A BPEL Based Implementation of Online Auctions

Morad Benyoucef[1] and Ronald Pringadi[2]

[1] School of Management
[2] School of Information Technology and Engineering
University of Ottawa, 136 Jean-Jacques Lussier St. Ottawa, Ontario K1N 6N5, Canada
{benyoucef@management.uottawa.ca, pringadi@site.uottawa.ca}

Abstract. Service oriented architectures have been adopted by many organizations in order to increase business automation, integrate enterprise systems, and reach more business partners and customers. Among the business processes that can greatly harness the advantages of web services are online auctions. This paper proposes a new approach for modeling online auctions and provides guidelines on how to use web service orchestration to model them. Our research involves the design of an auction server and its clients for manual as well as automated agent-based bidding. We propose a framework for a server that is generic enough to host various types of auctions and extensible enough for future component additions. We report on an implementation of our framework based on the BPEL web service orchestration language.

Keywords: online auction, process modeling, web service orchestration, BPEL.

1 Introduction

Negotiation, as defined in [1], is 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 very intensive, time consuming, and costly; therefore, there is a need to automate it [2]. Any negotiation conducted using computers or other electronic devices is referred to as electronic negotiation (*e-negotiation*). Because of the strong domination of auctions in the field of negotiations, some might think that e-negotiations primarily consist of online auctions [4]. This is obviously not true. Although this paper concentrates on online auctions, we believe that other negotiation protocols (i.e., scenarios or styles) can and should be conducted electronically.

Current research on e-negotiation systems (ENS) mostly focuses on negotiating software agents, negotiation strategies, and negotiation automation. Few initiatives addressed the problems of designing a platform that enables these interactions. After a thorough review of the literature on e-negotiation frameworks for servers and clients [5] we learned that although these frameworks have several design qualities, most do not foster reusability and extensibility. Furthermore, most frameworks fail to address the following requirements: the design should support various negotiation protocols [6], it has to offer high flexibility [7], and it must foster easy development and deployment [8]. Moreover, we believe that it must facilitate the interactions of negotiators, enable the participation of human as well as software agents, and permit the

D. Georgakopoulos et al. (Eds.): ICSOC 2006 Ws, LNCS 4652, pp. 104–115, 2007.

seamless integration of the e-negotiation platform with different applications inside the organization and across partner organizations.

Designing an e-negotiation platform which addresses all the requirements mentioned above is obviously not an easy task, especially knowing that it will 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. A marketplace-based architecture connects companies more efficiently than a point-to-point connection between every buyer and every supplier. Other design challenges include formalizing the shared protocol [10], extracting the business logic from the server process, which is needed by the clients to properly interact with the server [8], and validating the rules of negotiation to ensure a fair and correct process.

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 operational costs. This move has shifted the trend of object-oriented design into a service oriented design, where software modules are converted into web services and published over the Internet to be used as-is or to be integrated with other applications. Because web services are XML based protocols, they have the advantage of providing a platform-independent service which facilitates B2B and A2A integration.

The contributions of this paper are threefold. First, we work within a new vision for designing e-negotiation systems (servers and clients) using a service oriented approach. We propose a design framework for auction servers and clients to be used for manual as well as automated agent-based bidding. The framework is generic enough to host various types of auctions and extensible enough for future component additions. The framework is centered on modeling auctions as web service orchestrations. We deploy a common interface for human participants to interact directly with the server, and for software agents to be configured and controlled by their owner. Second, we report on an implementation of our framework based on the BPEL web service orchestration language. Third, we focus on the task of modeling auction processes as web service orchestrations. Although in this paper we only report on the English auction, we modeled and enacted several auction processes such as the Dutch and sealed-bid auctions as well as negotiation processes such as the two-party negotiation. We provide modeling guidelines and identify common components in the models that can constitute a repository to be used when modeling negotiation protocols.

The rest of the paper is organized as follows. Section 2 discusses auctions as a fundamental form of negotiation. Section 3 introduces general design considerations for e-negotiation systems. Section 4 reports on the architecture and implementation of our online auction framework. Section 5 is dedicated to modeling auction processes. We discuss our modeling approach in Section 6, describe related work in Section 7, and wrap up the paper with a conclusion in Section 8.

