	GNP
Platform service	Basic error handling	Pervasive error handling Registration tool Persistent management tool
Time Management	Time delay for every round	Negotiation process delay, divided into several limited time periods based on the negotiation phases Relative time delay and absolute date calculation
Management for reusable objects	Ad-hoc/unmanaged	Reusable objects are organized into a set of negotiation toolkit
Designer's interface	Need knowledge of programming and economic market design	No need for knowledge in programming

Table 2: Changes from GEE to GNP

GNP is implemented using Enterprise Java Beans (EJB), Servlet/JSP, RMI/CORBA, and a relational database management system (RDBMS) as part of its main application server, as seen in Figure 5. GNP's application logic is composed of a JPython Script to connect to the EJB, negotiation toolkits, and a state chart description of the negotiation algorithm. The negotiation toolkit level 2 is a service for analyzing negotiation rules that can process a formalized description of negotiation processes. The negotiation toolkit 1 is a toolkit that manages frequently used components or objects. The state chart description of negotiation algorithms was not actually implemented but was part of the proposed conceptual architecture of GNP. The requirement at the client side is quite simple; it only needs a web browser that supports Java applet plug-in and XML/CSS parsing.

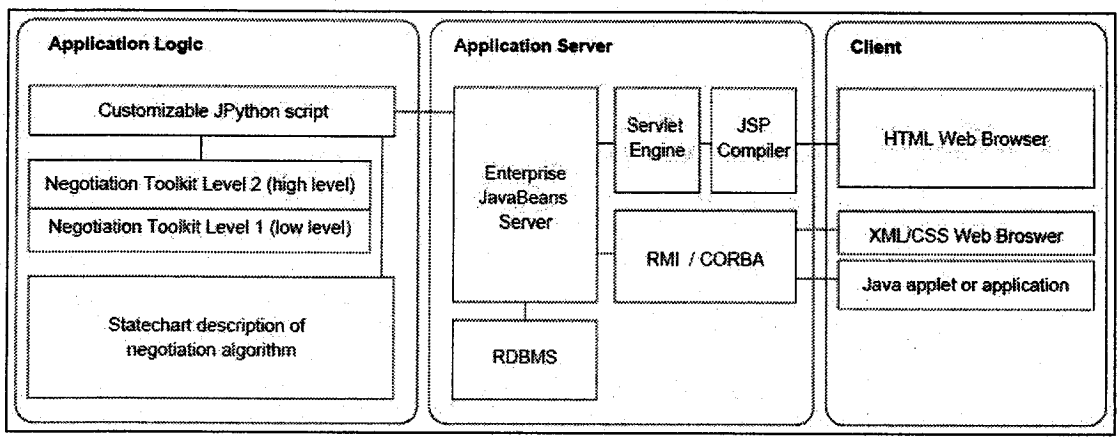

Figure 5: GNP architecture [7]

In our evaluation, GNP provides a flexible platform for implementing various types of auctions. It does not impose many system requirements on the client side. The relocation of the business rules from the main system design to XML and the reorganization of reusable objects into a toolkit give generic customizable feature to the architecture. However, GNP does not provide any NSS or negotiation automatization using software agents.

3.1.3 A Multi Agent Platform for E-negotiations

Bartolini and Preist et al. (2002) designed a multi agent platform for e-negotiations using JADE[4] (Java Agent Development Framework) [10]. The authors basically divided the agent role in the negotiations into two parts: participant and host. The participants are entities that try to reach an agreement, while the host is the entity responsible for enforcing the protocol and the rules of negotiation. The negotiation participants' main obligations are to submit proposals according to the rules defined by the negotiation host. The negotiation host is responsible for the implementation of rules and their enforcement, and the execution, resolution and termination of the negotiation process. The negotiation host's role is divided into several sub-roles, each one characterized by a set of rules, as summarized in Table 3. The relationships between the sub-roles are seen in Figure 6.

[4] jade.tilab.com

Sub-role name	Descriptions	Rules
The gatekeeper	entity that enforces policy governing admission to the negotiation	participant admission rules
The proposal validator	entity that ensures that a proposal is well formed, and acceptable to the required format	proposal validity rules
The protocol enforcer	the one that ensures participants' proposals are posted and withdrawn according to the rules	protocol enforcement rules
The agreement maker	entity that ensures that agreement are well formed	agreement formation rules
The information updater	entity that notifies the participants of the current states or changes	status and participants update rules
The negotiation terminator	the one that has the authority to terminate the negotiation according to the specified rule	negotiation lifecycle rules

Table 3: Sub-roles of a negotiation host, descriptions and rules

This platform was designed as a generic interaction framework which is built as a shared protocol. This protocol can be parameterized with different sets of negotiation rules. By choosing different sets of rules, different negotiation mechanisms can be implemented. The general negotiation protocol covered by the authors is the English auction. The protocol uses a negotiation locale, a medium which acts as a blackboard where all the events and changes are posted (shares the same information or database), and a negotiation template, a fixed set of rules that the participants agreed to during the registration process. The participants then submit the negotiation proposals and are cross-checked with the negotiation rules. Rules which define a specific protocol are validated using Java Expert System Shell (JESS)[5] and can be defined at any time.

[5] www.jessrules.com

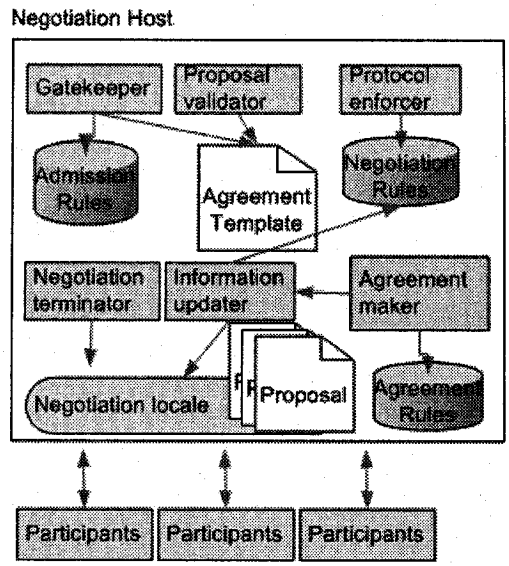

Figure 6: Bartolini and Preist's multi agent platform architecture: sub-roles & relationships - modified [10]

The layered design architecture of Bartolini and Preist's multi agent platform is quite simple. The system is built on top of JADE (a multi-agent-oriented platform which complies with the FIPA abstract architecture); on top of this platform lays the negotiation host and negotiation participants, and the general negotiation protocol (the English auction). The participants will then submit the negotiation proposals and will be validated with the negotiation rules (see Figure 7). The negotiating agents have a simple strategy to automatically bid whenever the highest bidder is not them and the price range is still on their budget (also known as automatic proxy bidding).

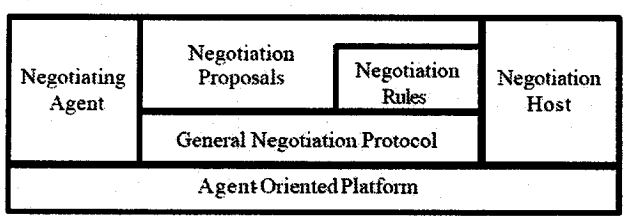

Figure 7: JADE-based architecture of negotiation framework [10]

Our overall evaluation of this platform concludes that it is flexible enough that only the general interaction framework needs to be agreed to in advance and explicitly hard-coded into the software agents. Because the system is designed on top of JADE, it uses an

agreed upon protocol specification between agents and can then be extended for future work such as developing negotiation strategies. The research only focuses on the shared protocols over agents without developing any NSS. At the communication level, the agent-oriented platform exchanges messages using Java-RMI which is a proprietary protocol. In order to be able to participate in this system the agents must have the same environment. We prefer to use a platform independent solution such as web services.

3.1.4 ANTS: an API for creating negotiation applications

ANTS is a Java API for developing negotiation platforms for negotiating software agents [8]. Inspired by Contract-Net [17], ANTS was designed to provide a generic software architecture for contract-based negotiations to help application developers in their work. The API supports one-to-one, one-to-many, and many-to-many negotiation cardinalities. It also supports an anti-deadlock system by using a limited delay to specify the maximum time for the participant to respond to the contract offered by his/her opponent (i.e., negotiation partner).

ANTS can be used in several negotiation problems with the help of an XML file to set its configuration parameters. It is characterized by using a coordination of multiple agents; however, the participants can choose whether or not to have an agent to help them (see Figure 8).

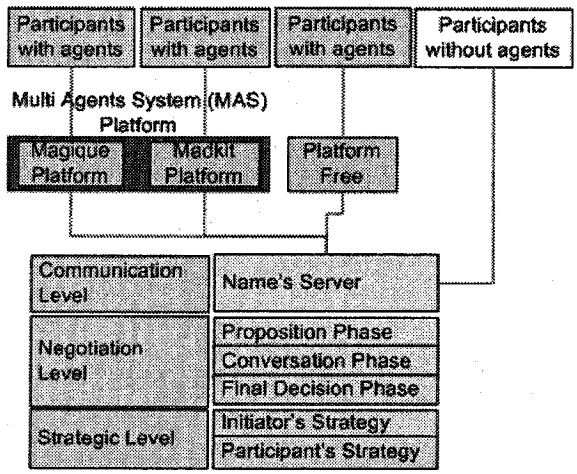

Figure 8: The three layers architecture of ANTS

From an architectural point of view, ANTS has three levels: the communication level, negotiation level, and strategic level. The communication level facilitates agents' communication. In order to achieve this, the agents have to register with the "Name's Server" first (see Figure 8). Upon a connection, an agent will receive a list of other agents that are already on the server and the other agents will be notified that there is a new participant joining the negotiation. Apart from that, the Name's Server is also responsible for notifying other agents if an agent decides to disconnect. The messages are sent in a centralized way if agents are on the same computer or in a distributed way if they are on different computers.

The negotiation level contains the negotiation protocol and data structure management. The negotiation level also facilitates negotiation conversations between agents to evolve their knowledge, for example: an initiator agent may propose a contract, and the participating agents can reject it, accept it, or propose a modification, etc. (Figure 9). This level also divides into proposition phase, conversation phase, and the final decision phase. In the proposition phase, the initiator may propose a contract to the participants and awaits their answer. In their response to the proposition, each participant answers whether to agree or reject it. The conversation phase only happens if there are not enough participants who agreed to the contract proposition. A conversation between the initiator and participants occurs while the modification propositions are exchanged. If the initiator proposes a new contract to the participants, then the cycle will go back to the proposition phase. The final decision phase is the phase during which to confirm or cancel the contract. This decision is taken by the initiator in response to participants' answers.

At the strategic level, there are two kinds of strategies available for the initiator and the participants. The strategy for the initiator helps the user to decide, confirm, or cancel the contract and to choose among the different modification propositions. The strategy for the participants helps the user decide whether to accept a proposition or not, and to choose a modification proposal in case of a request for modification in the proposal.

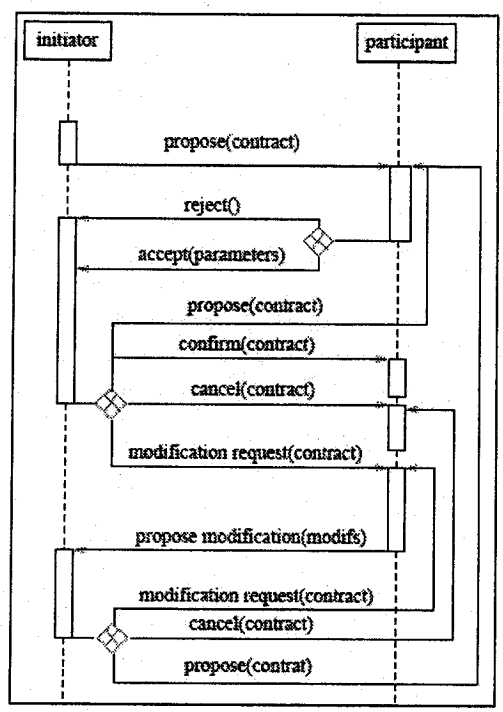

Figure 9: Negotiation protocol of ANTS

Our evaluation of ANTS is fairly positive considering the system supports multiple agents and multiple negotiation cardinalities. Although ANTS does not provide any NSS, the flexibility offered by the design of the system is good. Design features include the separation of the negotiation parameters into external XML files and the freedom to choose whether or not the participant is represented by an agent. The developed agent strategy is still quite simple, including making priority preferences. Apart from those are mentioned, the author does not specify the exact nature of ANTS's operation medium; for example, whether it is over the Internet or a local application. Judging from the screenshots provided, we can conclude that this work is an offline application-based system used only for experimentation purposes.

3.1.5 Invite

Invite [18], introduced as a software platform for multiprotocol e-negotiations, is a three-tier Internet-based application for mediating negotiation processes. Invite enables participants (buyer or seller) to flexibly negotiate many issues as well as adding new issue to the negotiation. In general, Invite "provides a virtual negotiation table allowing users to ex-

change text-based offers and messages. It facilitates communication among negotiators but does not provide any type of decision support such as analysis, visualization, or simulation" [19].

Invite's design is primarily constructed around a set of negotiation activities, components, web pages, and protocols. The negotiation protocols are a set of rules that govern activities which are allowed in every state, such as the expected input and output. The components are programs that help generate output or get input from the user. As shown in Figure 10, there are two components (Component1 and Component2) that work together with page composer, and produce two outputs (C1 and C2) which become a page that is displayed to the user in a certain activity.

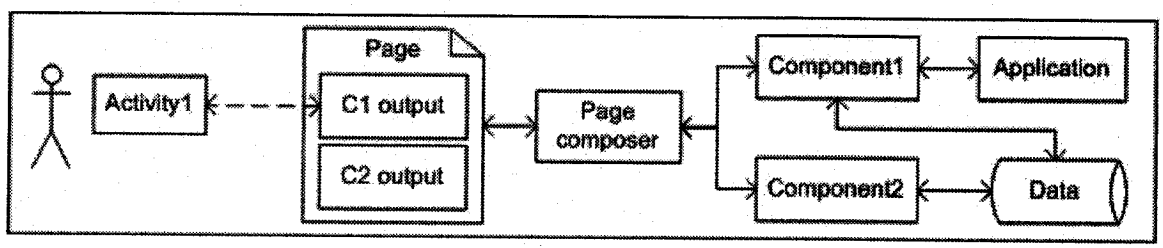

Figure 10: Activities , components and pages [18]

The three layers architecture design of Invite is shown in Figure 11. The client-layer only requires a web browser. The front-end layer or the application server is implemented with a set of fusebox[6] components. The negotiation controller is the part that synchronizes the components, protocols and the page composer to display the correct page for the current state. The back-end layer or the database system is implemented in MySQL. The database is divided into two sections for security and portability; the system database and the user database. The user database contains information about users, states, and histories of a negotiation instance, while the system database contains information about cases, sides, issues and options in the negotiation.

[6] Macromedia ColdFusion MX - www.adobe.com/products/coldfusion/

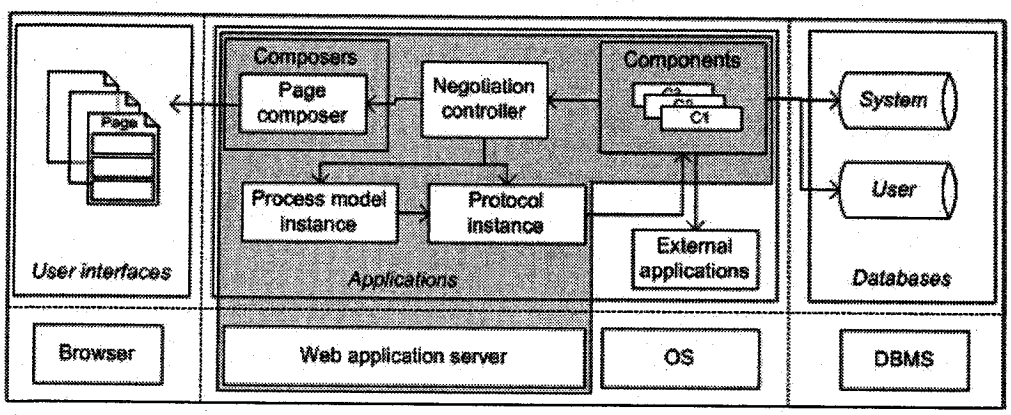

Figure 11: The Invite system architecture [18]

Our overall evaluation of this system concludes that this negotiation platform is fairly simple and can facilitate a two-party negotiation with a one-to-one relationship. The flexibility lies in the ability to add new issues to the negotiation during the negotiation phase.

3.1.6 Inspire

INSPIRE (InterNeg Support Program for International Research Experiments) is the first academic NSS for two-party negotiation with multi-attribute support deployed on the web [20]. The negotiation process in INSPIRE consists of three distinct phases: pre-negotiation, negotiation, and post-settlement [21].

During the pre-negotiation phase, the NSS helps negotiators to understand the cases, the issues involved and their potential values. Then the negotiators will set their preference by using the NSS to rate the issues and options (such as: price, delivery time, warranty, etc.). These preferences will be used as a utility function for each negotiator. Second, during the negotiation phase, negotiators use the NSS to build offers, evaluate counteroffers based on the utility function, and exchange messages. The NSS also helps the negotiators to view their negotiation history in a graphical representation and to update their preferences by changing the rates of the issues and options. Finally, in the post-settlement phase, the NSS acts as a mediator and checks for the Pareto-optimality of the agreement. If the system decides that the agreement is not optimal, it suggests different alternatives to the negotiators.

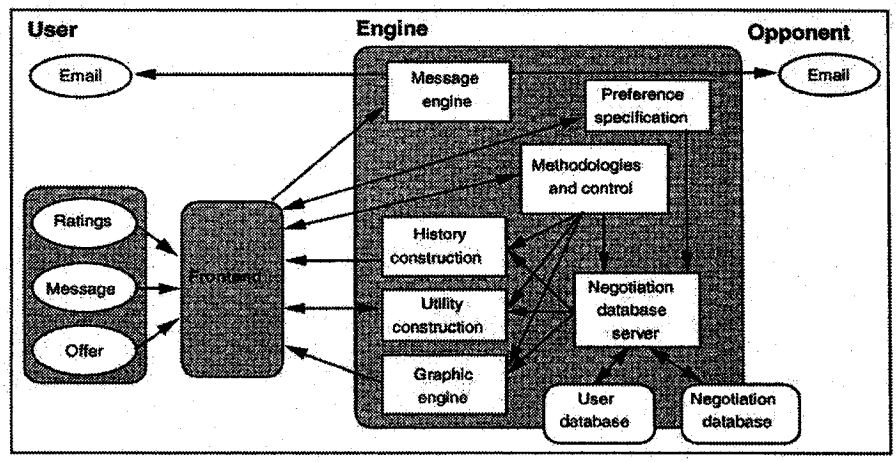

Figure 12: INSPIRE's architecture: individual support Mode [20]

INSPIRE is implemented as a client-server application in which the client is referred to as the frontend and the sever as the engine (see Figure 12). The frontend is visible to the user, composed of HTML files, Java Scripts, and Java Applets. This component takes care of the users' rating preferences input, messages, and offers. The engine is a collection of programs in C++ that can be invoked through a web server using a CGI protocol. This component deals with the preferences utility construction, graphic generation from the history, negotiation methodology and control, and the databases.

INSPIRE has several drawbacks such as a fixed number of negotiation issues and options and the inability to add more participants to the negotiation [19]. This work can be classified as a tightly-coupled e-negotiation media with NSS because the NSS is built on top of the negotiation platform.

3.1.7 Aspire

A successor to INSPIRE is Aspire [21], which is an ENS that combines INSPIRE's NSS with a software agent called Atin. Atin is built on top of INSPIRE so that it can retrieve the information from the negotiation knowledge-base and make appropriate suggestions (if any) to the negotiator. Figure 13 shows a summary of how Atin works for each phase in INSPIRE.

