Revised version appeared in:

Proceedings, Enterprise Modelling and IS Architectures
October 24-25, 2005, Klagenfurt, Austria

A Model-Driven Approach for the Rapid Development of E-Negotiation Systems ♦

Morad Benyoucef and Stefanie Rinderle[1]

University of Ottawa
136 Jean-Jacques Lussier St.
Ottawa, Ontario K1N 6N5, Canada

Abstract

Most of today's e-marketplaces support a single negotiation protocol. The protocol is usually built into the e-marketplace infrastructure, therefore if a new protocol is introduced then a time consuming and complex process of implementing it takes place. Moreover, participants in the e-marketplace need to adapt their interfaces to the new protocol, especially if they make use of automated means such as software agents to interact with the e-marketplace. This paper reports on a model-driven approach and a framework for rapid and user-friendly development of configurable e-marketplaces and automated e-negotiation systems. A designer on the e-marketplace specifies negotiation protocols using Statecharts and feeds them to a mapping system that transforms them into web service orchestrations. Participants use automated negotiation systems to interact with the e-marketplace. An automated negotiation entity capable of interacting with the e-marketplace is generated based on the negotiation protocol implemented on the e-marketplace. The automated negotiation entity is provided with negotiation strategies and tactics specified in a declarative format. We propose a mapping algorithm to transform Statechart models of negotiation protocols into web service orchestrations and we report on the implementation our framework.

♦ This work has been partially supported by the Natural Sciences and Engineering Research Council, Canada and the Social Sciences and Humanities Research Council, Canada.

[1] Stefanie Rinderle is from University of Ulm, Germany. This work was conducted during her post-doctoral stay at the University of Ottawa, Canada

1. Introduction

The Object Management Group (OMG) defines negotiation as "mechanisms that allow a recursive interaction between a principal and a respondent in the resolution of a good deal" [OMG99]. Usually, the principal and the respondent are a buyer and a seller who are set to negotiate the price, the delivery date, the conditions of the purchase, the terms of the guarantee, etc. The principal and the respondent can be individuals or organizations. Negotiation is vital in establishing business-to-business (B2B) relationships and, to a lesser extent, in facilitating consumer-to-consumer (C2C) commercial interactions. Indeed, a report by the Hurwitz Group claims that "80% of commerce is performed through negotiated trade" [Hur00]. Today, most of this negotiated trade takes place electronically, facilitated by e-commerce - i.e., the process of buying and selling goods and services electronically; and by e-business - i.e., the use of the Internet and information technology (IT) to execute all business processes in the enterprise, including e-commerce. By integrating their IT infrastructures with those of their partners, traditional businesses are moving closer towards becoming real e-businesses. We believe that IT supported negotiation is a cornerstone in that integration.

Electronic negotiation (e-negotiation) takes place when the negotiating function is performed through electronic means. We talk of fully automated e-negotiation when all parties involved are software agents, semi-automated e-negotiation when a human negotiates with a software agent, and manual e-negotiation when all parties are human [BSS96]. The interest in e-negotiation is motivated by its potential to provide business partners with more efficient processes, enabling them to arrive at more satisfying agreements in less time.

The most basic form of e-negotiation is the fixed price sale where an online seller offers goods or services at "take-it-or-leave-it" prices then buyers decide whether or not to make a purchase at the posted price. Auctions are a bit more complex and they are currently the most visible category of e-negotiations. Online auctions can reach a large and physically distributed audience at reduced cost, whereas offline auctions tend to cost more and require that the participants gather in one physical location. E-negotiations can take an even more complex form called bargaining. This involves making proposals and counter-proposals until an agreement is reached [San99]. Bargaining can be bilateral or multi-lateral, depending on whether there are two parties or many parties involved in the negotiation [OMG99]. If the object of the negotiation has more than one negotiable attribute (e.g., price, quality, and delivery date) then we talk of multi-attribute e-negotiations.