2 Auctions as a Form of Negotiation

Auctions are a form of negotiation [4]. Negotiation itself is divided into business and non-business negotiation. Business negotiation includes auctions, two-party

negotiation (i.e., bargaining), business procurement, brokerages, exchanges, and cartels [3], while non-business negotiation includes dispute resolution and voting.

There are various types of auctions, and at least five key elements can be extracted from them [9]: (1) a deal which can be in various states such as "negotiable offer" or "final offer"; (2) participants such as buyers, sellers, and auctioneers; (3) messages sent by participants to modify the deal such as "new bid"; (4) process flow describing how the state of the deal changes as a result of the messages sent by participants; and (5) messages sent to participants as the deal changes.

Auction scenarios (also called protocols) specify the rules of the negotiation [11], and different scenarios serve different purposes. Some of the most common scenarios are the English auction, Vickrey auction, sealed bid auction, and Dutch auction. The English auction is perhaps the most widely used scenario. Its popularity has grown mainly as a result of its adoption by eBay (www.eBay.com). 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 protocol [12]. The English auction is a process where a seller tries to sell an item and potential buyers bid on the item. The bid increases over time. At the end of the process, which can be triggered by a timeout or when no bids have been received for a period of time, the highest bidder wins the item. A starting bid can be set to ensure that bids are close to the item's estimated value. A reserve price can be set to protect the seller from selling below a certain price. A variation of the English auction is the Vickrey auction. Here, the winning bidder pays the second highest bid, which encourages potential buyers to outbid one another. Sealed bid auctions are held when, among other situations, it is impractical for bidders to prepare bids instantaneously or the bid confidentiality is important. In single round sealed bid auctions, all bidders submit their bids by a deadline; the bids are then 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 started with a new deadline. The Dutch auction is a mechanism where the seller starts the process by setting the price of an item. As the auction progresses, the seller lowers the price gradually until a buyer bids on the item. If there is only one unique item, the auction ends immediately with the bid. Otherwise, the auction continues, and the seller keeps lowering the price and buyers making bids until the last item is sold.

3 General Design Considerations

There are two basic components of e-negotiations: protocols and strategies [13]. Protocols define the rules of interaction between negotiation agents (participants) and the sequences of allowed offers. In general, agents must agree on the protocol before negotiation begins. Strategies are action plans for the negotiation agents to follow in evaluating offers and formulating counteroffers. Usually there are several strategies for a particular protocol, and each one may produce a different outcome.

E-negotiations are further divided into unsupported, supported, and automated [1] processes. In unsupported e-negotiations, the participants control and manage all tasks without support or advice from a system. Supported e-negotiations involve the help of a system for decision making. Automated e-negotiations involve software agents that make decisions based on negotiation strategies and tactics provided by their owner.

Finally, at the system level, e-negotiations are divided into Negotiation Support Systems (NSS), Negotiation Software Agents (NSA), and e-negotiation servers. 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. NSA are AI (artificial intelligence) enabled software entities which communicate with other entities and make decisions on behalf of their owner [7]. NSA's main concerns are about software agents' strategies and performance [14]. E-negotiation servers are software systems that implement a negotiation protocol and provide a platform for participants to interact. In the literature, negotiation servers are also referred to as negotiation media, platforms, or negotiation-enabled e-marketplaces.

4 Architecture and Implementation

Web service orchestration is a way of composing and coordinating web services to obtain higher-level business processes. It describes how web services interact with each other at the message level and tracks the sequence of messages including the business logic and execution order of the interactions [15].

There are standards available to orchestrate web services such as BPEL (Business Process Execution Language) [16]. BPEL is designed to address the orchestration complexity, thereby reducing time-to-market and costs and increasing the overall efficiency and accuracy of business processes. It 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 operation logic sequences. BPEL supports common repetition (while-loop), selection (if-then-else, select-case), error handling (try-catch), parallel processing, and Java embedding. As a widely adopted orchestration language providing the necessary business logic to compose complete running processes, we decided to use BPEL to model and enact online auction processes.

4.1 Server Design

The e-negotiation server is where the online auction process is created and executed. It can be deployed either by one of the participants or by a third party provider (i.e., facilitator). We believe a third party deployment is better because it provides for a common ontology and interfaces to be shared by participants as well as transparent, optimal and efficient processes. The server's conceptual architecture is shown in Fig. 1.