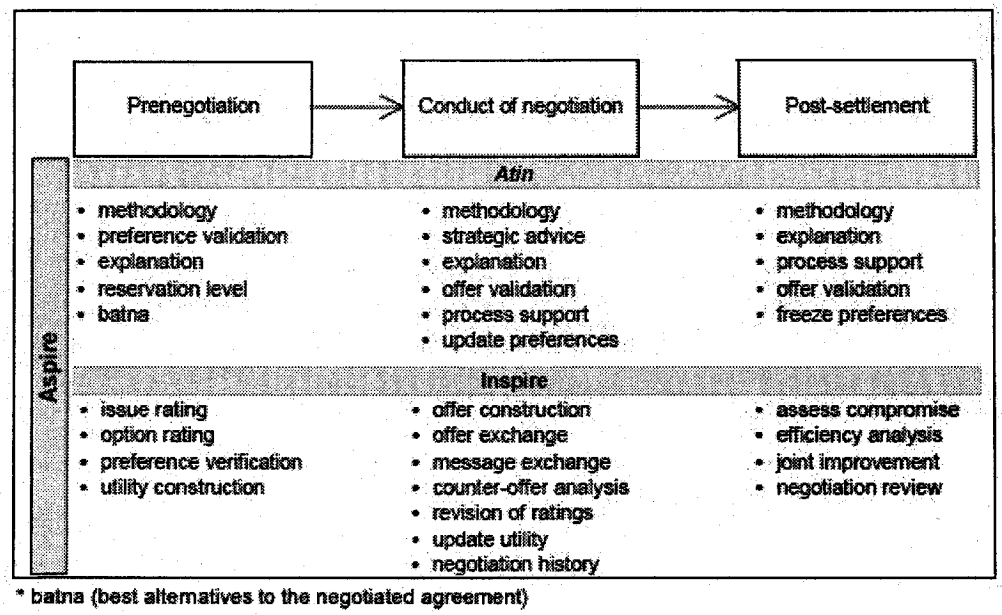

Figure 13: Aspire's support in the three negotiation phases [21]

Atin can access the results from the INSPIRE system by using a data conversion tool. After that, it validates the conditions within the inference engine with the help of JESS, records all the information transaction logs on its own database, and gives suggestions to the user. Atin's engine is written in PHP and C++. It can extract data from the MySQL database, invoke JESS and prepare the information, or request the user's input. This architectural design is summarized in Figure 14.

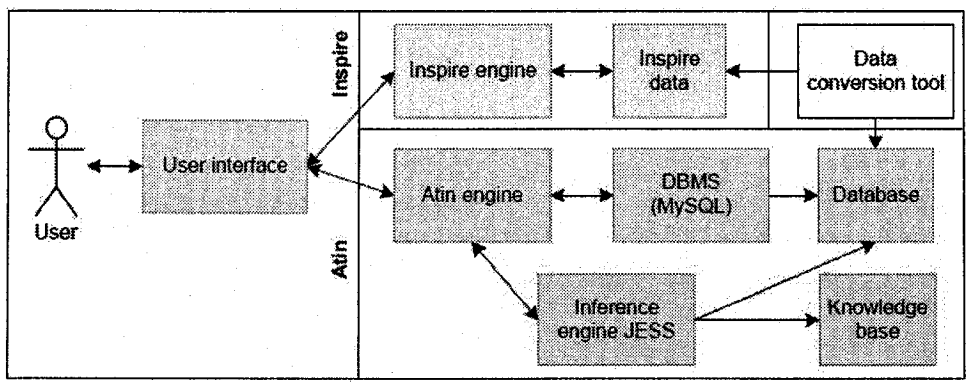

Figure 14: High level design of Aspire [21]

From the architectural point of view, Aspire is a three-tier Internet-based application. Its main purpose is to extend the INSPIRE system with an NSA. Although claimed to

possess a loosely coupled design, Atin is connected directly to INSPIRE's user database (using a conversion tool).

3.1.8 Attribute-based Negotiation Process Composer

Kim and Segev (2005) proposed an attribute-based negotiation process composer [5], which is a software tool that enables the negotiation designer to easily generate a set of BPEL[7] (Business Process Execution Language) constructs. These constructs are useful to easily create a new e-negotiation marketplace and orchestrate its interactions in a web service environment. The software tool's main function is to customize, generate, and validate a series of constructs for BPEL processes (Figure 15). Beside this software tool, the authors did not provide any user interface, such as dynamic web pages that can access the web service for the negotiating users (buyers or sellers).

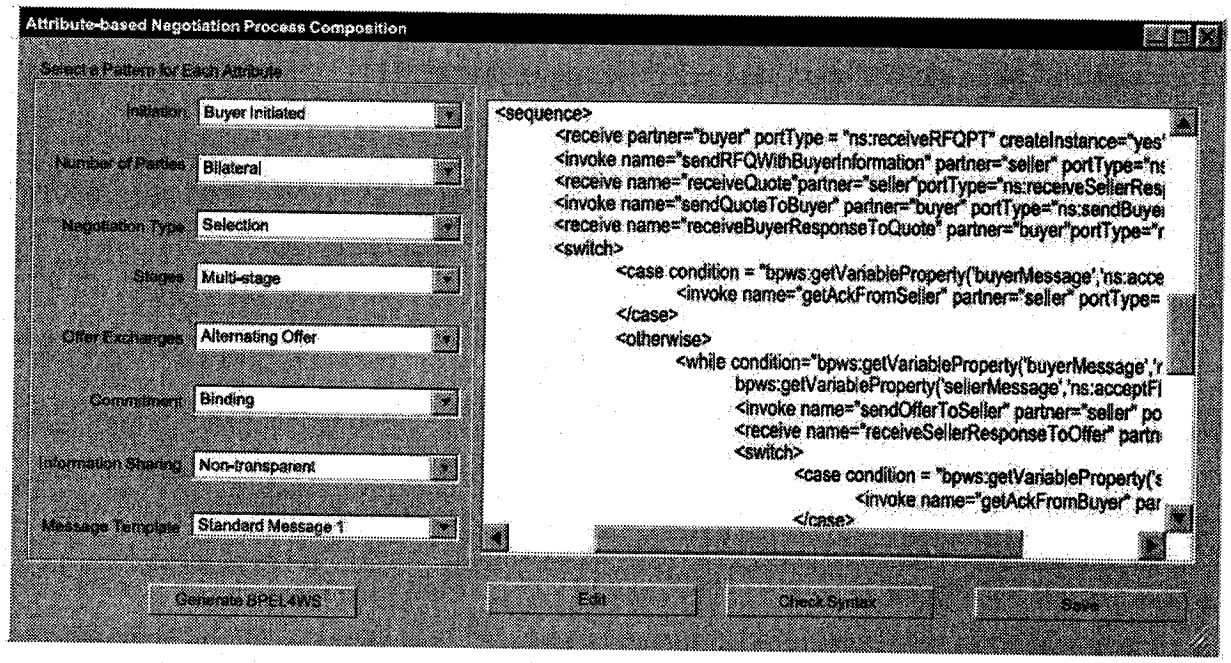

Figure 15: User interface for attribute-based negotiation process composition [5] - modified

We think that this work has presented an innovative and interesting idea. The work however is still incomplete. We understand that after the generation of BPEL constructs,

[7] www.ibm.com/developerworks/library/ws-bpel

they still need to be complied and deployed on the BPEL engine (Figure 16). However, this can become a repetitive and time consuming process for slight modifications. In our project, we prefer to model the auction scenarios (English, Dutch, or procurement) and store the attributes in an external configuration file, such as an XML file; this way we can avoid re-compilation and re-deployment after minor changes.

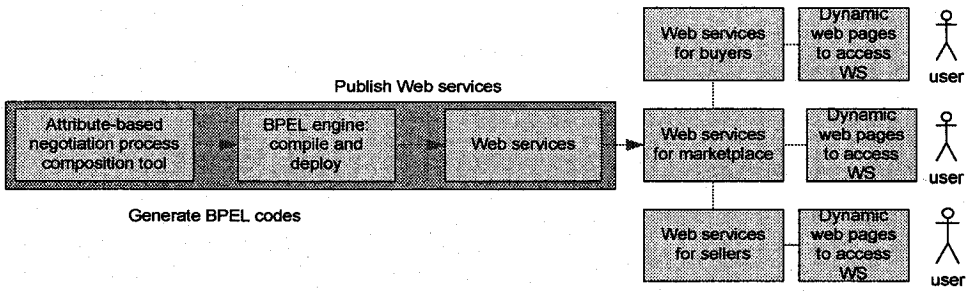

Figure 16: BPEL process development life cycle.

3.1.9 Negotiation Platform for Auction Reference Model

Rolli and Eberhart (2005) proposed a conceptual model for Internet-based auctions called Auction Reference Model (ARM) [22]. This model acts as a reference to describe and run auctions. There are three layers in the model's architecture that consist of auction data, auction mechanism, and auction participants. To support this architecture, the authors implemented a system prototype using Collaxa's[8] BPEL4WS graphical modeling editor. The files produced by this tool are compiled using BPEL2Java[9] then executed within a Java environment.

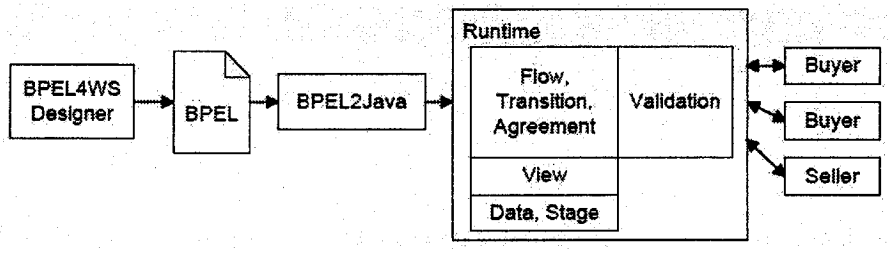

Figure 17: ARM's negotiation platform [22]

[8] Collaxa, a start-up company, was acquired by Oracle Corporation on the 29th of June, 2004 - www.internetnews.com/bus-news/article.php/3374851

[9] http://www.eclipse.org/stp/

Based on the listing code given in this paper, we deduce that the authors used BPEL4WS for modeling the general negotiation activities only. However, the modeling needs to be completed with Java code, which is not entirely convenient for the negotiation designer. The interactions between the platform and participants are carried out using Java-RMI[10]. However, this paper proposed a unique idea of composing a negotiation process by using a model driven architecture.

3.2 Classification

We classified the related work on e-negotiation servers based on the following elements: the negotiation scenario, the availability of NSS or NSA, the software quality attributes, and the architecture. The subsequent paragraphs explain these elements and present the classification (see Table 4).

3.2.1 Negotiation Scenario

Every e-negotiation server is usually intended to facilitate a certain negotiation scenario. The most commonly used negotiation scenarios are: two-party negotiation, English auction, Dutch auction, and procurement auction.

The two-party negotiation, which is also referred to as bargaining, is a negotiation between two entities to reach an agreement by submitting their proposals and terms and conditions. The role of electronic media in this case is to facilitate, optimize, and organize the phases of the negotiation in a systematic way. New issues may be introduced during the negotiation process, and some proposals are usually allowed to be exchanged in an unstructured format such as e-mail. That is the reason why two-party negotiation is categorized as semi-structured by Bichler and Kersten, et al. (2003) [1]. According to Su et al. (2000) [23], the challenges in designing a bargaining e-negotiation server are its semi-structured

[10] java.sun.com/products/jdk/rmi/

nature and the involvement of artificial intelligence (which can be part of an NSA) for automation in resolving conflicts (such as rating the issues and decision making processes).

Auction is a case of a fully structured negotiation; the rules are well-defined and each participant has his/her own precise role and task. In the English auction, the seller tries to sell an item by starting at a relatively low price, and the buyers bid for the item. The price increases as the buyers outbid one another to win the item. This mechanism will result in maximum profit for the seller. The challenges in designing the e-negotiation server for this scenario include: validating users registration; facilitating fast-paced interaction format with one-to-many negotiation cardinality; assuring the sender's identity during the bidding process; recording the transaction history; and providing additional service such as proxy bidding or implementing an NSA for a more complex bidding strategy.

The Dutch auction is characterized by a seller who tries to sell an item at a relatively high price. The seller will gradually reduce the price until a buyer is willing to bid on the item. If the item quantity is only one, the auction ends, if not, the process continues until all items are sold. Procurement auction is a mechanism where a consumer asks a group of suppliers to provide an item at a low price. The consumer will gradually increase the price until one of the suppliers is willing to bid. The challenges in developing these types of auctions are relatively the same as those for the English auction.

Variations such as Vickrey auction or sealed bid auction may tune-up the auction scenario a bit, but the main concepts remain the same. We can understand more about the ENS's main purpose by observing the ENS's negotiation scenario. This might also lead us to design a negotiation media that is generic enough to facilitate various negotiation scenarios.

3.2.2 Availability of NSS or NSA

The reason for the negotiation media to provide a NSS or an NSA within its system is simple but crucial; to provide assistance to the user. From the software engineering perspective, if the negotiation media does not provide any form of NSS or NSA, it would require major changes to the software design or even a major redesign in order to offer these features.

As illustrated in Figure 18, negotiation media (the server) is usually tightly-coupled with the NSS because both share the same database, as presented in INSPIRE [20], and the NSS is usually tightly-coupled with the NSA because the NSA uses negotiators' preferences that reside within the NSS to evaluate them with the strategy, as presented in Aspire [21].

A good loosely-coupled ENS design should not share the same database among the negotiation media, NSS and NSA. Instead, it should provide a gateway as a common communication interface point for accessing the necessary information between components. This way the participants can easily choose whether or not to utilize the provided NSS or NSA, or even to access this gateway and build their own NSS or NSA based on their own preferences, ratings, strategies and tactics.

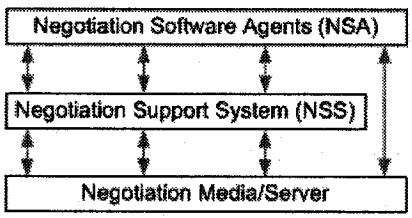

Figure 18: Negotiation agents, support system, and media

Architecting a system that can adapt to and facilitate added components is not an easy task, especially with tightly-coupled designs. Components such as NSS and NSA will become an integral part of the negotiation media and would certainly require interfaces, a shared ontology, and access to the system's common features.

3.2.3 The Quality Attributes

The study of software quality attributes by Barbaci et al (1995) [24] has brought insight in the desirable characteristics of a software application. These quality attributes of software usually end with "–ility" such as: reliability, scalability, response-ability, maintainability, adaptability, reusability, and so on. Reliability refers to the software capability of producing accurate, free-from-error results. Scalability is the software ability to run within a stressed environment such as multiple instances, clients, requests, and demands. Response-ability is the capability to process operations in an acceptable period of time. Maintainability,

adaptability and reusability are terms used for describing the easiness of maintaining the software after the development cycle, handling future requirements, and being part of a new system.

Although it is good to have all those quality attributes, during the developing period, software designers have to emphasize certain quality attributes and neglect others [25]. For example, software that is being designed for high security and tolerability will probably have its performance degraded in terms of speed or response-ability.

In Table 4 we will highlight several quality attributes found in the e-negotiation servers mentioned above. This will make it easier for us to organize and study the quality attributes that are desirable when designing e-negotiation media.

3.2.4 The Software Architecture

The software architecture of a program or computing system can be defined as "the structure(s) of the system, which comprise software components, the externally visible properties of those components, and the relationships among them" [26]. There are several common architectures for designing computer software, such as: client-server, distributed computing, peer-to-peer, blackboard, implicit invocation, pipes and filters, plug-in, three-tier model, component-based, model-driven, and service oriented architecture [27].

Most e-negotiation servers still inherit the client-server or the three-tier architecture. The main difference between them is the absence of a database on the client-server architecture. The three-tier architecture is composed of the client tier or the browser, the front-end tier or the application server, and the back-end tier or the database server. This architecture gives elegance and simplicity to the development of web-based applications. However, the front-end server is usually a collection of tightly-coupled modules, making the system difficult to adapt to future changes. It is also not very scalable for handling a large amount of requests because the front-end server is usually deployed on one machine, unless server mirroring and routing techniques are used.

Some work conducted on e-negotiation servers is still based on offline application systems. These applications usually serve as experimentation tools for the designers when they focus their work on NSA. The agents are tested through various competitions on top of

a platform to accommodate their interactions. The architecture for these systems can be categorized as component-based architecture.

3.2.5. Classification Summary

Earlier in this chapter, we reviewed the related research work on e-negotiation servers accomplished by the academic community. We have highlighted their main characteristics and design components. We introduced several sub categories, such as the negotiation scenarios, the presence of NSA/NSS, and the system's quality attributes and architecture, which can be used for the classification of e-negotiation servers. As shown in Table 4, the most widely used negotiation scenarios are the English auction and the two-party negotiation. We can conclude from Table 4 that the majority of existing e-negotiation servers are still without NSS or NSA, but those that provide NSS or NSA have these two components tightly-coupled into them.

The quality attributes that seem to be covered are the flexibility in defining negotiation attributes and rules, the expandability for future addition, and the reusability of the common functions. There is a tendency for the negotiation attributes to be assigned into a separate XML file which can be modified anytime during the negotiation (i.e., at run-time). The negotiation rules can be processed by using a separate expert system (such as JESS). This way, like the negotiation attributes the negotiation rules can also be modified at run-time. The most commonly adopted architecture for ENS is the three-tier architecture because of its simplicity, but the distributed system architecture provides an open standard interface to facilitate access from different locations and the possibility of future additions to the system.

System	Negotiation Scenario	NSA/NSS availability	Quality attributes	Architecture
GEE [7]	Multi negotiation game (ex. given: stock market)	None	Limited flexibility for game designing	Client-server, component-based
GNP [7]	Multi auctions (ex. given: English auction)	None	System configuration flexibility, reus-ability toolkits	Three-tier, component based
Multi agent platform for e-negotiations [10]	English auction	NSA	Flexibility in defining rules, reusable protocols, expandable for new NSA for analysis purpose	Blackboard Component based Non-online system*
ANTS [8]	Two-party negotiation with 1-to1, 1-to-many, and many-to-many	Integrated NSA, with flexi-bility to use NSA or not	Flexibility in negotiation attribute con-figuration	Component based, non-online system*
Attribute-based ne-gotiation process composer [5]	English auction	None	Flexibility in accessing the service by using web service, good expandability with open standard interface	Distributed system
INSPIRE [20]	Two-party negotiation with 1-to1	NSS	Flexibility in setting preferences, rat-ings etc.	Three tier, component based
Aspire [21]	Two-party negotiation with 1-to1	NSS + NSA	Flexibility in defining the rules	Three tier, component based
Invite [18]	Two-party negotiation with 1-to-1	None	Flexibility in adding new issue during negotiation phase	Three tier, component based
Negotiation platform for ARM [23]	English auction with Vickrey's for-mat	None	Flexibility in modeling visually	Client-server, component-based, model driven

* The assumptions are made based on the examples, code, or the screenshots provided

Table 4: E-Negotiation server comparisons

Chapter 4

Methodology

E-negotiation processes require a software architecture that encourages efficient iterative communications, the capability to reach a maximum number of potential customers and business partners, interoperability across platforms, quick and easy development, deployment, maintenance, and system expansion.

Our vision is to provide a flexible negotiation server which enables the negotiators to connect either using the provided components (such as user interface, NSS, or NSA) or using the ones that they develop on their own to suit their needs. This vision can be realized by designing a system that builds upon open standards, uses shared communication interfaces and ontology, and has a loosely coupled and extensible design. One of the methodologies that we consider feasible to deal with such requirements is service oriented architecture.

In this chapter, we introduce the concepts of SOA and its underlying components and building blocks. We will however concentrate of web service orchestration since it's the main methodology followed in this thesis.

4.1 Service Oriented Architecture (SOA)

A service oriented architecture (SOA) is "an evolution of the component-based architecture and distributed systems" [28]. Component-based architecture is an architecture where the functionality of the whole is divided into smaller functions, each encapsulated in a component. A distributed system is an extension of component-based architecture and refers to components that may exist in different physical locations. "The main advantage of a component-based architecture is that it facilitates reusability and repurposing of specific components and that it makes maintenance easier. Reusability and repurposing are often primary business drivers for adopting a SOA" [28].

Due to SOA's wide acceptance, there are many companies such as Amazon, Google, Yahoo!, eBay, and Cathay Pacific's, have expanded their business offering to in-

clude web service interfaces for their clients. Businesses adopt SOA because of several benefits, including: interoperability for various platforms and proprietary systems, easiness for future expansion by maximizing loose coupling, repurposing and reuse of existing components, easier maintenance, and reaching more potential business partners in an open and secure environment.

SOA is a software architecture that configures entities (services, registries, contracts and proxies) to maximize loose coupling and reuse [29]. It enables data to be passed across different platforms over the Internet in a structured text-based format such as XML. A component offers a service and idles until receiving an incoming service request. A service consumer can make a service request by passing the appropriate information and data structure. Once the process is achieved, the service provider will provide a service response based on the agreed data structure (as can be seen in Figure 19).

This methodology fits our purpose of building an e-negotiation server that can be accessed from any physical location. For example, in the English auction scenario, one of the buyers can send a bid along with his/her user id, password, and auction id, which can be illustrated as a service request sent to the e-negotiation server - Bid(userID, password, auctionID, bid). Subsequently, the e-negotiation server will respond "ACCEPTED" if the given user information is valid and the bid amount is higher than the highest current bid, or "NOT_ACCEPTED" if the user information is invalid or the bid amount is lower than the highest bid at that moment.