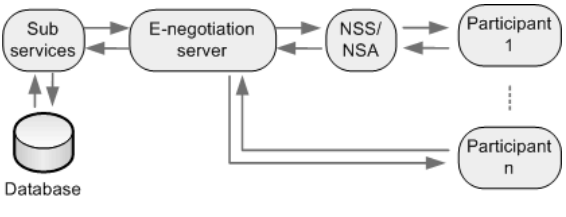

Fig. 1. Conceptual architecture of the e-negotiation server

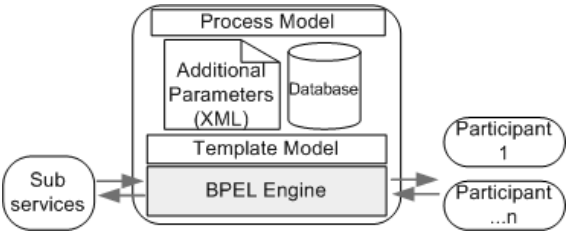

Fig. 2. Internal design of the e-negotiation server

Table 1. XML configuration files

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

As suggested in [3], there are several tasks common to all auction scenarios. These are implemented as the following services (sub services in Fig. 1): registration and authentication; posting; bidding; information retrieval, history log, and gateway services which coordinate all the services. Participants have the option to interact with the server directly or through an NSS or NSA. We designed the e-negotiation server and the participants (sellers and buyers) as web services. To orchestrate these web services we use the BPEL Process Manager. We visually model the auction using JDeveloper [17]. The resulting model is compiled and deployed on the BPEL engine (see Fig. 2). To avoid an eventual recompilation and redeployment of the model after a minor change, we transfer additional parameters (bid visibility, idle time, auction deadline, etc.) out of the business logic of the auction process and into an external XML file. The template model in Fig. 2 is a collection of classes which bind together the BPEL process model, the XML configuration file, and the database. This is achieved using the `java embedding` component. The auction model is implemented in the BPEL process flow, which will be discussed in Section 5. The XML configuration files for the English and Dutch auctions are shown in Table 1. The BPEL engine is the core component of the e-negotiation server where the execution of communication level operations with the sub services and the participants takes place. The participants access all services offered by the server through the BPEL engine.

4.2 Client Design

We consider two types of participants: one who uses the provided interface (*manual negotiation client*), and one who builds an interface and connects it to the

e-negotiation server for automation purposes and for employing AI techniques such as learning and using 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 server and join the negotiation process by accessing the BPEL engine and using the WSDL file. They can call the services offered by the server using its operation name and passing the necessary variables. The gateway service will ensure that all clients have valid identification and proper access through every phase of the negotiation process.

The web service discovery does not need to be done through UDDI; instead, it can be done through a web page that gives the location of the WSDL file and explains how to utilize the web service. The automated negotiation clients can process the information they get from the e-negotiation server and pass it to their NSS or NSA using a SOAP message tunneling mechanism.

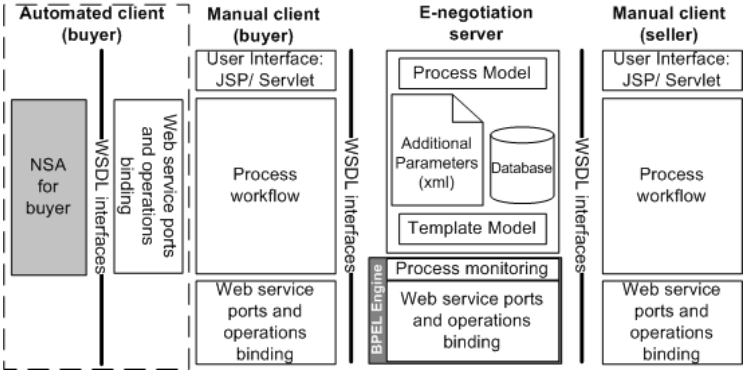

Fig. 3. Manual and automated clients

Fig. 3 shows the manual and automated clients. 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 connects to the server using the same web service ports and operations interface; however, the automated client passes the information from the server to the NSA. The NSA is implemented as an independent component offering various strategy-oriented web services. This approach increases the reusability of the NSA. The server's process model in Fig. 3 is the BPEL process mentioned earlier. The `java embedding` components in the BPEL process access specific negotiation attributes (XML file) and use the database.