Figure 19: Service Oriented Architecture [28]

Depending on the scenario being executed, the service response method may take the form of unicast, multicast, broadcast, and anycast message (see Figure 20). Unicast (point-to-point) is a methodology where the service provider sends a message from a single source to a single destination. Multicast (point-to-multipoint) is a parallel communication pattern in which a source host sends a message to a group of destination hosts. Broadcast (point-to-all-points) is a methodology where the service provider sends a transmission to all

message consumers. Anycast (point-to-point-to-multipoint) is a methodology that assigns a private address to several message consumers on a fabric. The message sender does not know or care who consumes the message or the details of the message's distribution list.

The unicast service can be illustrated in the previous English auction bid process example. The multicast service can be incorporated when a new item is posted by the seller. The server will inform every potential buyer in the auction. The Broadcast is a way to send a message to every participant, such as the buyers and the seller. An example of broadcast is the message notification at the end of the auction which indicates whether it was successful or not. Anycast is a special case when the participants access the server, use their own NSS or NSA, and then forward the information to their own service. For example, the server sends a BidUpdate(), and one of the participants forwards this information to his/her own NSA.

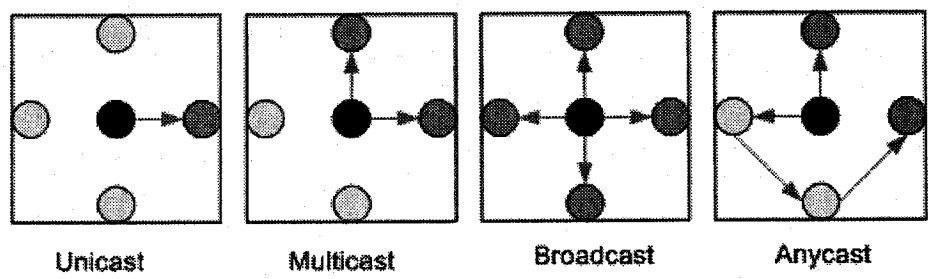

Figure 20: Unicast, multicast, broadcast, and anycast in SOA

4.2 Web Services

According to Kim and Segev (2005) [5] one of the main challenges in designing an e-negotiation server is the involvement of inter-organizational partnerships and systems, as well as the fact that the relationships between them are ad-hoc or dynamic. Web services resolve many problems regarding the interoperability of intra and inter-organizational applications. They also represent a component-based approach that is appropriate for designing an ad-hoc system.

According to the W3C (World Wide Web Consortium), a web service is a software system designed to support interoperable machine-to-machine interaction over a network. The exchanged messages between machines are conveyed using normal HTTP requests and

responses in XML based format. By posting the service on the Internet, and using XML-based messages, the information can be exchanged widely and across various platforms. This flexibility allows the information exchange between the e-negotiation server that, for instance, runs an English auction on top of a Windows system and one of the participants who wants to forward and process the information further in his/her own NSA on a Linux system.

Web service technology makes the exchange of business data much easier and can be used for A2A or B2B integration [30]. A2A integration refers to integration between different applications (e.g. Java and .NET) within a single organization; for this reason, it is also known as Enterprise Application Integration (EAI). B2B integration refers to business process integration involving business partners.

There are four components that make a web service work, namely: XML, SOAP, WSDL, and UDDI. They will be explained in the subsequent sub sections. The interactions between a web service client, SOAP, WSDL, and UDDI are illustrated in Figure 21.

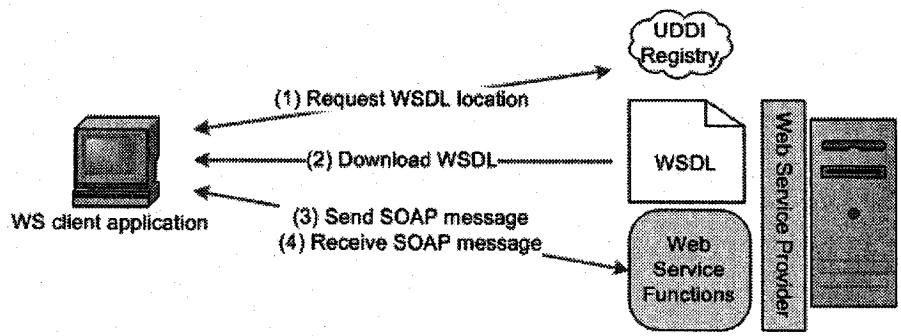

Figure 21: Web service interaction - modified [30]

4.2.1 XML and DTD

Extensible Markup Language (XML) is "the basic building block of web services" [30]. It can be used to organize business documents and data. XML files can be stored or transmitted between two applications on a network. From the file structure side, XML files are just plain text documents that contain special tags that label different parts of a document or field of data. XML allows designers to create their own customized tags that enable unlimited possibilities.

These unlimited possibilities need to be bounded with the agreed data structures between business partners by using a *DTD scheme*. The purpose of a DTD (Document Type Definition) is to define and validate the legal building blocks of an XML document, such as the valid element names, their nesting order, and their cardinality.

Another subject that is noteworthy about XML is the *XML namespace*. Because the elements in XML are not predefined, we frequently have the same element name in more than one XML document which has a different purpose. If those documents are added together it will cause a name conflict. XML namespace is a prefix mechanism to avoid this type of conflict. Together with a DTD file, XML namespace creates the agreed upon *ontology* for the information being transmitted between business partners.

Although XML is just a text-based document, the process of extracting the information can be tedious if done manually. This process is usually done with the help of a tool that can read and analyze an XML document also known as an *XML parser*, such as *SAX* (Simple API for XML) or *DOM* (Document Object Model).

4.2.2 SOAP

Web services depend on SOAP (Simple Object Access Protocol), which is a protocol for exchanging *XML-based messages* over a computer network, usually using HTTP. A SOAP document instance is commonly called a *SOAP message* or *envelope*.

A SOAP envelope encapsulates a set of header information and the body of the message (Figure 22). Although it is not an essential component, the header inside a SOAP envelope can define additional information such as the message-id, sender signature, encryption scheme, etc. The main idea of having a header section is to free the message body from any of this information. The body section contains the payload of the message being submitted to the counterpart system which is also referred to as *the endpoint*.

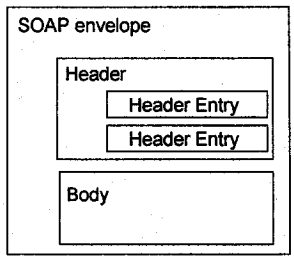

Figure 22: SOAP envelope[11]

Listing 1 represents a SOAP message captured by a TCP/IP monitoring tool such as *tcpmon*[12] during an online auction setting. In Listing 1 we can see that the header contains some information about the sender, and the body contains a message to bid for an auction. The element "<Bid>...</Bid>" is actually a function that was implemented as a service in the endpoint system. This remote procedure call details must conform to the specification specified in the WSDL, otherwise the communication will fail or not receive a response.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <soapenv:Envelope
3.  xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
4.  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5.  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
6.  xmlns:h="http://www.wrox.com/Header">
7.      <soap:Header>
8.          <h:from >pringadi@site.uottawa.ca</h:from>
9.          <h:member >yes</h:member>
10.     </soap:Header>
11.     <soapenv:Body>
12.         <Bid xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
13.         process/" >
14.             <userID>Pringadi01</userID>
15.             <password>71bc5ddb2b965ee666ae5cb0faddf888</password>
16.             <auctionID>BIN10072</auctionID>
17.             <value>1200</value>
18.         </Bid>
19.     </soapenv:Body>
20. </soapenv:Envelope>
```

Listing 1: SOAP request message

[11] http://www.vbip.com/books/1861005091/chapter_5091_02.asp

[12] http://ws.apache.org/axis/java/developers-guide.html

4.2.3 WSDL

Before using SOAP with a certain web service, we need to know in advance the structure required for the SOAP message to call the service at the endpoint, such as the function that needs to be called and the variables that need to be passed. In order to get this information, first we need to retrieve the WSDL (Web Services Description Language) file. WSDL is a document that describes the network services as a set of endpoints for operation on messages. WSDL specifies the exact message format, Internet protocol and address that clients need in order to communicate with a particular web service [30].

In a real implementation, WSDL files across different development frameworks such as Axis, .NET and BPEL can be very long and slightly different from one another, but the main elements are still bounded by the standard from W3C, WS-I, and OASIS[13].

A WSDL file is automatically generated by a helper tool such as J2EE JAX-RPC, IBM WebSphere, Microsoft .NET and Apache Axis[14]. These tools can also analyze the existing WSDL files and derive their stubs and endpoint interfaces to simplify the exchange of SOAP messages (see Figure 23). .

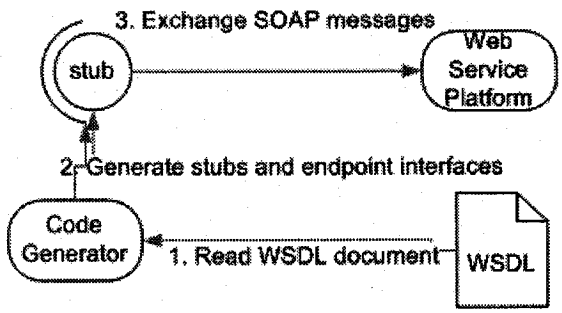

Figure 23: A WSDL code generator [30]

As can be seen in Listing 2, there are several key elements in a WSDL file such as `types`, `message`, `portType`, and `partnerLinkType` (`partnerLinkType` only be-

---

[13] WSDL formal specifications can be found at: www. w3.org/TR/wsdl20 (W3C), www.ws-i.org /Profiles/BasicProfile-1.1-errata.html (WS-I), and www.oasis-open.org/cover/wsdl20000929.html (OASIS)

[14] The online information regarding these technologies can be found at: java.sun.com /webservices/jaxrpc (JAX-RPC), www.ibm.com/websphere (WebSphere), www.microsoft.com/net (.NET), and ws.apache.org/ axis (AXIS)

longs to BPEL's WSDL). The `types` element describes the variable data types that need to be passed in order to access the service, for example: name as string, quantity as integer, and so on. The `type` can also be declared as a `complexType` to declare a data structure, such as customer's name as string, address as string, and phone as string and perhaps rating as an integer. The `message` elements usually bind the incoming request and outgoing response messages with the data types. The `portType` elements describe the incoming port and outgoing port along with their operations (functions) and which messages are involved. The `partnerLinkType` describes the role of a service as either a provider of the service or as a requester.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <definitions name="NewBPEL">
3.     <types>
4.                 <element </element>
5.                 <element</element>
6.     </types>
7.     <message> </message>
8.     <message> </message>
9.     <portType> </portType>
10.    <portType> </portType>
11.    <plnk:partnerLinkType>
12.          <plnk:role name="NewBPELProvider"> </plnk:role>
13.          <plnk:role name="NewBPELRequester"> </plnk:role>
14.    </plnk:partnerLinkType>
15. </definitions>
```

Listing 2: BPEL's WSDL file basic structure

4.2.4 UDDI

The last component of a web service is the Universal Description, Discovery, and Integration (UDDI). UDDI is an information repository for entities to find available web services. It is an optional component of web services according to the Basic Profile[15].

Although it is considered optional, there are several benefits to using UDDI, including: discovery of the correct business service, defining a way of using the web service, reaching potential business partners and customers, and therefore, extending the market reach, and describing services and business processes in an open and secure environment.

[15] www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

UDDI can also be found in a free public space such as in www.xmethods.net. The benefit of publishing a web service in a public facility is the high number of responses from the internet community. In an online auction setting like the one in this thesis, a UDDI registry will be best implemented in the private space because the e-negotiation server is assumed to deal with several registered known partners. In our simulation we do not use UDDI, instead we have several web pages that describe where to fetch the WSDL file, how to use the available functions (web services), and what response is expected from the e-negotiation server.

4.3 Web Service Orchestration

Web service orchestration is a way to compose and coordinate web services together to obtain higher-level business processes. It describes how web services can interact with each other at the message level and tracks the sequence of the messages including the business logic and execution order of the interactions [31]. Web service orchestration enables the developer to focus on the design of the business process rather than be trapped in the system's complexity.

A similar terminology that is usually associated with web service orchestration is *web service choreography*. Web service choreography deals with public message exchanges that occur between multiple web services rather than dealing with a specific business process that is executed by a single party. The important distinction between Web services orchestration and choreography is that orchestration refers to an executable business process that may interact with both internal and external web services viewed from the perspective of one of the business parties; choreography is more collaborative in nature, wherein each party involved in the process describes the part they play in the interaction. However according to Peltz (2003), recent enhancements and convergence of standards have somewhat blurred this distinction [31].

There are standards available to orchestrate web services such as BPEL, a join project of IBM, Microsoft, and BEA. BPEL is designed to reduce the complexity of orchestrating web services, thereby reducing time-to-market and costs and increasing the overall efficiency and accuracy of business processes [31].

BPEL stands as a layer on top of WSDL. While WSDL describes the messages' data types, port types, allowed operations, and partner roles. BPEL describes partner bindings, incoming and outgoing variables, and a set of operation logic sequences. Like other programming languages, BPEL also supports common repetition (while-loop), selection (if-then-else, select-case), error handling (try-catch), parallel processing, and Java embedding.

As one of the most adopted web service orchestration languages, BPEL provides the necessary business logic to compose a complete running process. After an evaluation of its capacity, we decided to use BPEL as a tool to implement our concept of an SOA-based e-negotiation server where negotiation processes are modeled as BPEL processes.

4.4 BPEL

BPEL4WS is an acronym for Business Process Execution Language for Web Service, which is often referred to as BPEL. Ever since the specification was submitted to OASIS in 2003, BPEL has gained the support of nearly every major industry vendor. While WSDL describes what service that can be called and the parameter structures that must be passed, BPEL describes the business logic, and the `invoke` and `receive` operations perform on other web services.

Just like any ordinary programming languages, BPEL has decision-making logic, conditional repetition, mathematical and logic operations, and fault handling; but unlike other programming languages, BPEL is an XML-based document. BPEL depends heavily on the use of XPath[16]. XPath is a language to navigate and address information through an XML document, for this reason it is also refered to as XPath query language. For example, to get a bid value form a buyer's SOAP message the syntax will be written as `"/buyer:Bid/buyer:value"`. Table 5 illustrates the differences between common programming syntax and BPEL syntax.

[16] http://www.w3.org/TR/xpath

Common programming syntax	BPEL syntax

If (condition) {
  statement
} else{
  statement
};
	<switch name="Switch_1">
	<case condition="(variable, part, *XPath-query)=something*">
	<assign name="Assign_1">
	<copy>
	<from (variable, part, *XPath-query)*/>
	<to (variable, part, *XPath-query)*/>
	</copy>
	</assign>
	</case>
	<otherwise>
	<assign name="Assign_2">
	<copy>
	<from (variable, part, *XPath-query)*/>
	<to (variable, part, *XPath-query)*/>
	</copy>
	</assign>
	</otherwise>
	</switch>

While (*condition*){
  statement
}
	<while name="While_1" condition= (variable, part, *XPath-query)*>
	<assign name="Assign_1">
	<copy>
	<from (variable, part, *XPath-query)*/>
	<to (variable, part, *XPath-query)*/>
	</copy>
	</assign>
	</while>

Table 5: BPEL syntax comparison

In BPEL, there are several process activities that are more commonly referred to as *BPEL components*, for example: `invoke`, `receive`, `assign`, `empty`, `while`, `switch`, `pick`, `flow`, `wait`, `user task`, `partner link`, and `java embedding`. The purpose of these components can be summarized in Table 6.

Component	Function
Invoke	Invoke other web services.
Receive	Receive data in response to invoke in asynchronous mode.
Assign	Assign values to the variables.
Empty	Idle the process (do nothing), and continue to the next sequence
While	Loop until condition is not valid anymore
Scope	A grouping mechanism which allows for a collapsible sub-region of the overall activity
Switch	Selection the logical branch (if-then-else).
Pick	Selection based on the event that comes first.
Flow	Execute more than one activity at the same time.
Wait	Wait until a certain time or wait for a period of time, then execute an activity. E.g.: wait until 2:00AM or wait for two minutes.
User task	Use façade object to ask and wait for a user input.
Partner Link	Define a new partner (another web service) to deal with
Java Embedding	Embed java which usually deals with database processing or XML parsing.

Table 6: BPEL components

Although the standard is fully defined as of July 2002[17], there are still some limitations regarding the expressive power of BPEL. According to Wohed et al. (2002) [32], BPEL supports 11 out of 18 commonly required workflow patterns. However, its expressive power is richer than any other web services-based process standard [33, 34]. Since BPEL is a new standard, its *Integrated Development Environment (IDE)* is still under development. However, most BPEL IDEs provide graphical modeling to avoid the tedious task of reading and writing XML manually.

[17] http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

Chapter 5

Analysis and Design

5.1 System Requirements

We plan to implement our e-negotiation server by using a web service orchestration language such as BPEL. There are several providers of integrated development environments (IDE) for BPEL, such as IBM, Oracle, CapeClear, and NetBeans. We decided to use the Oracle BPEL Process Manager v10.1[18] because it is a freeware; it has a good user interface and comes with excellent support from Oracle Corp. and public forums. This software previously belonged to Collaxa, one of the earliest providers of IDE for BPEL. Our software requirements for this thesis implementation can be summarized in Table 7.

Software	Version	Description
Oracle BPEL Process Manager	10.1	BPEL engine, where BPEL codes are compiled and executed.
JDeveloper 10g[19]	10.1	BPEL process modeling environment.
Oracle Application Server	10.1	Web server where all the web service produced by BPEL is hosted.
Apache Tomcat	5.5	Web server for hosting an NSA for the automated clients.
Axis	1.3	To be used with Tomcat to build an NSA web service and to validate the interoperability across different platforms.
MySQL	5.0	DBMS server to manage users and auctions information.
Java Expert System Shell (JESS)	6.1.8	A rule engine to parse the NSA's strategy and tactics.
Internet explorer or any web browser	6.0	To display the result from the system and to facilitate users inputs..

Table 7: Software requirements

[18] www.oracle.com/technology/bpel

[19] www.oracle.com/technology/products/jdev

In terms of hardware, one of the most demanding requirements of this project is the system's RAM. The Oracle BPEL Process Manager alone can consume around 450 Mbytes of RAM during the modeling process. Based on our estimations, to run efficiently, the system should have at least 1GByte of RAM and a processor that performs at least as fast as Pentium 4 2.8 GHz, with an output screen of 1024x768 pixels or larger for comfortable modeling.

5.2 Server Design

The server where the negotiation process is created and executed can be implemented either on one of the participants' side or by a third party playing the role of a mediator or facilitator. Deploying the server on one of the participant's side is much simpler, while deploying it on the side of a third party provides for a more neutral negotiation, as well as common interfaces and ontology to be shared by participants. A third party solution will also ensure an optimal and efficient negotiation process. For those reasons, we prefer to deploy our e-negotiation server in the third party side (see Figure 24).

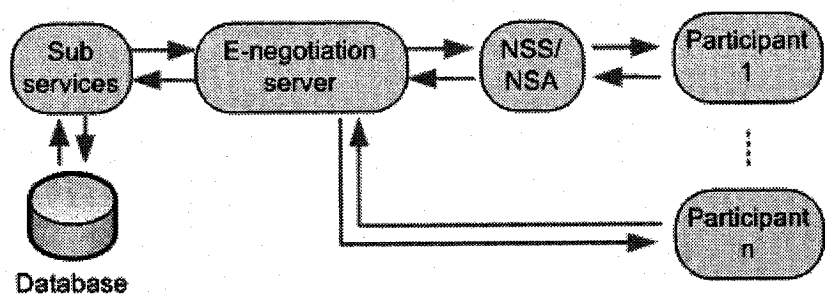

Figure 24: SOA-based e-negotiation server

We designed a system that is generic enough to run various types of auctions. In 1998, Kumar and Feldman [3] identified five common phases during auctions: initial buyer/seller registration; setting up a particular auction event such as posting; bidding process; evaluation of bids and closing the auction; and trade settlement. Each of these phases can be implemented as services, such as registration and authentication services, posting services, bid services which validate incoming bids, information retrieval services, history log services for audit trailing, and gateway services which coordinate all the services.

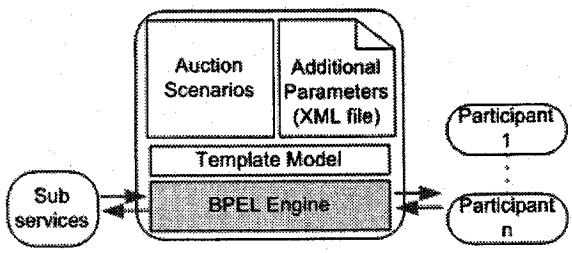

Figure 25: Internal design of the e-negotiation server

We designed the server and the participants (sellers or the buyers) as web services. To orchestrate these web services we use the BPEL Process Manager. We visually model the auction scenarios using JDeveloper. The auction scenarios are then compiled and deployed on the BPEL engine (BPEL Process Manager Server).

To avoid eventual recompilation and redeployment due to minor changes, we transfer additional parameters (bid visibility, idle time, auction deadline, etc.) out from the business logic of the negotiation process into an external XML file (Figure 25). The template model governs the format of the protocol and the XML files. The negotiation protocol is realized in the BPEL process flow, which will be discussed in subsection 5.2, and the XML configuration files, in the cases of the English and Dutch auctions, are shown in Table 8.

File Name	Listing
English.xml	<Auction scenario="EnglishAuction1"> <bidVisibility>TRUE</bidVisibility> <idleTime>5</idleTime> <deadline>15</deadline> <highestBidderVisibility>PUBLIC</highestBidderVisibility> <useReservePrice>TRUE</useReservePrice> <Auction>
Dutch.xml	<Auction scenario="DutchAuction"> <quantityVisibility>FALSE</quantityVisibility > <deadline>20</deadline> <multipleItem>TRUE<multipleItem> </Auction>

Table 8: XML configuration file

The BPEL engine is the core component of the negotiation platform where the execution of communication level operations with sub services and participants takes place. The participants access any web service offered by the e-negotiation server through the BPEL engine.

5.3 Modeling the Negotiation

Kumar and Feldman [3] proposed templates for modeling various auction processes using Finite State Machine (FSM) diagrams. By omitting details such as authentication from the process, the FSMs concentrate only on the interactions between *states* (event positions) and *participants*. In 2005, Rinderle and Benyoucef [35] introduced a way to map the states in these FSM diagrams into a set of BPEL components in a general manner. Together with Kumar and Feldman's FSM, their approaches have become our general guideline in modeling the auction processes in our system.

5.3.1 Direct Sales Modeling

The states of the FSM are represented with DT (Deal Template), Offer, D (Deal), and NA (Negotiation Aborted). While the incoming messages are illustrated with <Sender, Action>, the outgoing messages are illustrated with <<Recepient, Action>>. A direct sale process is illustrated in Figure 26. Direct sale is a special case of a negotiation protocol also referred to as *take-it-or-leave-it* protocol. The starting point for the process is the Deal Template. Once the seller sends OfferToSell, the state shifts to Offer. From here, if a buyer accepts the offer, there will be a deal. There will not be a deal if the seller withdraws the offer.

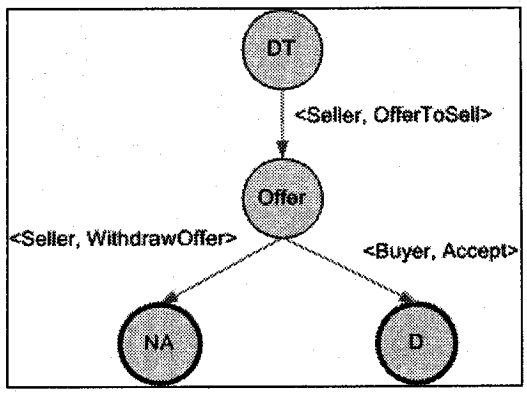

Figure 26: FSM for direct buy [3]

We visually modeled the services' logic and orchestrated the relationships between them in the JDeveloper modeling environment. This visual modeling approach makes the whole e-negotiation system easier to design for the software developer and more transparent for business analysts.

We describe our models using *BPEL process flow diagrams*, which are similar to *UML activity diagrams*. In addition to the fact that they are considered more organized and systematic, UML activity diagrams also capture more details of the process. A UML activity diagram divides the whole process into a set of procedural logic, business process and workflow [36]. It always starts with an initial node and ends with a final node; between these nodes, lays the process's activity in chronological order.

In our design, there are three participants who are involved in the negotiation process: the marketplace (the server), the seller, and the buyer. All three are implemented as web services. The main process focus is on the server. Every call to other web services (seller/buyer) is conveyed using the `invoke` component and the result is captured using the `receive` component (Figure 27).

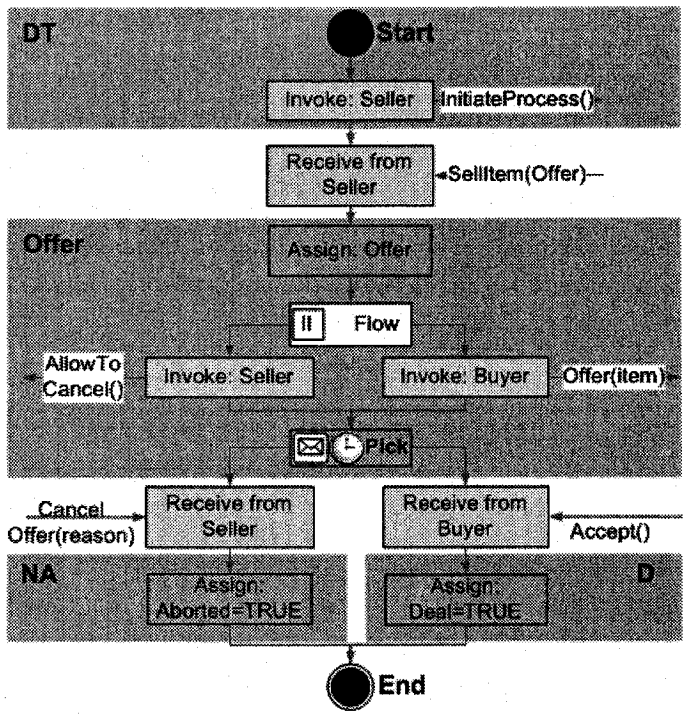

Figure 27: BPEL process flow diagram for the direct sale protocol

In Figure 27, the `flow` executes several activities simultaneously. However, the process will not continue if one of these activities is not completed. The `pick` construct is almost identical to the `flow` construct except that any message/event that arrives first will be executed while the other(s) will be abandoned. In Figure 27 there is a `pick` between re-

ceiving an input from the seller and the buyer. If the seller cancels the offer first, the negotiation will be aborted; if the buyer accepts the offer first, there is a deal.

5.3.2 English Auction Modeling

Figure 28 shows the FSM for the English auction. Almost the same as the FSM for direct sale, English auction modeling starts with an OfferToSell from the seller. The buyers can then bid to modify the offer. In an open cry English auction, this event will produce an outgoing message from the server to notify all the buyers that the best bid has changed; in a sealed-bid auction the best bid remains anonymous. In the Offer state, there are two other events that could occur. The seller can close the auction earlier, which will result in the Negotiation Aborted state, or the auction can end smoothly according to the pre-agreed time length (Figure 28).

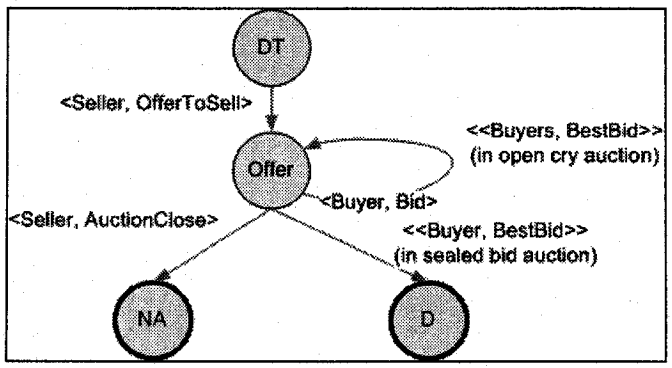

Figure 28: FSM for open cry English and sealed bid auction [3]

Due to the detailed process descriptions and the large size of the output being produced by the actual model developed in JDeveloper, we are not able to display the original BPEL model of the English auction without sacrificing its details and image sharpness. Screenshots of the original BPEL process can be seen in Appendix B. However, the following activity diagram presented in Figure 29 provides the same major functionalities as the real model.

For the sake of clarity, we present the English auction BPEL diagram using the `scope-and-expand` method. Each `scope` is actually a collection of several activities represented by a plus sign "(+)", and can be *expanded* into a subsequent activity diagram.

We assume that the participants have completed the registration process before the start of the auction. The registration process does not appear in the diagram.

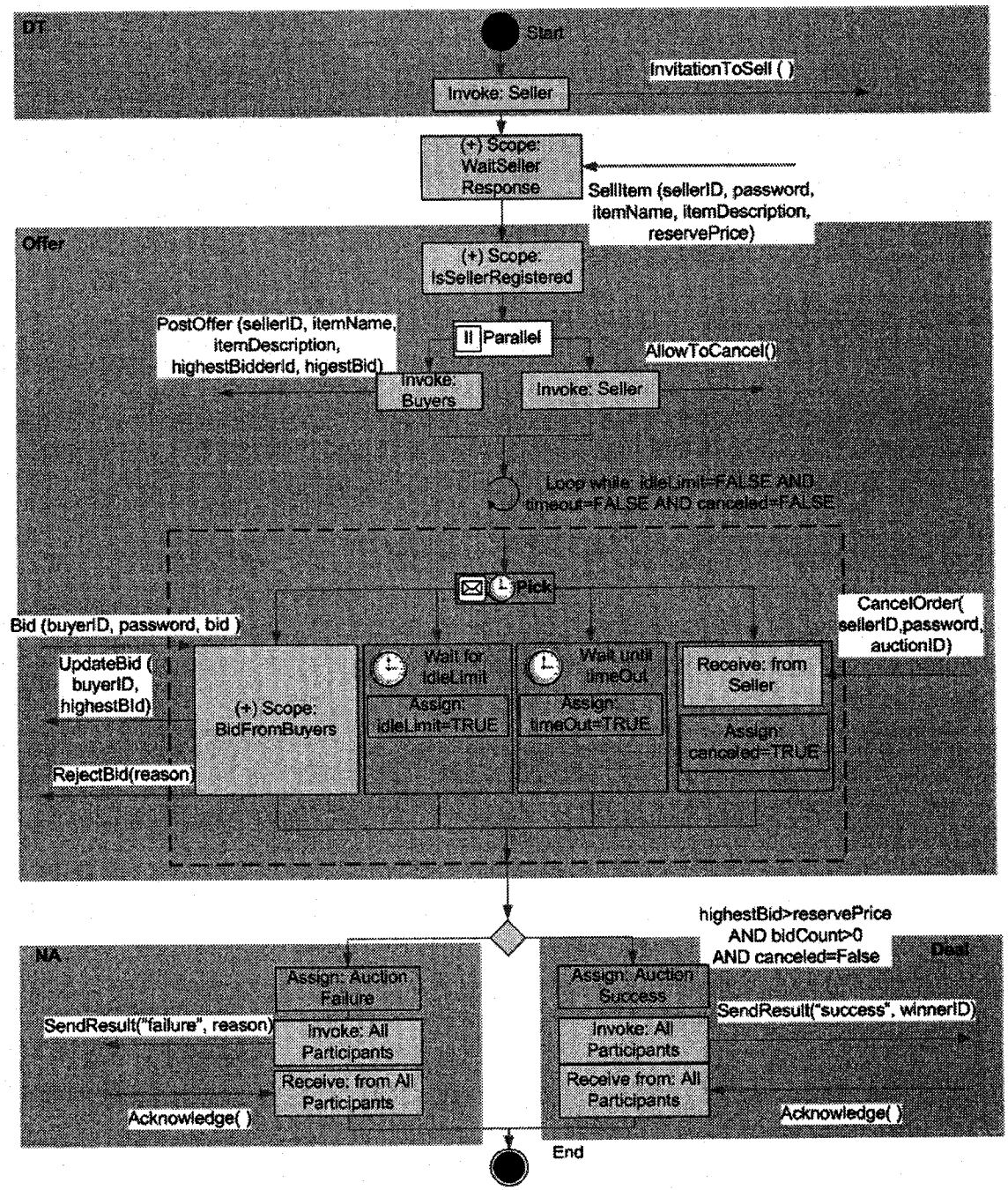

Figure 29: BPEL diagram for the English auction

The "Scope: WaitSellerResponse" is a collection of activities that wait for a response from a seller for a certain amount of time. This mechanism is a good way to ensure

that the seller has enough time to respond but does not ignore the invitation. If we expand this `scope`, it will look like Figure 30 - a.

After receiving an *item to sell* message from the seller, the server will check whether the seller is registered or not. If the seller is not registered, the process doesn't continue (Figure 30 – b). The server then executes a `flow` to initiate the bidding phase by invoking the seller so he/she can cancel the auction if he/she wishes to do so, and by posting the item to the buyers so that they can start the bidding process.

The bidding phase is a loop mechanism. According to Kumar and Feldman's FSM, activities that might happen during the bidding phase are: the *buyers bid* (back to loop); the *seller cancels* the auction (out from loop); and the auction reaches its *timeout/deadline* (out from loop). We introduced another variable into this phase: *idle time*. If there is no incoming bid after a certain amount of time, the bidding phase is over (out from loop). These four activities are utilized using a `pick`. If a buyer bids, the server then checks for his/her account authenticity and verifies the bid validity (Figure 30 – c).

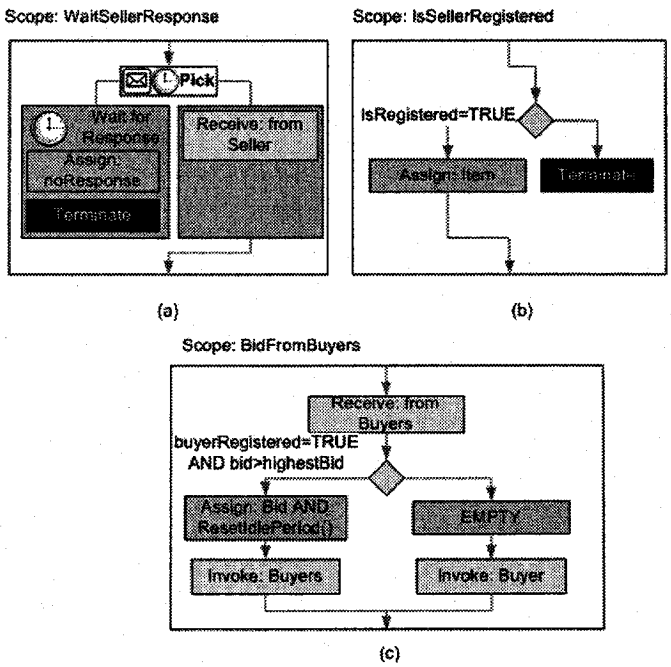

Figure 30: English auction subsidiary activity diagram

At the end of the bidding phase, a `switch` will decide whether the auction is successful or not. It is successful if: the *highestBid* is more than the *reservePrice;* there is more

than one *incomingBid*; and the seller does not cancel the auction. An outgoing message will follow to inform all participants of the result of the auction.

5.3.3 Dutch Auction Modeling

The FSM for a single item Dutch auction is represented in Figure 31. After the seller starts with an OfferToSell with a high enough price that typically no buyer is willing to pay, the process will shift to the Offer state. From there, the seller gradually decreases his/her price by sending NewOffer; one of the buyers can bid by accepting the offer; or the seller can close the auction earlier by sending an AuctionClose message (Figure 31).

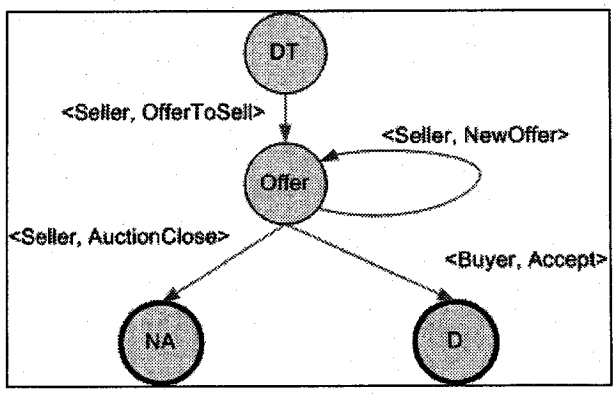

Figure 31: FSM for a single item Dutch auction [3]

Unlike the FSM shown above, we enabled our Dutch auction to deal with multiple items. If this modification is to be reflected in the FSM above, there will be an arrow pointing from the Deal state (D) back to the Offer state with a message <Server, QuantityUpdate>. This process will continue as long as items remain in the inventory. Our BPEL process flow is illustrated in Figure 33.

The process flow differences between the Dutch auction and the English auction are not too pronounced. Instead of submitting the reserve price during the selling process, the seller has to submit the unit quantity and the unit price (price of one item). After validating the seller's identity (Figure 32 - b), the sequence goes to the parallel `flow`. Three `invoke` constructs are executed at this event: to post the offer to the buyers so they can bid, to allow the seller to decrease the unit price, and to allow the seller to cancel or close the auction

early. Note that waiting for a response from the seller is similar to the one in English auction (see Figure 32 - a)

After this event, the process sequence will continue to a *while* loop and it will keep repeating until either a seller submits a "close auction" command, the bidding phase reaches its deadline, or the item quantity reaches zero, which means that there is no item to bid for.

The `pick` inside the `while` loop waits for any of the following events: a bid from a buyer, a timeout/deadline notification, or a "price update" or a "close auction" command from the seller. If a buyer bids, the server checks his/her identification and makes sure the bid should be less than or equal to the remaining item quantity (Figure 32 - c).

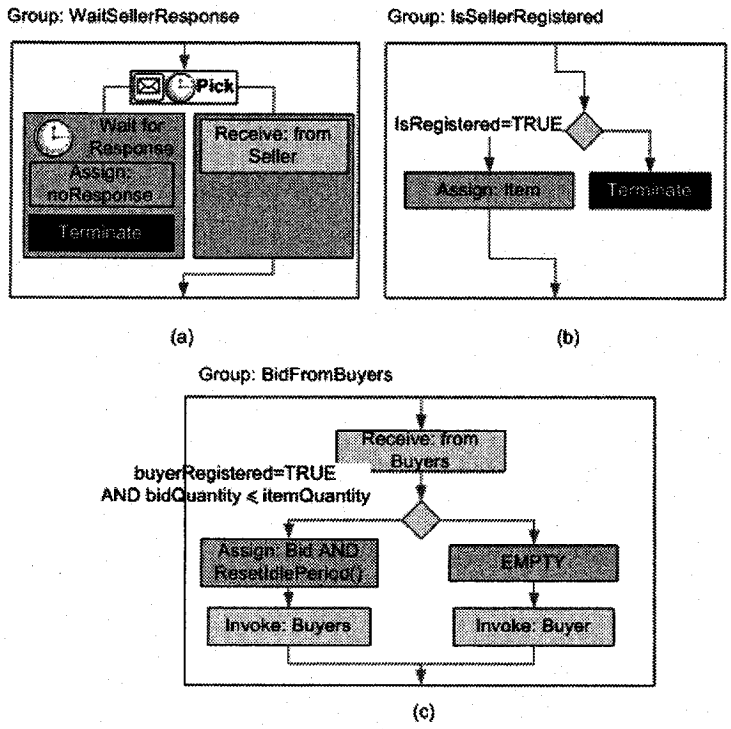

Figure 32: Dutch auction subsidiary activity diagram

At the end of the auction, the server will `invoke` participants to notify them with the result. Figure 33 shows the whole BPEL process flow diagram for a Dutch auction with multiple items.