To simplify the connection between the BPEL engine and the clients, we use the *Façade* design pattern to `get` and `set` variables in the BPEL engine, and to `invoke` or `receive` other web services (see Fig. 4). We also use the *Model-View-Controller design pattern*. The *model* holds the logic of the process such as handling incoming or outgoing messages, adding AI capabilities to the system, forwarding information further to a customized NSA, and giving instructions to the Façade object. 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 (see Fig. 4).

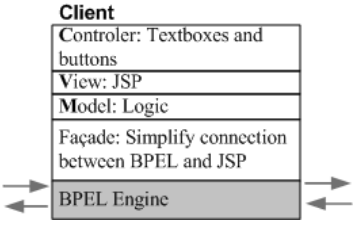

Fig. 4. Client's user interface

5 Modeling

Kumar and Feldman [3] proposed templates for modeling auction processes using Finite State Machine (FSM) diagrams. By omitting details such as authentication from the process, FSMs concentrate on the interactions between states and participants. We use these FSMs as general guidelines in our modeling. Fig. 5 shows the FSM of the English open cry and sealed bid auctions. It starts with a Deal Template (DT) state, followed by an `OfferToSell` from the seller leading to the Offer state. In an open cry auction, a bid from a buyer produces a message notifying all buyers that the best bid has changed; in a sealed-bid auction the best bid remains secret. Two other events can occur in the Offer state: the seller closes the auction resulting in the Negotiation Aborted (NA) state, or the auction ends successfully in the Deal (D) state.

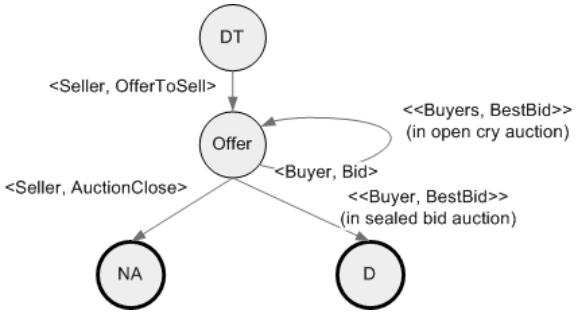

Fig. 5. FSM for the English open cry and sealed bid auctions [3]

Due to the detailed nature of auction processes and the large sizes of the models we developed, showing them in their original form (i.e., screenshots) would sacrifice their details and sharpness. Thus we use an activity diagram notation that provides the same functionalities as BPEL but which is more readable.

Fig. 6 presents the BPEL process description of the English auction. In the diagram, we use the `scope-and-expand` method. A `scope` is a collection of activities represented by a plus sign "(+)" that can be *expanded* into a subsequent activity diagram. We assume that all participants have completed the registration process before the start of the auction. The registration process does not appear in the diagram.