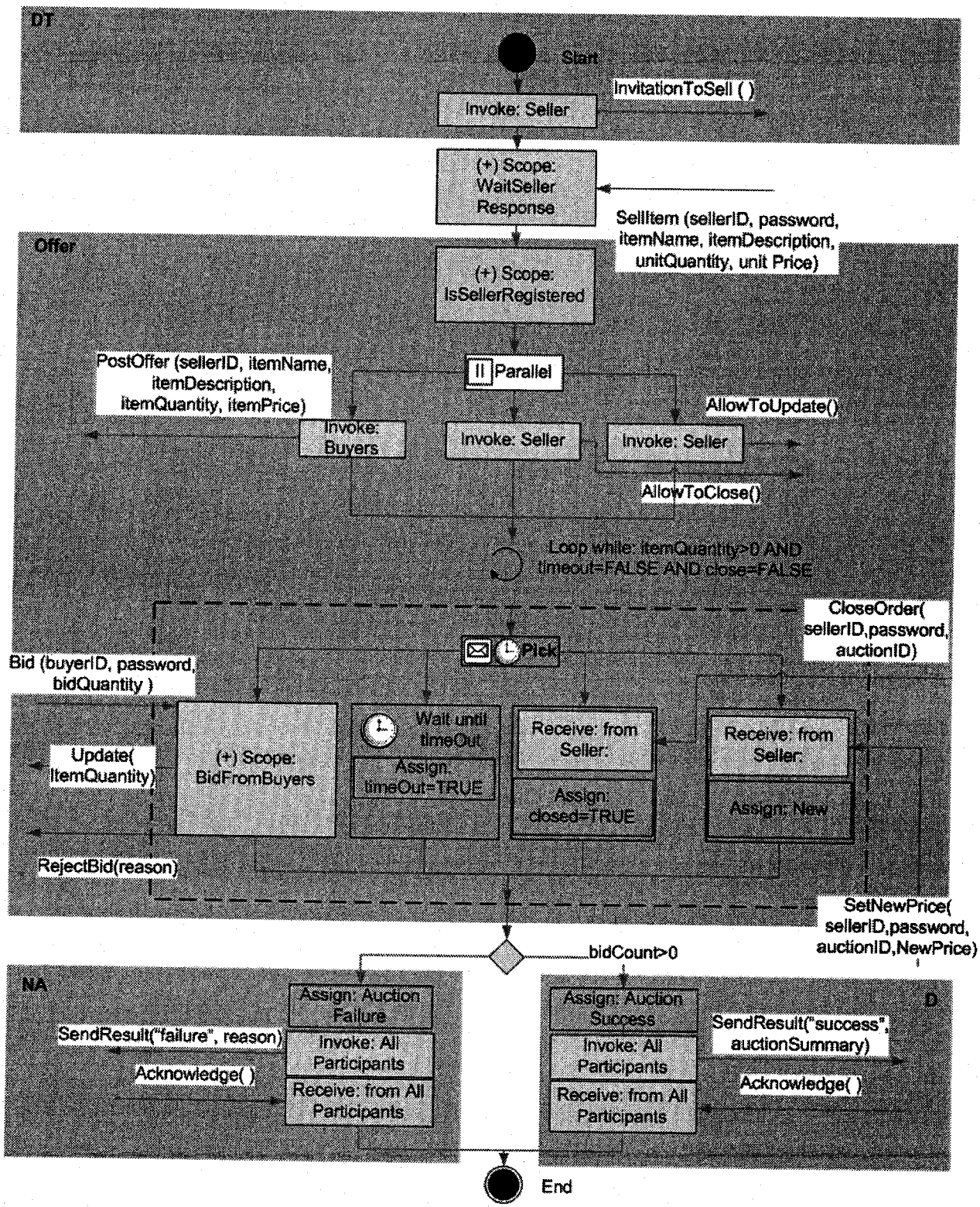

Figure 33: BPEL process flow diagram for Dutch auctions

5.3.4 Two-Party Negotiation Modeling

A two-party negotiation can be represented in the following FSM (Figure 34). This FSM is actually a hybrid of the direct sale and auction FSMs mentioned before. Similar to previous cases, the seller initiates the negotiation with an OfferToSell. At this point, the process will arrive to a Negotiable Deal (ND) which can be modified either by the buyer (Bid) or by the seller (AskPrice). The buyer and the seller can exit from this loop by submitting their FinalOffer. Each party can decide whether or not to accept the FinalOffer. If one is willing to accept the FinalOffer then there will be a deal (D); otherwise the negotiation is unsuccessful (NA).

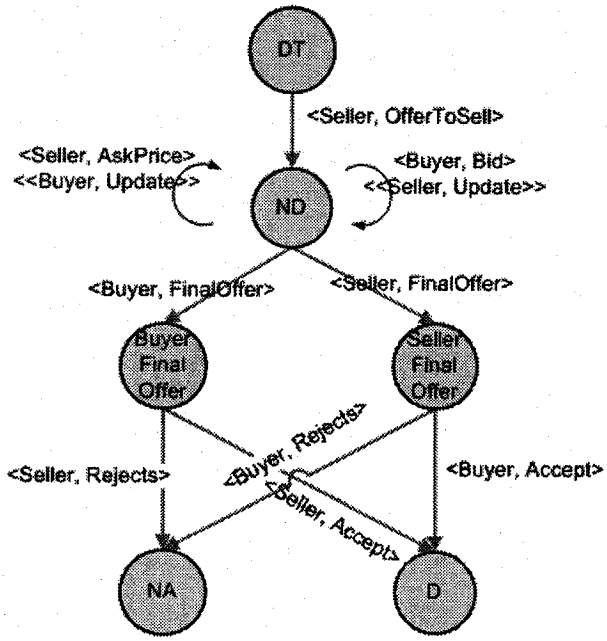

Figure 34: FSM for two-party negotiation [3]

Assuming the negotiation cardinality for this type of negotiation is one-to-one and that the negotiation issue is the price; by disregarding the authentication phase, the BPEL process flow for the two-party negotiation can be illustrated in Figure 35. The particularity of this process flow is that the server issues four `invokes` at the same time, allowing four `receives` from the participants' side. After the server `invokes` the participants, the sequence will enter a loop which stops if sellerFinalize or buyerFinalize equals TRUE. Buyer and seller can adjust the NegotiableDeal by sending Bids and AskPrices. After the se-

quence exits from the `while` loop, the server will `invoke` the party that receives the final offer from his/her negotiation partner. If this party decides to answer NO, then there is no deal; otherwise, there is a deal (Figure 35).

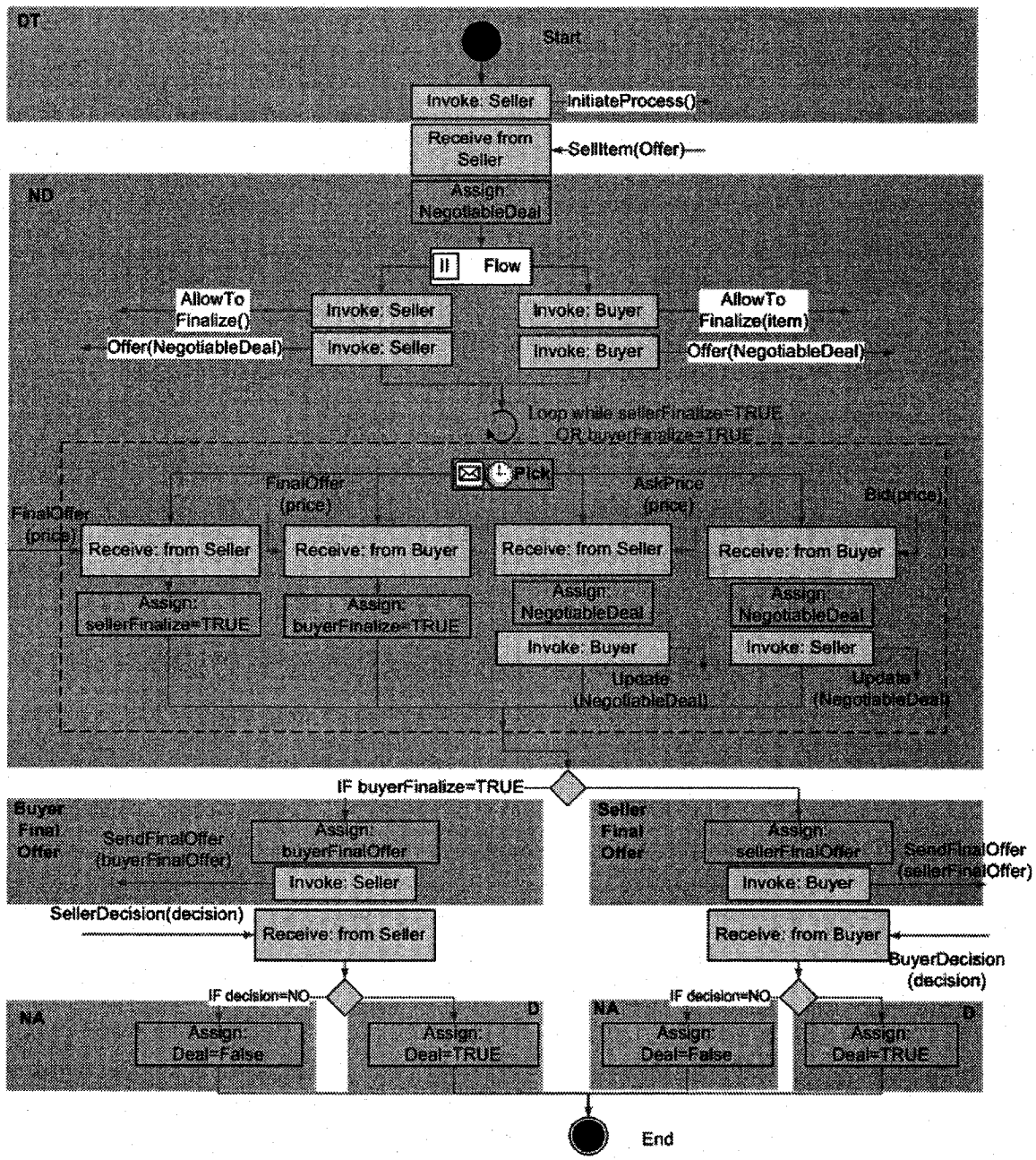

Figure 35: BPEL process flow diagram for a two-party negotiation

5.3.5 Evaluation of Modeling

Our acquired experience in modeling negotiation process and our close observation on the resulting BPEL process flow diagrams, led us to the following points:

- Kumar and Feldman's FSMs can be represented using BPEL process flow diagrams.
- The FSM diagrams from Kumar and Feldman as well as our BPEL process flow diagrams describe only the most common components of negotiations. In real-life situations the complexity of the negotiation process may vary, but the basic concepts remain the same. BPEL process flow complexity increases along with the sub processes that need to be handled, such as: authentication, validation of participant inputs, and the additional negotiation parameters (bid visibility, item quantity visibility, etc.)
- Web service orchestration is about organizing a set of web services to work together under the perspective of one of them. This particular web service should have the longest lifespan among the other web services so that it can control them. We decided to host this controller web service as a marketplace server. The process flows in the marketplace are executed chronologically according to the order of appearance in the BPEL process flow diagram.
- Each `invoke` to the participant (buyer or seller) is connected with `partner link` components. This component will then open a web service port and bind this port to the targeted partner. Each participant or partner must be registered before the auction process start. This registration phase is not discussed in this work.

We also found several similarities in the process flow that can be utilized as a collection of reusable objects. They are detailed below

- The Deal Template (DT) state can consist of specific negotiation protocols which are agreed upon before the process starts or at least it is assumed that all participants understand this protocol. Since BPEL stands on top of WSDL, it is easy for us to manage the message interactions. Every incoming or outgoing message format has to conform to the WSDL description. Deal template is always marked with a void invitation or initiation from the server. The seller then has to reply with an offer.

- The Offer state might vary slightly between negotiation processes. However, there is a unique sequence which represents this state in all BPEL process flow diagrams seen earlier. The sequences are: `Assign` which sets the offer from the seller; `flow` which `invokes` and allows participants to reply or send a message to the server; and `pick` which determines which events arrive first. For auctions, before `pick` there is a `while` which creates a repetitive cycle for the bidding phase.
- Mechanisms in `scope-and-expand` activity (such as "WaitSellerResponse", "Is-SellerRegistered", etc.) can act as reusable components for different types of negotiations. At this moment, BPEL does not support any process modularization or component reusability; therefore, there are no BPEL fragments that can be invoked from within the same or from different BPEL processes [37]. IBM and SAP are still working on a sub-process extension for BPEL (BPEL-SPE) [38].
- In the meantime, there are several possible solutions that we can adopt to address this problem. First, we can implement the reusable components as independent web services. For example, the main process calls a reusable web service such as Check-IsSellerRegistered. This web service answers with TRUE or FALSE. However, due to the nature of web services, this approach exposes the sub-components of the system, thus creating an integrity concern. The caller's IP address needs to be filtered to ensure security. Second, we can implement the reusable components as java class files. In the main BPEL process flow diagram, we can define a `java embedding` component which will replace the `scope-and-expand`, to call for this java class. This approach would defeat the purpose of visual modeling, therefore we prefer the first approach.
- Unlike BPEL-SPE (mentioned above), the two approaches do not have a synchronous connection between the main process and the sub processes. For example, if the sub processes are terminated abruptly, the main process cannot be informed; therefore, it may stall or produce errors.
- In one-to-many or many-to-many negotiations, it is best to set a broadcast message to all participants so that they will know the result of the negotiation. This message broadcast can be implemented as a reusable component.

5.4 Client Design

We consider two types of negotiators (also referred to as participants): one who uses the provided interface (***manual negotiation client***), and one who builds his/her own system and connects it to the e-negotiation server for automation purposes and for eventually employing AI techniques such reasoning about and learning negotiation strategies (***automated negotiation client***). Both clients have the same access point which is the BPEL engine's web service port. Automated negotiation clients can communicate with the negotiation server and join the negotiation process by accessing the BPEL engine and using the WSDL file. They can call the services offered by the e-negotiation server using its operation name and passing the necessary variables. The gateway service will ensure that all clients have valid identification and proper access to every phase of the negotiation process.

It is assumed that the automated negotiation clients are used by software developers or other advanced users who are knowledgeable in web service technologies and concepts. We provide the service discovery in static web pages. These web pages describe the location of the WSDL file and explain how to utilize the web service. The automated clients can process whatever information they get from the e-negotiation server and pass it to their NSS or NSA using a SOAP message tunneling mechanism.

Figure 36 shows two types of clients, manual and automated. Every interaction between the e-negotiation server and the clients is executed through web service ports and operations. Similar to the manual client, the automated client also connect to the e-negotiation server using the same web service ports and operations interface; however, the automated client passes the information from the e-negotiation server to the NSA. The NSA assists the automated client to make decisions. In our project the NSA is implemented as an independent system that offers various web services that deal with negotiation strategies. We took this approach in order to increase the reusability of this NSA. The server's process model in Figure 36 is the BPEL process flow diagram explained earlier. The `java embedding` components access specific negotiation attributes (in an XML file) and use the MySQL database.

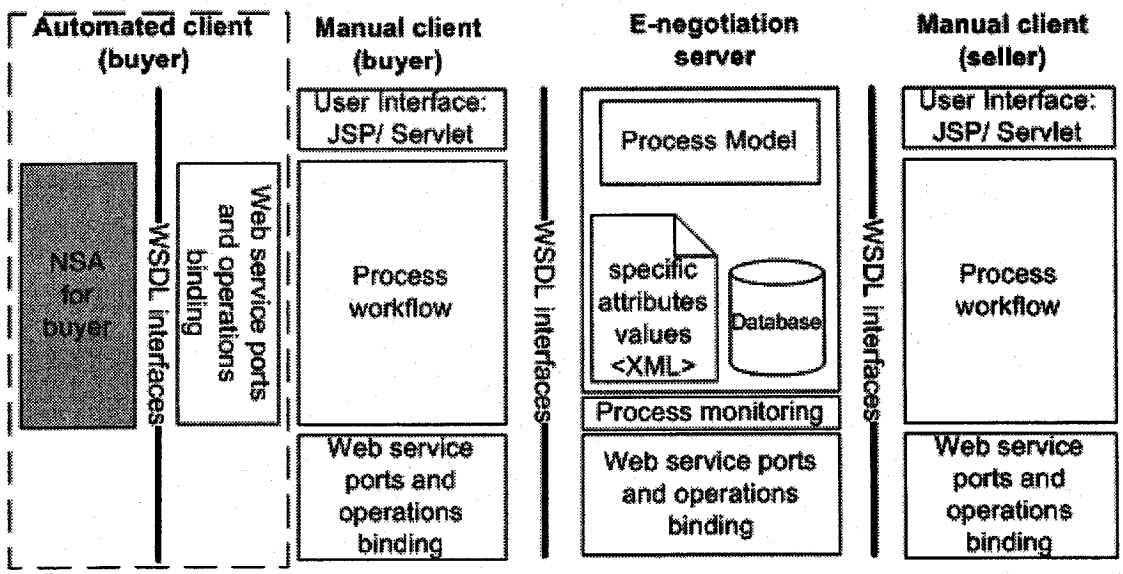

Figure 36: Manual and automated clients

To simplify the connection between the BPEL engine and the clients (manual or automated), we use the *Façade object*. The Façade object can be used to `get` and `set` variables in the BPEL engine, or `invoke` other services (see Figure 37).

In terms of using the MVC design pattern, the *model* is where clients keep the logic of the process, such as handling incoming or outgoing messages; adding artificial intelligence capabilities to the system; forwarding the information further to a customized NSA, and giving the Façade object instructions (`get`, `set`, `invoke`, or `receive`). The *view* is where the display is generated based on the current negotiation phases and variables. The *controller* is an HTML form where the user provides input (textboxes, buttons, drop-down menu box, etc.).

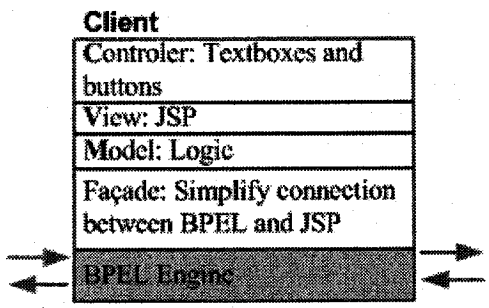

Figure 37: Clients' user interface

Each participant (seller or buyer) may depend on a user input during the negotiation process. To address this requirement, BPEL provides a component called `user task`. This component works together with the Façade object so that it can forward the demanded input to an HTML form. For example, when the server sends an InvitationToSell( ) to a seller, the seller will see that there is a hyperlink captioned "InvitationToSell" on his/her web page. This hyperlink caption can be renamed to provide a better user interface. The appearance of this hyperlink depends on the browser's *refresh button*. Once the seller clicks on that hyperlink, a new page will be displayed with a set of textboxes (to fill the user's ID, password, item name, item description, and price) and a submit button (to send the offer to the server).

Chapter 6

Implementation and Testing

6.1 Implementation

We have implemented and deployed two instances of our e-negotiation server, one running and English auction and one running a Dutch auction. Both instances are based on BPEL process flows which were described earlier in Chapter 5. These BPEL process flows are compiled by BPEL Process Manager and deployed and hosted on one of our workstations using Oracle Application Server.

We also implemented a client for sellers and two types of clients for buyers. The difference between the two buyer clients is that one of them uses a simple NSA. In our English auction, this NSA compares the current highestBid with myBudget (maximum amount to be spent). If the highestBid is lower than myBudget and the highestBidderId is not identical to the buyer's id, the NSA will automatically bid on behalf of the buyer; this is also known as *proxy bidding* (similar to the bidding system offered by eBay.com, but different because in our case proxy bidding takes place on the bidder's side, not on the server).

The NSA strategy, whether it is simple or complex, can be implemented as a set of hard-coded if-then rules, as an external XML configuration file, or as a web service (see Figure 38). By following the last approach, participants can extend and customize their own bidding strategy to suit their needs.

In our implementation, we decided to dispatch and process the information from the buyer interface to an NSA web service which is deployed using the Apache Tomcat, hosted in a different physical location. This expansion was easily accomplished since web services are interoperable across various platforms. This NSA web service creates an instance of JESS, a rule engine, to make inferences using a knowledge base of binding strategies. We refer to this instance as JessAgent (Figure 38 step 4). The rule engine provides an added value to our architecture. It can flexibly parse an external rule file which can be modified anytime including at runtime.

Figure 39 illustrates the steps for running our experiment on three workstations. The steps are sequenced as follows: (1) We run the server and invite the seller to sell; (2) The

seller responds by sending his/her ID, password, the item description, and the reserve price; (3) The server posts this item to all possible buyers; (4) The participants who use NSA enabled clients have to set a variable: MyMaxBid, which is the maximum price they are willing to pay for the item. Then the automated clients forward to the NSA their ID, password, and MyMaxBid. Participants who do not use an NSA enabled client do not go through steps 4, 5, 6 and 7 in Figure 39. Instead they decide whether or not to bid manually. (5) Upon receiving the SOAP message, the workstation that is hosting the NSA web service creates an instance of a JessAgent. (6) After evaluating the current bid and the MyMaxBid, the JessAgent will make a decision whether to bid or not. (7) This decision is sent back to the caller. (8) The automated clients forward the NSA's decision, whether to bid or not, to the server.

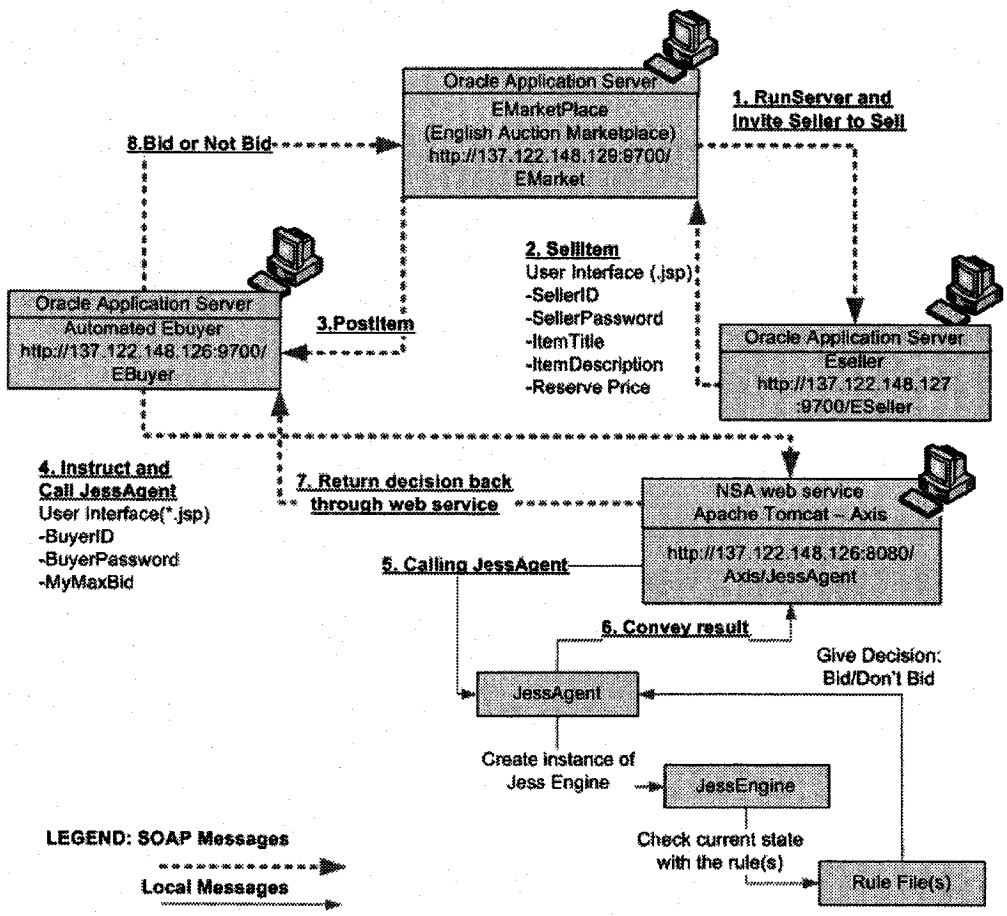

Figure 38: Complete English auction scenario

6.2 Testing and Case Study

In this sub section we will provide several screenshots that we captured during the testing phase of this thesis. We tested this our framework using three desktop PCs.

6.2.1 English Auction Simulation

The English auction starts whenever a seller replies to an InvitationToSell( ) that is sent by the e-negotiation server. From the seller's side, this invitation is captured and reflected as a hyperlink (Figure 39).

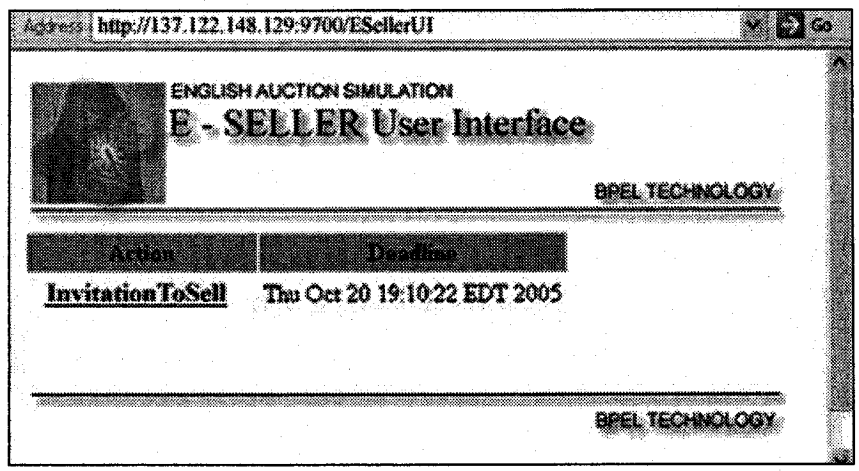

Figure 39: Invitation to sell

Once the seller clicks on the link above, he/she will go to the next screen where he/she will be asked for the user ID, the password, the name of the item, the item's description, and the reserve price. All fields are required. The user ID and the password are used for authenticating the seller. For instance, as can be seen in Figure 40, the seller ID is "Mr.Scheller", and he wants to sell an "Acer Aspire Notebook" with a reserve price of $1800.

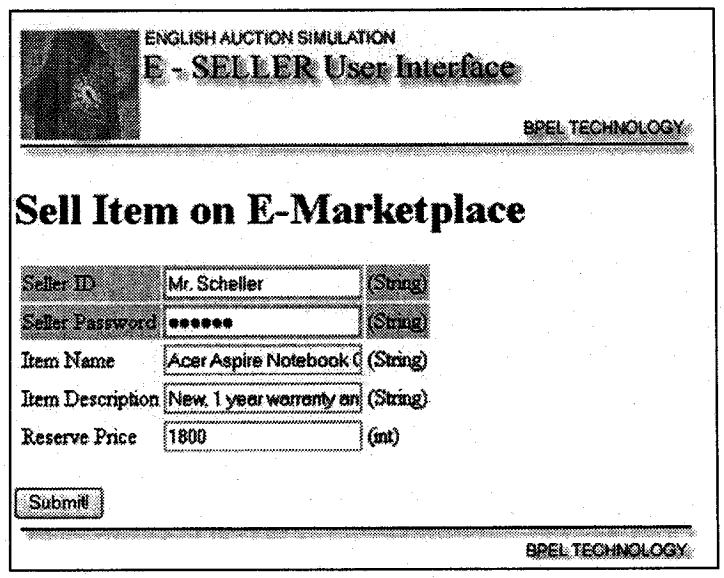

Figure 40 Sell item on e-marketplace page

The submit button will take the seller to the confirmation page as shown in Figure 41. The server will add the item to the database and then broadcast the PostOffer( ) to all possible buyers. If buyers are interested, they can Bid( ) on this item and an Acknowledge( ) message will indicate to them whether their bid is valid or not.

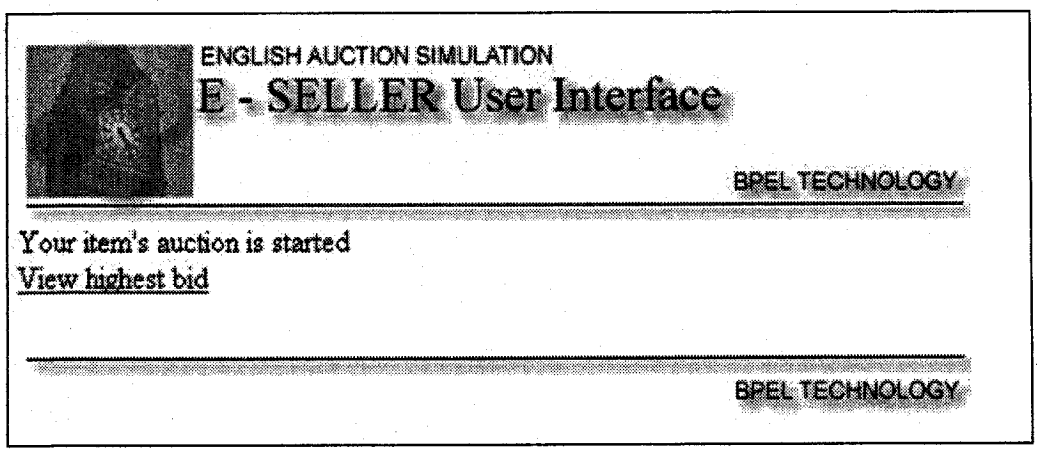

Figure 41: Sell item confirmation

Potential buyers can see the offers that are posted (Figure 42). In our case here, there is only one item to auction off, which is the Acer notebook. If a buyer is interested in bidding on this item, he/she will click on the link and continue to the next screen.

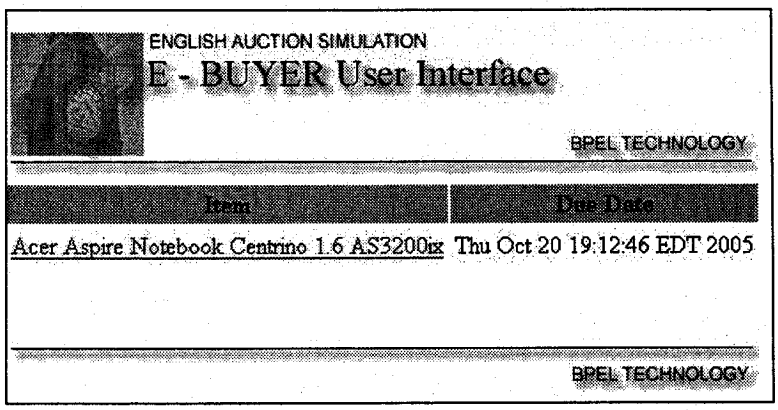

Figure 42: Offers to sell

From the buyer's side, two types of clients/interfaces were implemented for this simulation: one for manual bidding, and one for automated bidding with the help of an NSA. The NSA (the JessAgent) performs a simple proxy bidding on behalf of the human bidder. The principle is to keep bidding automatically as long as a certain dollar amount is not reached. Note that the NSA can adopt a more sophisticated behavior, but this not the subject of this thesis. Figure 43 shows what the manual bidder will see when he/she accesses the link.

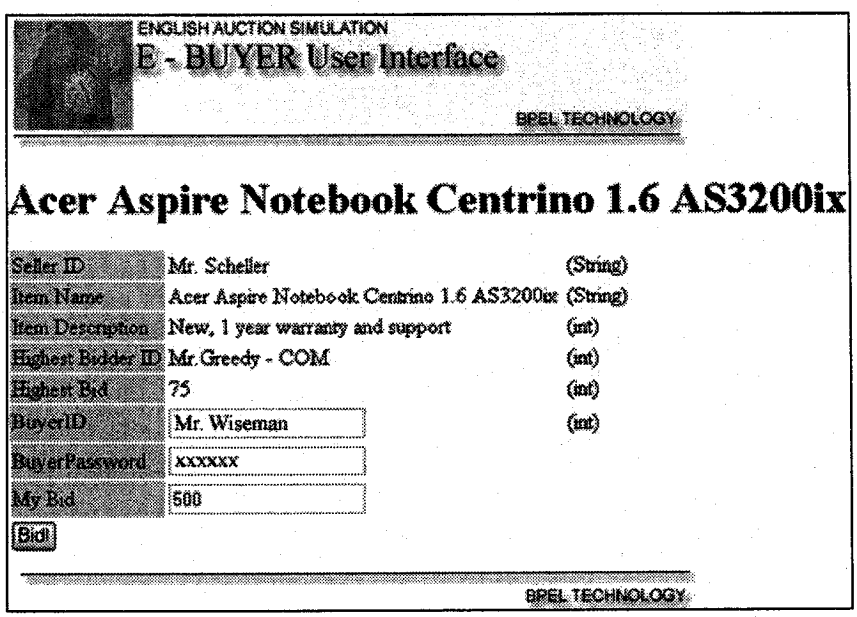

Figure 43: Manual bidding

After the simulation is over, we can examine a trace of the process by viewing the server's log screen. As shown in Figure 44:

(1) First the seller posts his item

(2) A JessAgent, with user ID "Mr. Greedy" then bids $75

(3) Then a human buyer, Mr. Wiseman, bids $500

(4) The agent again bids for $575

(5) Towards the end of the auction, the human bids $1800.

(6) Because the reserve price is $1800, the human buyer became the winner of the English auction simulation.

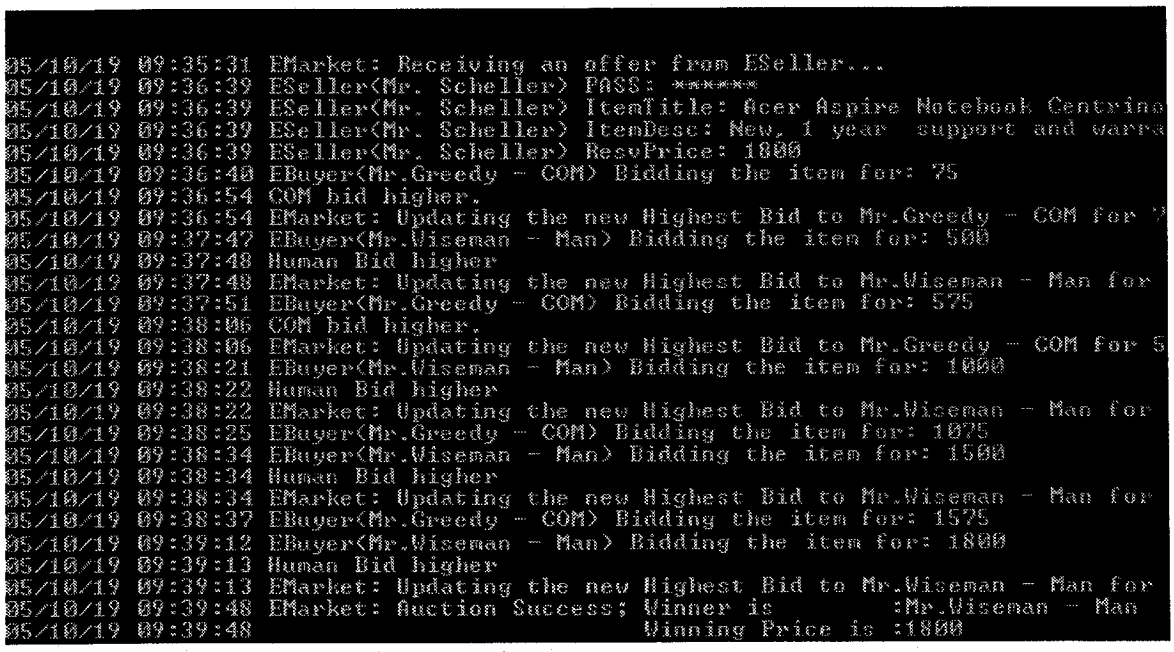

Figure 44: English auction log screen

.

6.2.2 Dutch auction Simulation

In a Dutch auction, the seller receives an invitation to sell from the server, which will be reflected in the seller's page as shown in Figure 45.

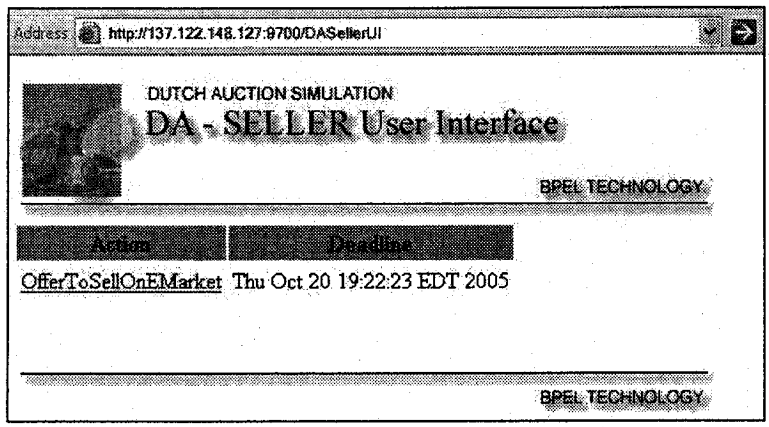

Figure 45: Invitation to sell in a Dutch auction

The seller must then submit his/her sellerID, sellerPasword, itemName, itemDescription, numberOfItem, and unitPrice. After submitting his/her offer (Figure 46), the seller is given the opportunity to gradually reduce the price until the end of the auction. The price reduction can be automated to occur periodically or it can be done manually through seller input, as shown in Figure 46.

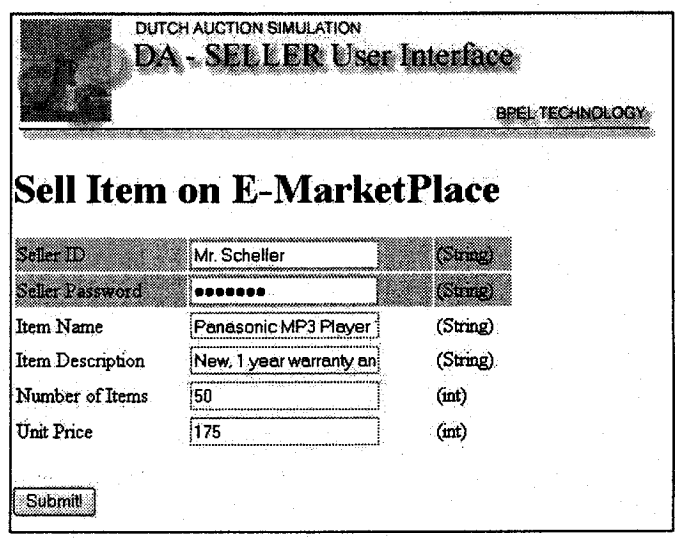

Figure 46: Selling multiple items in Dutch auction

Dutch auctions are well known for being highly competitive and fast-paced since the first bidder will automatically win the item in question. If there is more than one item, the auction will continue until all the items are sold or the deadline of the auction is reached. In this case, the last bidder pays the lowest price.

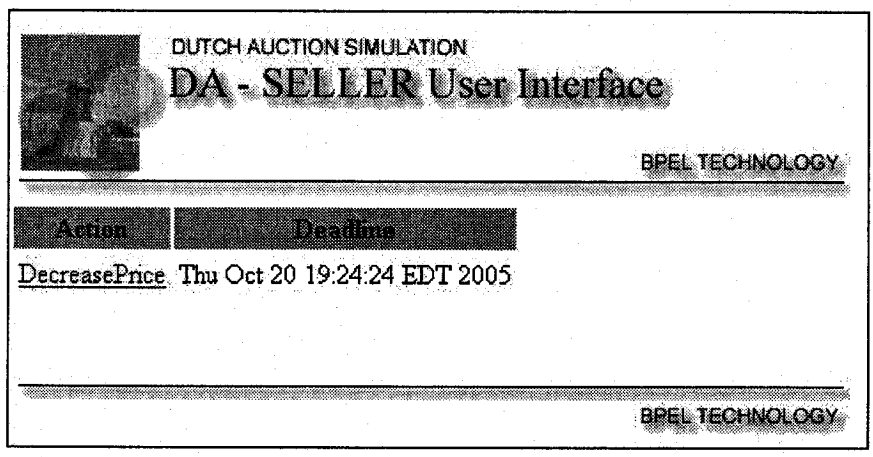

Figure 47: Option to decrease the price in a Dutch auction

The Dutch auction bidding strategy requires a more complex NSA which can be better implemented as another web service. The participant's system must pass enough information to this service in order to make the appropriate bidding decisions. The decision making strategy for the NSA is beyond the scope of this thesis and will be discussed at a later date in another publication. However, providing an extendable platform which accommodates this requirement is explained below.

For example, human bidder can instruct the NSA to buy a certain quantity of the item being auctioned off once the price has dropped past a certain amount. Figure 48 shows the interface for bidders with NSA.

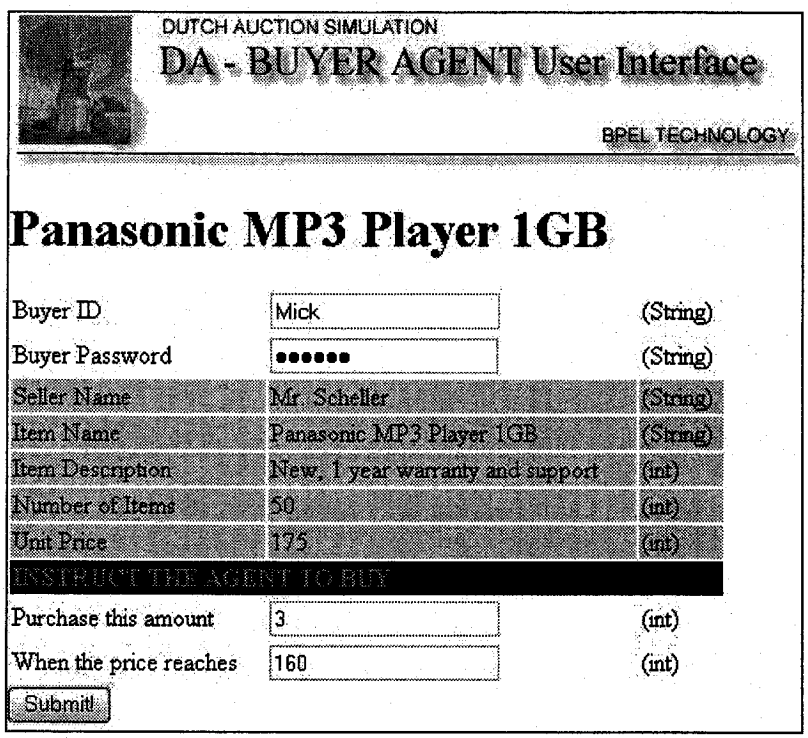

Figure 48: Instructing the NSA to buy

In Figure 48, the bidder will have to fill out his/her ID, password, and instruct the NSA to buy a certain number of items once the price reaches a preferred amount. In this example, the seller is offering 50 items for $175 each. The buyer with the NSA would like to buy 3 of them once the price reaches to $160. As shown in the Figure 49 below.

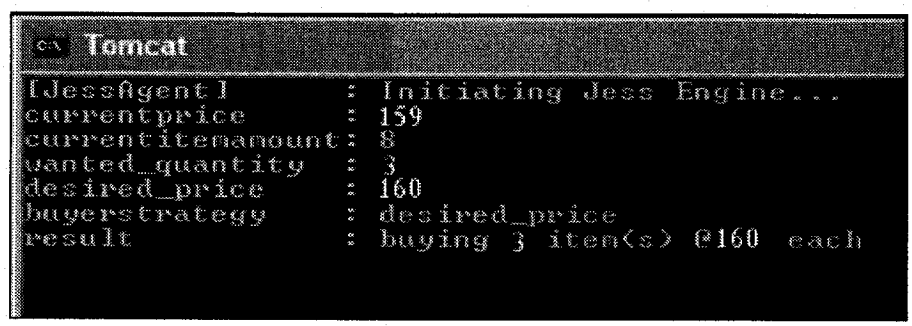

Figure 49: NSA web service log screen

As for bidders without NSA, they use another interface to enter the user ID, password, and the number of units they want to buy at the current price as shown in Figure 50.