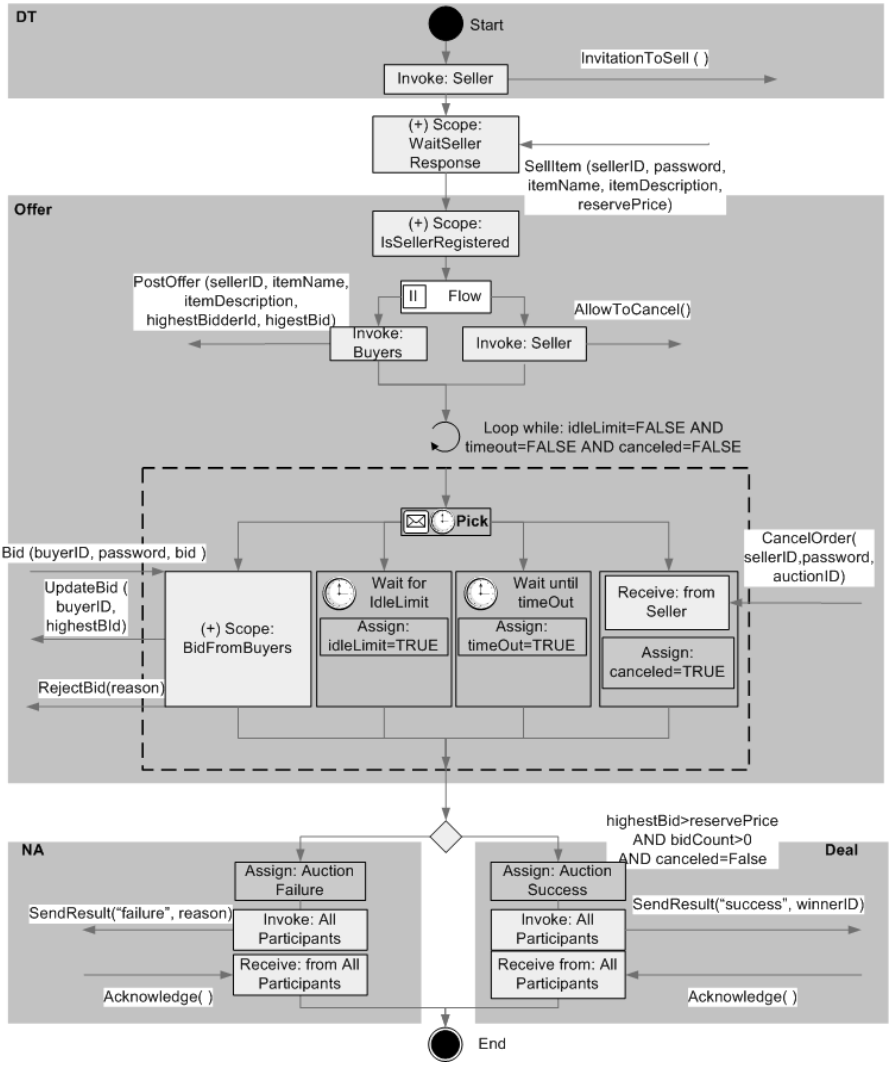

Fig. 6. BPEL diagram for the English auction

The "`Scope`: WaitSellerResponse" is a collection of activities that `wait` for a response from a seller for a certain amount of time, ensuring that he/she has enough time to respond. The expansion of this `scope` is shown in Fig. 7-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 does not proceed (Fig. 7-b). Otherwise, the server executes a parallel `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 so that buyers can start bidding on it. The bidding phase is a loop mechanism. In the English auction FSM (Fig. 5), the activities that can happen during the bidding phase are: *buyer bids* (back to loop); *seller cancels* (out of loop); and

auction reaches its deadline (out of loop). To replicate the "going-going-gone" found in offline auctions, we introduced another variable into this phase: *idle time*. If there is no incoming bid after a certain amount of time, the bidding phase ends (out of loop). These four activities are guarded by a `pick` construct. The server checks for the authenticity of bidders and verifies the validity of their bids (Fig. 7-c).

At the end of the bidding phase, a `switch` construct will decide whether the auction was 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 inform participants of the result of the auction.

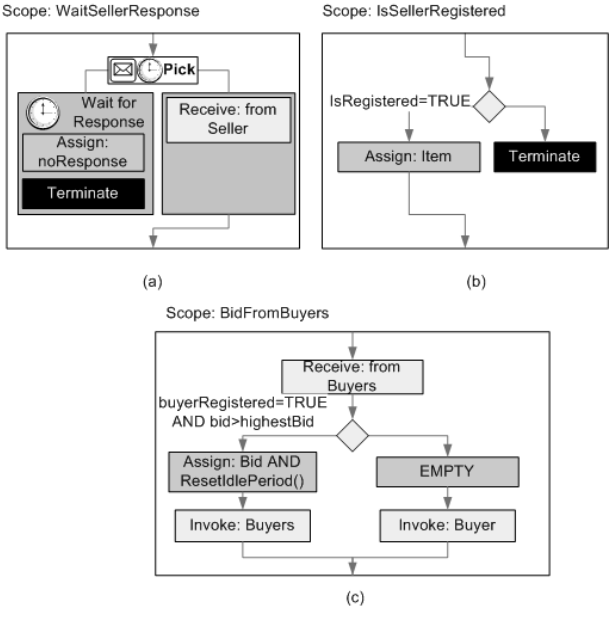

Fig. 7. English auction subsequent BPEL diagrams

Finally, it should be noted that we deployed and tested several auction scenarios, but there is not enough space in this paper to report on all of them. The English auction gives a good idea on the expressive power of BPEL and web service orchestration languages in general for modeling and enacting auction scenarios.

6 Discussion

The formalisms used to describe auction processes usually capture their most common components. In real-life situations the complexity of the auction process may vary, but the basic concepts remain the same. BPEL process flow complexity increases as we deal with sub processes such as authentication, validation of input, and additional negotiation parameters (bid visibility, item quantity, etc.).

By looking at the process diagrams of various auctions, we identified the following similarities which may constitute a basis for a repository of reusable templates.

(1) The Deal Template (DT) state (Fig. 6) consists of specific negotiation rules which are agreed upon before the process starts. Since BPEL stands on top of WSDL, it is easy to manage the message interactions. Every incoming or outgoing message format has to conform to the WSDL description. In all our BPEL process diagrams, the DT is always marked with a void invitation or initiation from the server; the seller has to reply with an offer.

(2) The Offer state (Fig. 6) might vary slightly between different auction protocols. However, there are unique sequences present in all our BPEL process diagrams. The sequences are: `receive` which gets the offer from the seller; `flow` which `invokes` participants and allows them to reply by sending a message to the server; and `pick` which determines what events arrive first. Before the `pick` construct there is a `while` which creates a repetitive cycle for the bidding phase.

(3) Mechanisms in `scope-and-expand` activities (Fig. 7) can act as reusable components for different auction protocols. At the 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 [18]. But IBM and SAP are working on a sub-process extension for BPEL (BPEL-SPE) [19]. 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 "IsSellerRegistered", and the 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 raising a security concern. One solution could be to filter the caller's IP address as a precaution. Second, we can implement the reusable components as java classes. In the BPEL process flow diagram, we can define a `java embedding` component which will replace the `scope-and-expand`. But since this approach would defeat the purpose of visual modeling which we seek in our framework; we prefer the first approach. Additionally, unlike BPEL-SPE, 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.

7 Related Work

Kim and Segev (2005) proposed an attribute-based negotiation process composer [7], which is a system that enables the negotiation designer to generate a set of BPEL constructs. These constructs are useful in creating a new e-negotiation marketplace and orchestrating its interactions in a web service environment. The system's main function is to customize, generate, and validate a series of constructs for BPEL processes. There is no user interface, such as dynamic web pages that can access the web service for the participant. After the generation of the BPEL constructs, they still need to be complied and deployed on the BPEL engine. This can become time consuming when slight modifications are introduced. In our framework, we prefer to model the

auction scenarios and store the attributes in an external XML configuration file; this way we avoid re-compilation and re-deployment after minor changes.

Rolli and Eberhart (2005) proposed a model to describe and run Internet-based auctions [20]. The model presents three layers: auction data, auction mechanism, and auction participants. The authors implemented a prototype using Collaxa's BPEL4WS modeling editor. This editor is now part of Oracle BPEL PM [21]. The produced models are compiled using BPEL2Java [22] then executed within a Java environment. BPEL was used for modeling the general negotiation activities, and the modeling was completed with Java code, which is not convenient for the negotiation designer. Interactions between the platform and the participants are carried out using Java-RMI [23]. Our approach is more user-friendly and takes advantage of the modeling paradigm.

8 Conclusion

This paper presented a service oriented online auction framework that is generic enough to support various auction protocols. To create an online auction, designers visually model the process using a web service orchestration language such as BPEL, possibly reusing existing components. The approach reduces the need for designers to understand programming concepts (at least for the high level design), allowing them to focus on the business view. Our framework promotes loose coupling and reusability. It is easy to extend the system with additional components such as an NSA and still use the same ports and interfaces. With standard open interfaces such as web services, every service is platform independent. To our knowledge, this research is the first to thoroughly test the use of web service orchestration for hosting online auctions.

There are limitations inherent to web service orchestration languages. Unlike other programming languages, BPEL is still growing. The real purpose of BPEL is to model and orchestrate web services; therefore its expressiveness is not as effective as a real programming language. For instance, there is only one starting point and one exit point in a BPEL process, which makes for one monolithic process. Furthermore, although BPEL provides java embedding, its Java Runtime Environment (JRE) still lacks compatibility with existing java implementations. We did not address or implement any form of security, this issue being beyond the objectives of this paper.