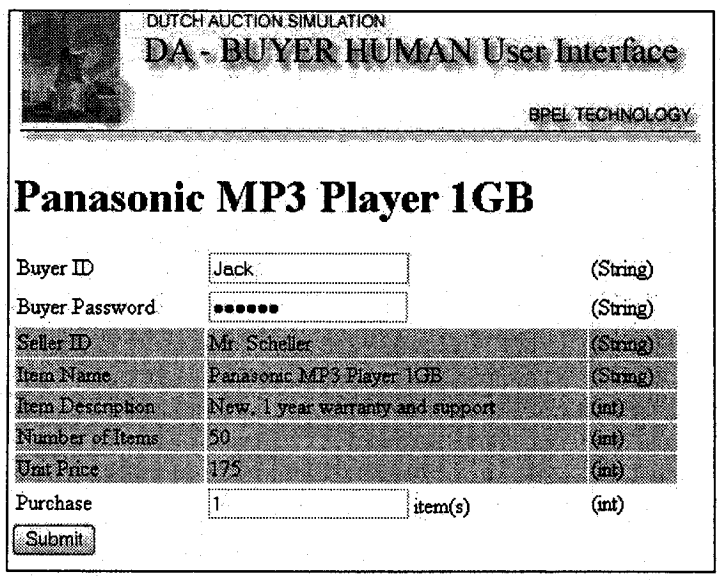

Figure 50: Manual bid in a Dutch auction

In the example above, the buyer make a bid for 1 item at $175. Figure 51 shows the interface for the seller to decrease the price.

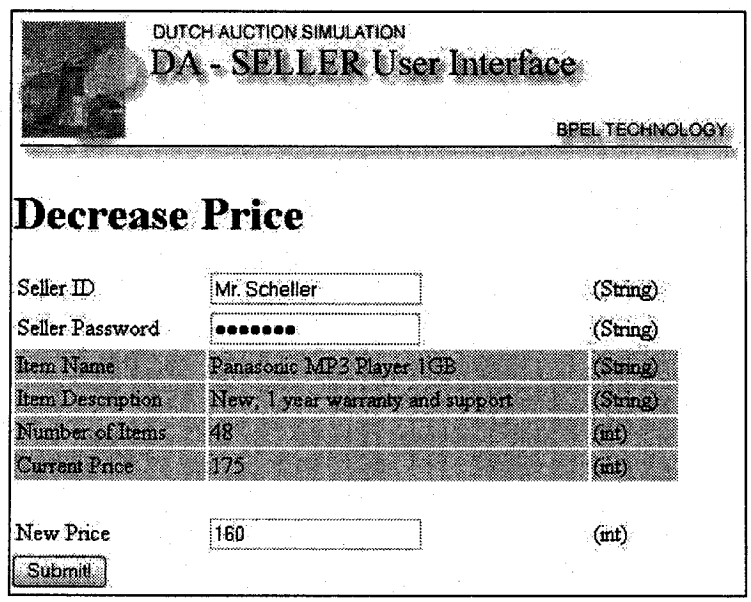

Figure 51: Seller decreases the price in a Dutch auction

The full log of a Dutch auction simulation can be seen on the log screen below (Figure 52). During the process, we can see that the seller submits his 50 items for $175 each. A buyer named Jack then buys one of the items. After a period of time, another buyer named John buys another item. The amount of items is now 48 units. Then, the seller decreases the price to $160 per unit. The NSA then automatically buys three items for a buyer named Mick. After the timeout, the auction will end.

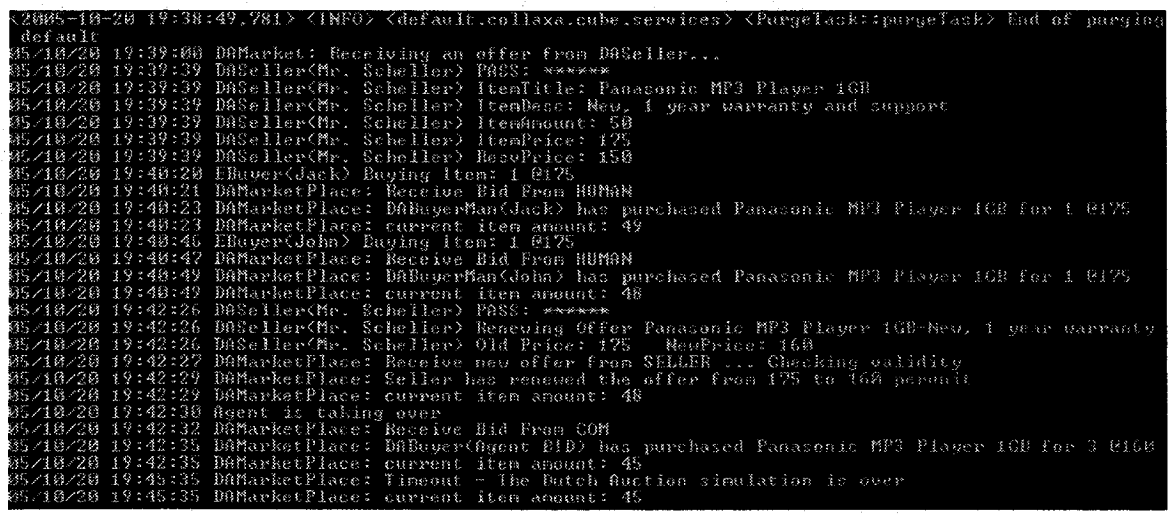

Figure 52: Log screen for a Dutch auction simulation

Chapter 7

Conclusion and Research Perspectives

7.1 Conclusions

In this thesis, we presented a thorough classification of existing e-negotiations servers. This classification has become the reason why we propose a new e-negotiation server that is based on SOA. Our server is generic enough to facilitate various types of negotiations, and encourage reusability. To create a new negotiation, the system designer only needs to model the activity diagram visually (using BPEL) or reuse existing components. This approach reduces the necessity for system designers to understand programming concepts, at least for the high level design, keeping them from being trapped in the design's complexity, and allowing them to focus more on the business view.

Our e-negotiation server promotes loose coupling and reusability. It is easy for us to extend the main system with an additional component such as an NSA, but still using the same ports and interfaces.

A major contribution of this thesis is an extensible negotiation server for future component add-ins. This research has also opened new opportunities for the study of agent behaviors by using various strategies based on this platform (this is the focus of another student thesis, which is still on progress).

With standard open interfaces, such as web services, every service is platform independent. As described in the implementation section, the NSA part is implemented on another machine and hosted by a different web server.

To our knowledge, this research is the first one that thoroughly tests the usage of web service orchestration languages such as BPEL for hosting an e-negotiation server. The closest works are from Kim and Segev (2005) [5] and Rolli and Eberhart (2005) [22]. Other researchers for instance use web service orchestration to develop a business composition for a travel agency that deals with airplane tickets and hotels or for a corporate supply chain case [39, 40].

7.2 Limitations

The limitations of our design are as follows:

- We did not implement any form of security, this issue being beyond the objectives of the thesis.
- During the development process, we discovered several limitations due to web services orchestration languages. Unlike the existing programming languages, BPEL is still growing. The real purpose of BPEL is to model and orchestrate different web services; therefore, its expressiveness is still not as effective as real programming languages. These limitations includes:
    - In a BPEL process activity, there is only one starting point and one exit point for the process. This makes for one monolithic process.
    - Although BPEL provides `java embedding`, its Java Runtime Environments still lacks compatibility with java implementations and APIs, for example Jess. Therefore additional parameters are needed during the class call.
- To utilize a database, we added a `java embedding` component after a `receive` component. For example, after receiving a bid from a buyer, the server writes the transaction (time, bidderID, itemID, and bidAmount) into the bid table. This mechanism is done using a `java embedding` component and a java library extension such as "mysql-connector-java.jar". The java code can be written inside the `java embedding` component or in a separate class file.

7.3 Proposal for Future work

This is an ongoing research; and in the near future, we plan to finalize and tune-up the prototype that validates our approach. Various e-negotiation scenarios can be implemented in addition to the ones we already deployed on our platform. This research has also opened an opportunity for developing a more sophisticated NSA that is based on this server. Various strategies and tactics can be developed within the NSA such as reasoning and prioritizing user preferences over a certain condition. This system can then be used for experiments

such as bidding tournaments between software and human agents.

References

1. Bichler, M., G. Kersten, and S. Strecker, *Towards a Structured Design of Electronic Negotiations.* Group Decision and Negotiation Vol.12, 2003: p. 311–335.

2. Hurwitz, *Negotiated Trade: the Next Frontier for B2B e-commerce*, in *Technical Report.* 2000.

3. Kumar, M. and S.I. Feldman, *Business negotiations on the Internet.* IBM Research Division - T.J. Watson Research Center, 1998.

4. Kersten, G.E. *E-negotiation systems: Interaction of people and technologies to resolve conflicts.* in *UNESCAP Third Annual Forum on Online Dispute Resolution*, Melbourne, Australia. 5-6 July, 2004.

5. Kim, J.B. and A. Segev, *A Web Services-enabled Marketplace Architecture for Negotiation Process Management.* Decision Support Systems Vol. 40, 2005: p. 71-87.

6. Rust, J., J. Miller, and R. Palmer, *Behavior of trading automata in a computerized double auction market.* The Double Auction Market: Institutions, Theories, and Evidence. 1993: Addison Wesley. p153–196.

7. Benyoucef, M., et al. *Towards a Generic E-Negotiation Platform.* in *Proceedings of the 6th International Conference on Re-Technologies for Information Systems*, Zurich, Switzerland. 2000. LNCS 1830 p.95-109.

8. Mathieu, P. and M.-H. Verrons. *ANTS: an API for creating negotiation applications.* in *10th ISPE International conference on concurrent engineering: research and application*, Madeira Island - Portugal. 26-30 July, 2003.

9. Basu, A. and A. Kumar, *Research commentary: workflow management issues in e-business.* Information Systems Research, 2002, Vol.13: p. 1- 14.

10. Bartolini, C., C. Preist, and N.R. Jennings, *A Generic Software Framework for Automated Negotiation.* Trusted E-Services Laboratory - HP Laboratories Bristol, 2002.

11. Weinhardt, C. and P. Gomber. *Agent-Mediated Off-Exchange Trading.* in *Proceedings of the 32nd Hawaii Conference on System Sciences*, Maui, Hawaii, USA. 5-8 January, 1999. p.6-9.

12. Ehtamo, H., R.P. Hämäläinen, and V. Koskinen. *An E-learning Module on Negotiation Analysis.* in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, Maui, Hawaii, USA. 5-8 Jan, 2004. p.9 pp.

13. Lomuscio, A.R., M. Wooldridge, and N.R. Jennings, *A Classification Scheme for Negotiation in Electronic Commerce.* Group Decision and Negotiation, 200312 (12): p. 31–56.

14. Gulliver, P.H., *Disputes and Negotiations: A Cross-Cultural Perspective.* Orlando, FL: Academic Press, 1979.

15. Strobel, M. and C. Weindhardt, *The Montreal Taxonomy for Electronic Negotiations.* Group Decision and Negotiation, 200312 (12): p. 143–164.

16. Bartolini, C., C. Preist, and N.R. Jennings. *Architecting for reuse: A software framework for automated negotiation.* in *Proceedings of the 3rd International Workshop on Agent-Oriented Software Engineering*, Bologna, Italy. 2002. p.87-98.

17. Smith, R.G., *The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver.* IEEE Transaction on Computers, 1980, Vol.C-2912 (12): p. 1104-1113.

18. Kersten, G.E., K.P. Law, and S.E. Strecker, *A Software Platform for Multiprotocol E-Negotiations.* InterNeg Research Papers, INR04/04, 2004.

19. Wu, S., G. Kersten, and M. Benyoucef. *INSS - A New Approach in Designing Web-Based Negotiation Support Systems.* in *The Montreal Conference on e-Technologies*, Montreal, Canada. May 17-19, 2006. p.3-9.

20. Kersten, G.E. and S.J. Noronha, *WWW-based Negotiation Support: Design, Implementation, and Use.* Decision Support Systems Vol. 25, 1999: p. 135-154.

21. Kersten, G.E. and G. Lo, *Aspire: Integration of Negotiation Support System and Software Agents for E-Business Negotiation* International Journal of Internet and Enterprise Management Vol. 1, No. 3, 2003.

22. Rolli, D. and A. Eberhart, *An Auction Reference Model for Describing and Running Auctions.* Wirtschaftsinformatik, 2005: p. 289-308.

23. Su, S.Y.W., C. Huang, and J. Hammer. *A Replicable Web-Based Negotiation Server for E-Commerce.* in *Proceedings of the IEEE 33rd Hawaii International Conference on System Sciences, Volume 8, IEEE Computer Society*, Maui, Hawaii, USA. 2000.

24. Barbacci, M., et al., *Quality Attributes.* Carnegie Mellon University - Software Engineering Institute, 1995.

25. Kazman, R., et al., *The Architecture Tradeoff Analysis Method.* Carnegie Mellon University - Software Engineering Institute, 1998.

26. Bass, L., P. Clements, and R. Kazman, *Software Architecture in Practice.* 1997: Addison-Wesley.

27. Gamma, E., et al., *Design Patterns: Elements of Reusable Object-Oriented Software.* Reissue ed. 1995: Addison-Wesley Professional.

28. Nickull, D., *Service Oriented Architecture* Adobe Systems Incorporated White-paper, 2005.

29. McGovern, J., et al., *Java Web Services Architecture*. Data Management Systems. 2003: The Morgan Kaufmann Series.

30. Monson-Haefel, R., *J2EE Web Services*. 2003: Addison-Wesley.

31. Peltz, C., *Web Services Orchestration.* Hewlett-Packard Company, 2003.

32. Wohed, P., et al., *Pattern Based Analysis of BPEL4WS.* Technical Report FIT-TR-2002-04, Queensland University of Technology, Australia, 2002.

33. Aalst, W.v.d., A. Ter Hofstede, and M. Weske. *Business Process Management: A Survey*. in *Proceedings of the International Conference on Business Process Management*, Eindhoven, The Netherlands. June 26-27, 2003. LNCS 2678 p.1-12.

34. Aalst, W.v.d. *Workflow Patterns*. 2003 [cited 2005 August]; Available from: http://tmitwww.tm.tue.nl/research/patterns.

35. Rinderle, S. and M. Benyoucef, *Towards the Automation of E-Negotiation Processes Based on Web Services - A Modeling Approach.* Lecture Notes in Computer Science, 2005, Vol.3806: p. 443 - 453.

36. Fowler, M., *UML distilled: a brief guide to the standard object modeling language.* 3rd ed. Addison-Wesley object technology series. 2004, Boston: Addison-Wesley.

37. Trickovic, I., *Modularization and reuse in WS-BPEL.* SAP Developer Network - SDN Community Contribution Whitepaper, 2005, Vol.October.

38. Kloppmann, M., et al., *WS-BPEL Extension for Sub-processes - BPEL-SPE.* SAP Developer Network Whitepaper, 2005, Vol.September.

39. Charfi, A. and M. Mezini. *Hybrid web service composition: business processes meet business rules*. in *Proceedings of the 2nd International Conference on Service Oriented Computing*, New York, NY, USA. 2004. p.30-38.

40. Tewoldeberhan, T.W., A. Verbraeck, and S.S. Msanjila, *Simulating Process Orchestrations in Business Networks: A Case Using BPEL4WS.* ICEC, 2005, Vol.August 15-17: p. 471-477.

41. Costa, M.B., et al. *Business to Business Transaction Modeling and WWW Support.* in *Business Process Management International Conference No.2*, Potsdam, Gemany. 2004. LNCS 3080 p.132–147.

42. B.Juric, M., B. Mathew, and P. Sarang, *Business Process Execution Language for Web Services Ch. 4.* 2004: Packt Publishing.

43. Mathieu, P. and M.-H. Verrons. *A Generic Negotiation Model for MAS Using XML.* in *ABA workshop Agent-based Systems for Autonomous Processing, IEEE International Conference on Systems, Man and Cybernetics*, Washington, D.C., USA. 5-8 Oct, 2003. p.4262- 4267

44. Gérin-Lajoie, R., et al., *GNP V1.0 - Architecture Document.* Cirano, 2000, Vol.July.

45. Futrell, R.T., D.F. Shafer, and L.I. Shafer, *Quality Software Project Management* 1ed. 2002: Prentice Hall.

46. Buschmann, F., *Building Software with Patterns.* Siemens AG Corporate Technology, 1999.

47. Garland, D. and M. Shaw, *An Introduction to Software Architecture.* World Scientific, 1994.

48. Hélou, C., R. Dssouli, and T.-G. Grainic, *Performance Testing of a Negotiation Platform.* Information and Software Technology No.44, 2002: p. 313-330.

49. Cheung, S.-C., D.K.W. Chiu, and S. Till, *A Three-Layer Framework for Cross-Organizational e-Contract Enactment.* Lecture Notes In Computer Science, 2002, Vol.2512: p. 78 – 92.

50. Kersten, G.E., *E-Negotiations: Towards Engineering of Technology-based Social Processes.* E-Negotiation Engineering, 2002.

51. Whitby, A., Theory and Design of a Generic Auction Server, Bachelor of Computer Science Thesis, University of Queensland, Queensland-Australia, 2004.

Appendix

Appendix A: Research plan

Activities	Start Date	End Date	Notes
Web service and WS Orchestration modeling	May 05	August 05	Oracle BPEL Process Manager, IBM Rational XDE, BEA WebLogic
E-Negotiation theories, concepts, and implementations	September 05	December 05	Reading related work in academic papers
Extending BPEL to JESS	November 05	January 05	Moving the execution logic to JESS Rule engine
Writing thesis chapter 1 to 4	February 06	May 06	Introduction, negotiation, e-negotiation, auctions, classifications, e-negotiation servers, and the proposed framework, SOA, etc.
Writing thesis chapter 5 to 7	May 06	July 08	Requirement analysis, design, implementation, testing, case study, and conclusions.
Revision from supervisor	August 08	August 08	Revision, addition, and removal of some materials on this Thesis.

Table 9: Research plan

Appendix B: BPEL Process Flow Diagrams – Screenshots

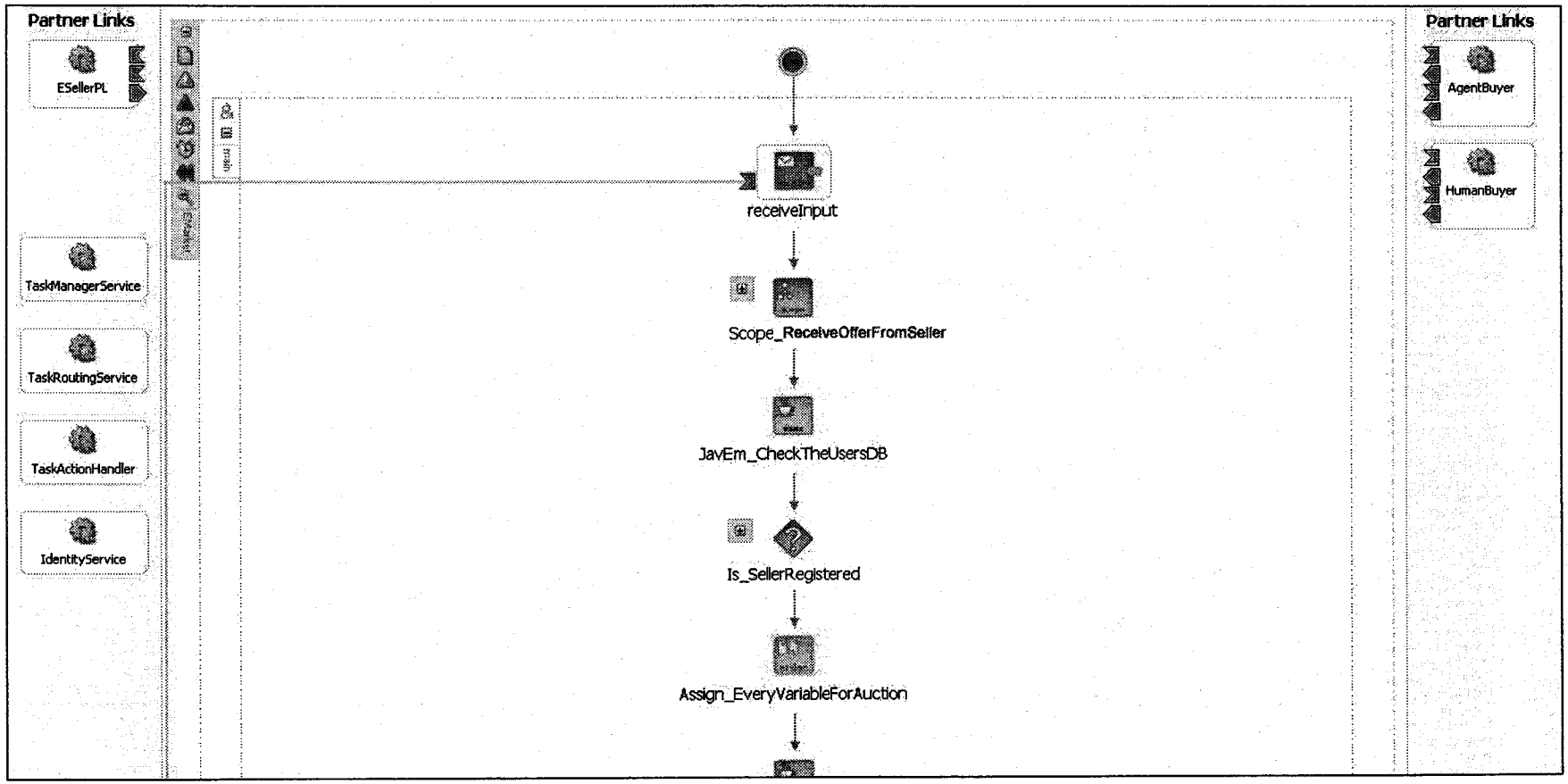

Figure 53: BPEL Process Flow Diagram for the English auction (top section)

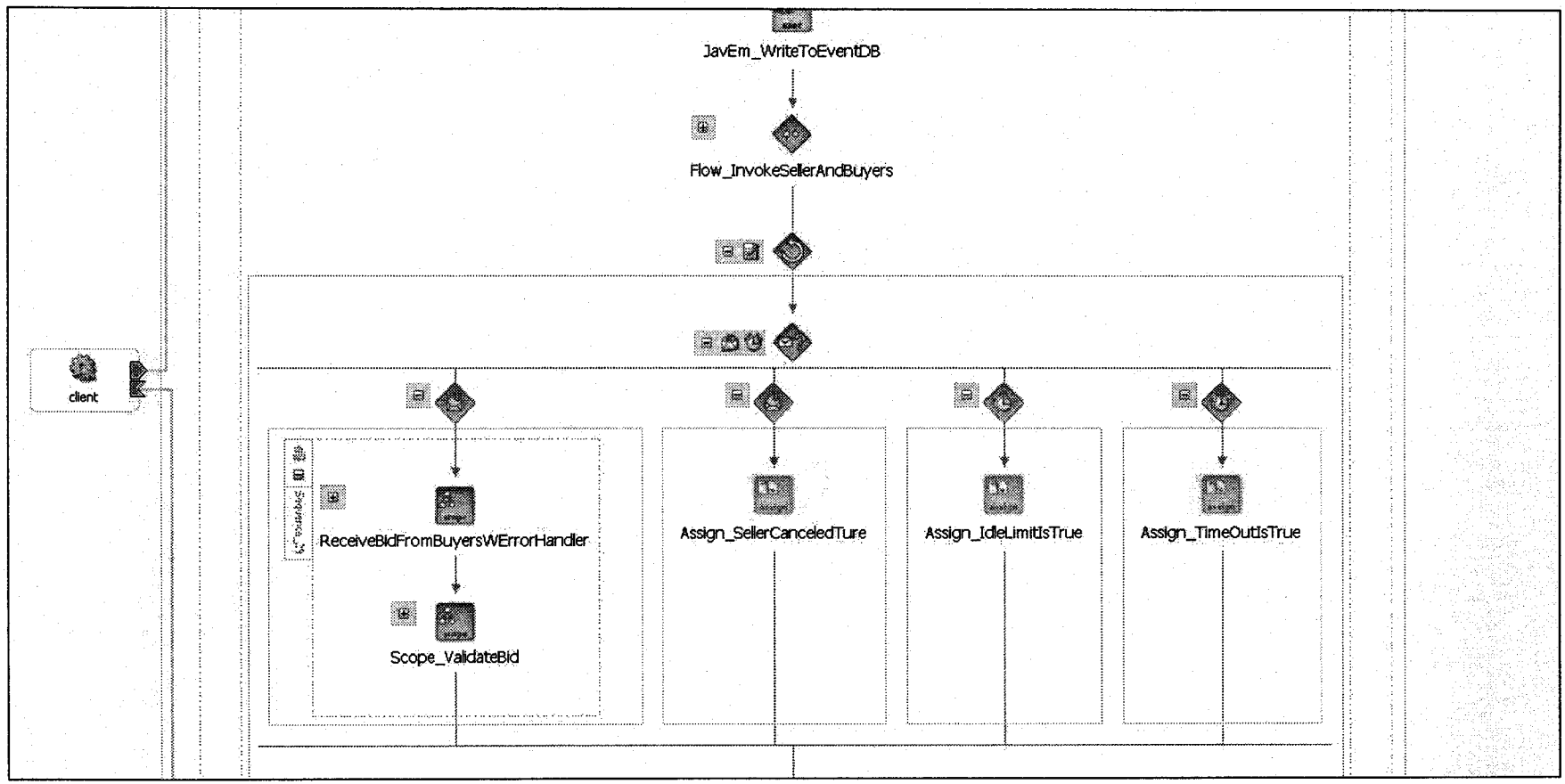

Figure 54: BPEL Process Flow Diagram for the English auction (middle section)

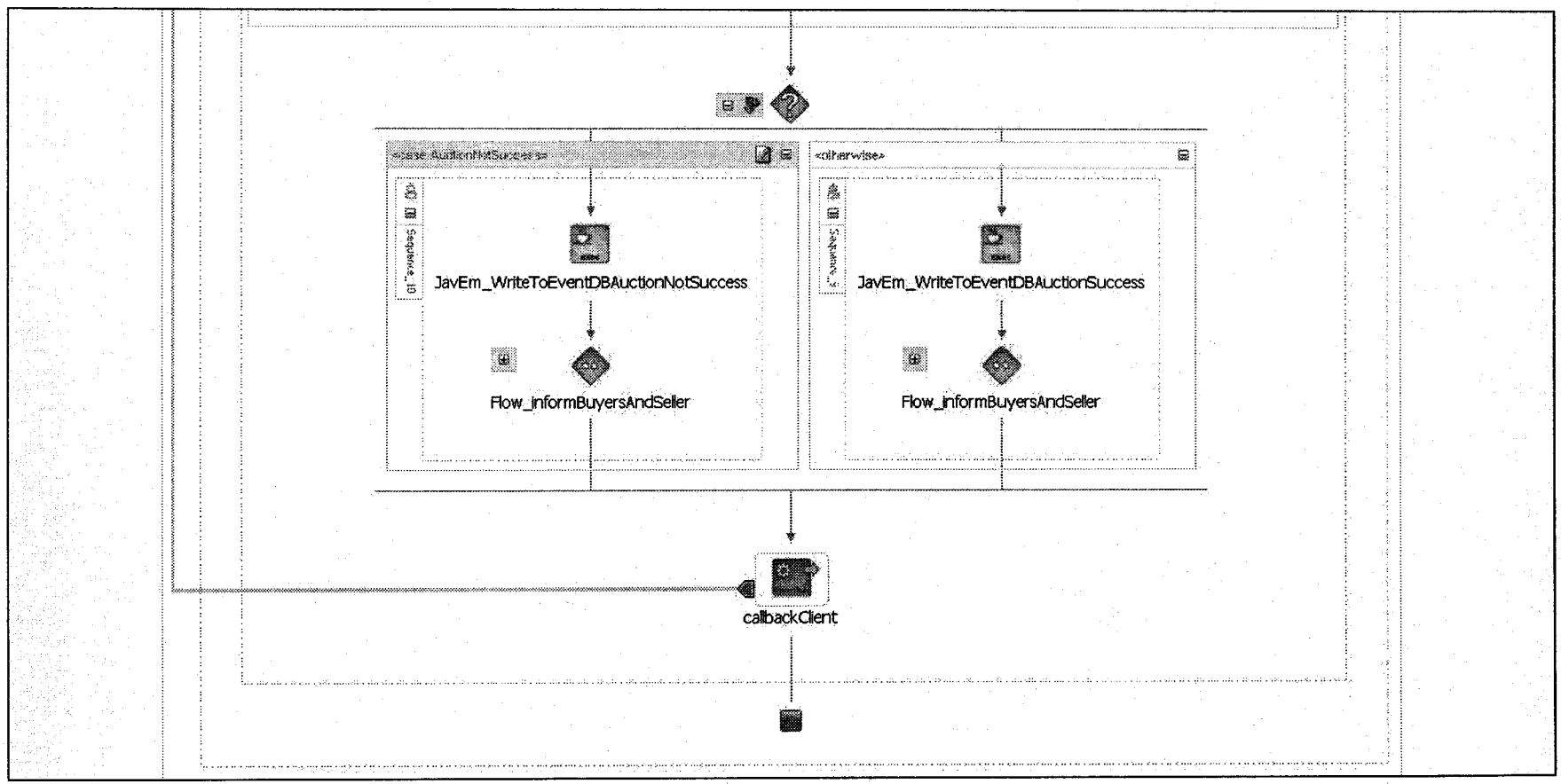

Figure 55: BPEL Process Flow Diagram for the English auction (bottom section)

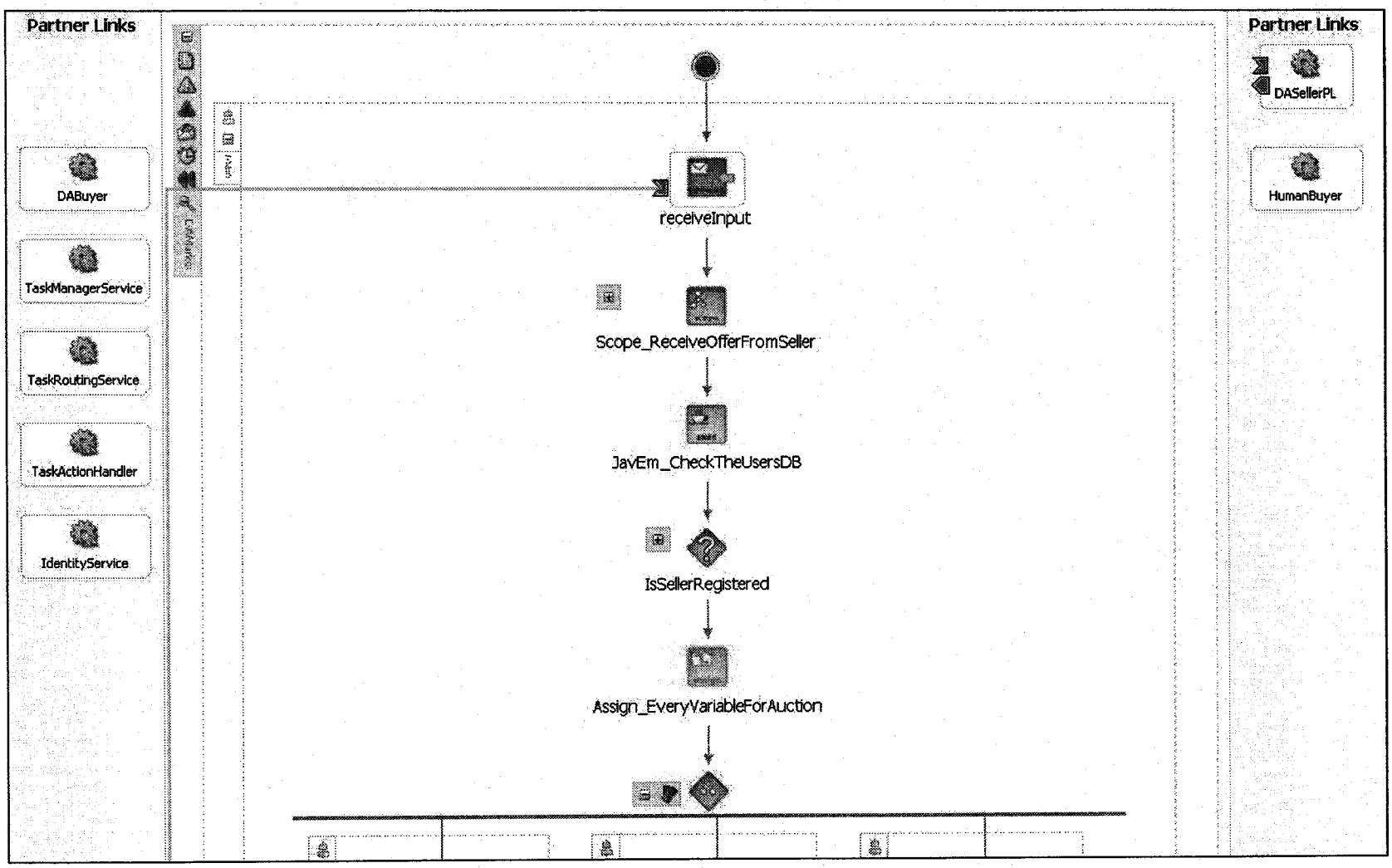

Figure 56: BPEL Process Flow Diagram for the Dutch auction (top section)

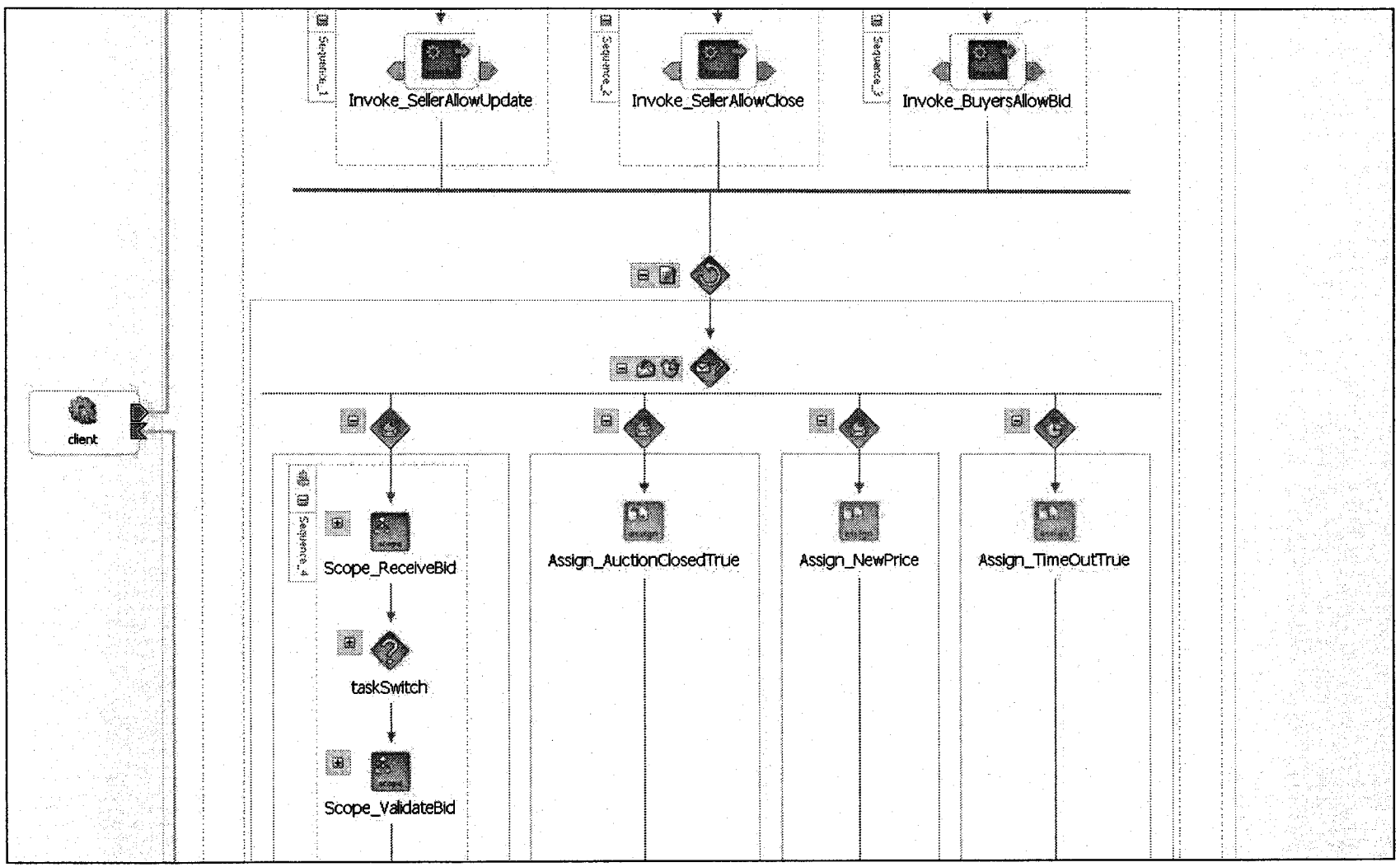

Figure 57: BPEL Process Flow Diagram for the Dutch auction (middle section)

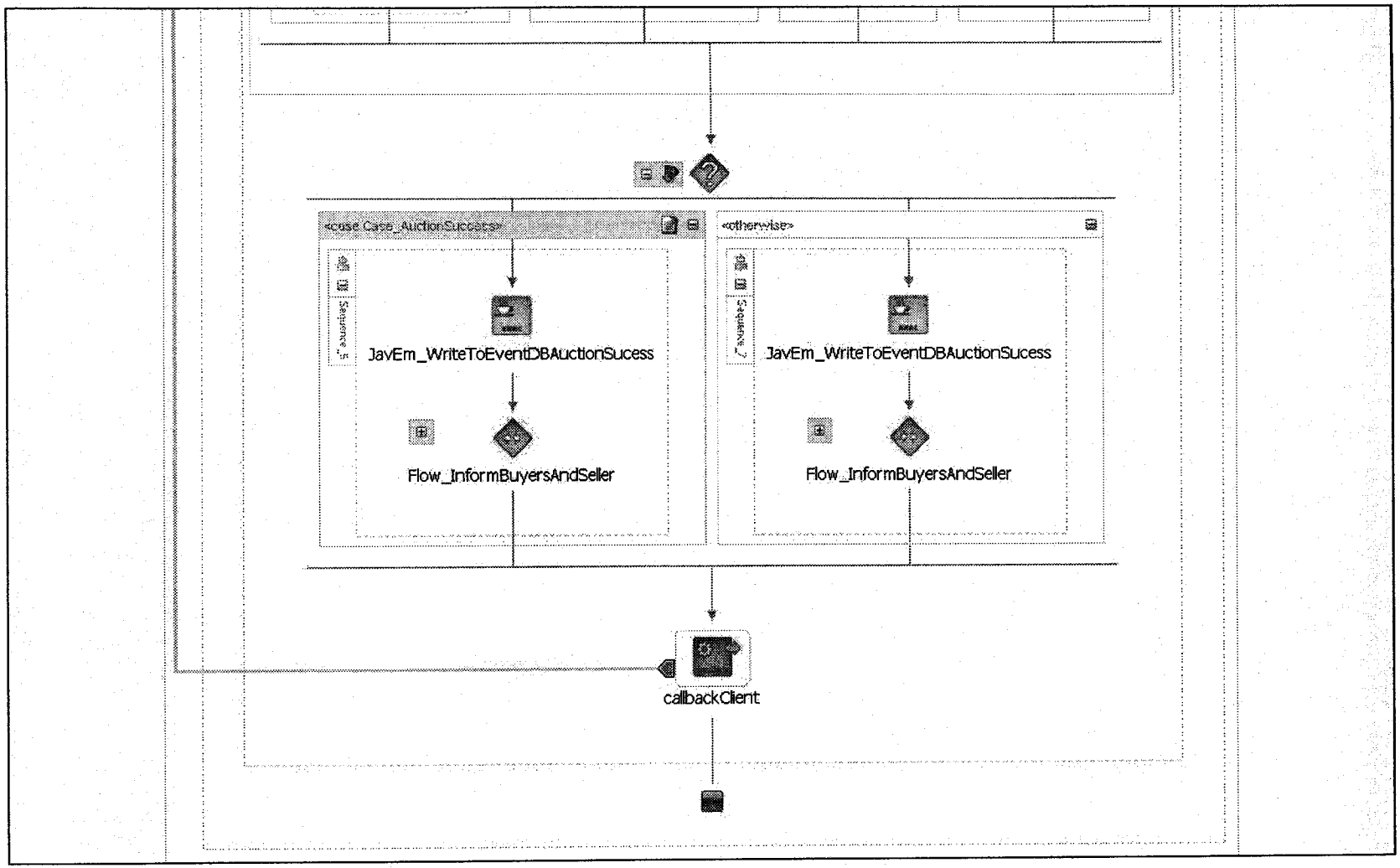

Figure 58: BPEL Process Flow Diagram for the Dutch auction (bottom section)