This research opened new opportunities for studying the behavior of software agents when provided with bidding strategies. We are using an implementation of our auction framework to conduct bidding tournaments between software agents. The separation of the auction description (the model) from the server is enabling us to be more efficient in deploying and studying new auction scenarios.

References

1. Bichler, M., Kersten, G., Strecker, S.: Towards a Structured Design of Electronic Negotiations. Group Decision and Negotiation 12, 311–335 (2003)
2. Hurwitz.com, Negotiated Trade: the Next Frontier for B2B e-commerce. Tech Report (2000)

3. Kumar, M., Feldman, S.I.: 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, July 5-6, 2004, pp. 5–6 (2004)
5. Pringadi, R., Benyoucef, M.: Web Service Orchestration of E-negotiation Interactions. Working Paper #wp06-28, School of Management, University of Ottawa (2006)
6. Benyoucef, M., et al.: Towards a Generic E-Negotiation Platform. In: Kropf, P.G., Babin, G., Plaice, J., Unger, H. (eds.) DCW 2000. LNCS, vol. 1830, pp. 95–109. Springer, Heidelberg (2000)
7. Kim, J.B., Segev, A.: A Web Services-enabled Marketplace Architecture for Negotiation Process Management. Decision Support Systems 40, 71–87 (2005)
8. Mathieu, P., Verrons, M.-H.: ANTS: an API for creating negotiation applications. In: 10th ISPE International conference on concurrent engineering: research and application, Madeira Island - Portugal (July 26-30, 2003)
9. Basu, A., Kumar, A.: Research commentary: workflow management issues in e-business. Information Systems Research 13, 1–14 (2002)
10. Bartolini, C., Preist, C., Jennings, N.R.: A Generic Software Framework for Automated Negotiation. Trusted E-Services Laboratory - HP Laboratories Bristol (2002)
11. Weinhardt, C., Gomber, P.: Agent-Mediated Off-Exchange Trading. In: Proceedings of the 32nd Hawaii Conf. on System Sciences, Maui, Hawaii, January 5-8, 1999, pp. 6–9 (1999)
12. eBay.com. eBay Announces First Quarter, Financial Results - 19 April. 2006 [cited August 2006] (2006), Available from http://investor.ebay.com/releases.cfm?FYear=2006
13. Lomuscio, A.R., Wooldridge, M., Jennings, N.R.: A Classification Scheme for Negotiation in Electronic Commerce. Group Decision and Negotiation 12(12), 31–56 (2003)
14. Rust, J., Miller, J., Palmer, R.: Behavior of trading automata in a computerized double auction market. In: The Double Auction Market: Institutions, Theories, and Evidence, pp. 153–196. Addison Wesley, Reading (1993)
15. Peltz, C.: Web Services Orchestration. Hewlett-Packard Company (2003)
16. IBM, et al.: Business Process Execution Language for Web Services version 1.1, [cited August 2005] (2002), Available from www.ibm.com/developerworks/library/ws-bpel
17. Oracle.com. Oracle JDeveloper 10g. [cited 2006] (2005), Available from: www.oracle.com/technology/products/jdev/index.html
18. Trickovic, I.: Modularization and reuse in WS-BPEL. SAP Developer Network - SDN Community Contribution Whitepaper (October 2005)
19. Kloppmann, M., et al.: WS-BPEL Extension for Sub-processes - BPEL-SPE. SAP Developer Network Whitepaper (September 2005)
20. Rolli, D., Eberhart, A.: An Auction Reference Model for Describing and Running Auctions. Wirtschaftsinformatik, pp. 289–308 (2005)
21. Boulton, C.: Oracle Goes SOA with Collaxa Buy 2004 [cited 2005 August 15] (2004) Available from: http://www.internetnews.com/bus-news/article.php/3374851
22. eclipse.org. BPEL to Java (B2J) Subproject. [cited 24 August 2006] Available from: http://www.eclipse.org/stp/b2j/
23. sun.com. Java Remote Method Invocation (Java RMI). [cited 24 August 2006] (2006), Available from http://java.sun.com/products/jdk/rmi/