The negotiation process is manually intensive therefore using e-negotiation systems to automate it can reduce the costs associated with the negotiation [Hur00]. This is why the interest in designing these systems has focused on achieving higher efficiency and lower transaction costs [Mal87]. We distinguish three categories of e-negotiation systems [BKS03]: (1) *negotiation support systems* assist users with communication and decision-making activities; (2) *negotiation software agents* replace users in their communication and decision-making activities; and (3) *e-negotiation media* provide a platform that implements a negotiation protocol.

There are two categories of e-negotiation media: *servers* which implement multiple protocols, and *applications* which implement a single protocol. Traditionally, applications have dominated negotiation design, but lately, the importance of servers has increased, and a need for configurable servers is being felt [NBBV03]. Attempts were made to design configurable e-negotiation media to support more than one negotiation protocol. They were partially successful, but they were designed in an ad-hoc manner. Some of these attempts were: the AuctionBot [WWW98] which supports the configuration of various auctions; GNP [BKL+00] which separates auction specifications from the

logic of the server, and eAuctionHouse [Eau02] which allows for the configuration of auctions with the help of an expert system. Recently, Kersten et al. [KLS04] designed a configurable negotiation server that supports bargaining, based on a process model which organizes negotiation activities into phases; and a set of rules that govern the processing, decision-making, and communication. The main problem in designing e-negotiation media is the lack of a systematic approach. Indeed, to this day, design has been a trial-and-error process.

This paper presents a model-driven approach for rapid and user-friendly development of e-negotiation systems. We consider two categories of e-negotiation systems: “e-negotiation media” and “negotiation software agents”. We propose a new framework for configurable e-negotiation systems in which “e-negotiation media” is the electronic marketplace (e-marketplace) where human and software participants meet to negotiate deals. Automated negotiation systems provide a framework for the existence of “negotiation software agents” and serve as the interface between the negotiator and the e-marketplace. The e-marketplace enforces negotiation protocols and therefore should make these protocols available for consultation and automation purposes. Separating the protocols from the e-negotiation media is a first step towards a configurable e-marketplace. Separating negotiation strategies from protocols also brings flexibility to the design of automated negotiation systems. Clearly, the design of e-marketplaces has a direct effect on the design of automated negotiation systems.

The first objective of this paper is to propose a service oriented framework for e-negotiation systems. The two main components of such framework are an e-marketplace and an automated negotiation system. The framework enables a designer on the e-marketplace to specify negotiation protocols and feed the resulting specification to a mapping system that transforms them into executable processes described using a web services orchestration language. The novelty here lies in the separation of the protocols from the e-negotiation media as well as in the mapping algorithm. Since no protocol exists that can fit the needs of all participants, the e-marketplace must be able to implement any new negotiation protocol with minimum time and effort. The automated negotiation system is also based on the separation of the negotiation protocols and strategies from the system. Based on the negotiation protocol implemented on the e-marketplace, an agent factory generates the component of the system that automates the exchange between the participant and the e-marketplace. A designer on the participant’s side can specify negotiation strategies and tactics using a declarative format. The second objective is to propose a mapping algorithm that transforms Statechart models of negotiation protocols into processes described using a web services orchestration language. The third objective is to report on our current implementation of the framework.

The paper is organized as follows: In Section 2 we discuss the use of Statecharts to specify negotiation protocols. In Section 3 we detail our service oriented e-negotiation framework. Section 4 describes the mapping of Statechart descriptions of negotiation protocols into processes described using a web services orchestration language. In Section 5 we report on the current implementation of the framework and provide an example. In Section 6 we discuss some related work, and in Section 7 we conclude and discuss the perspectives and future work.

2. A Generic Statechart template for business transactions

Based on a requirements analysis for modeling e-negotiations, Statecharts [Hare87] qualify as an adequate formalism. Statecharts have a good formal basis and can be serialized, visualized, and executed. They are well established, easy to understand, complete, and can be converted into other formalisms. Statecharts enable the designer of negotiation protocols to understand, implement and test

different negotiation protocols more thoroughly before making them available for general use (with or without software support). For more details on the use of Statecharts to specify negotiation protocols refer to [RiBe05].

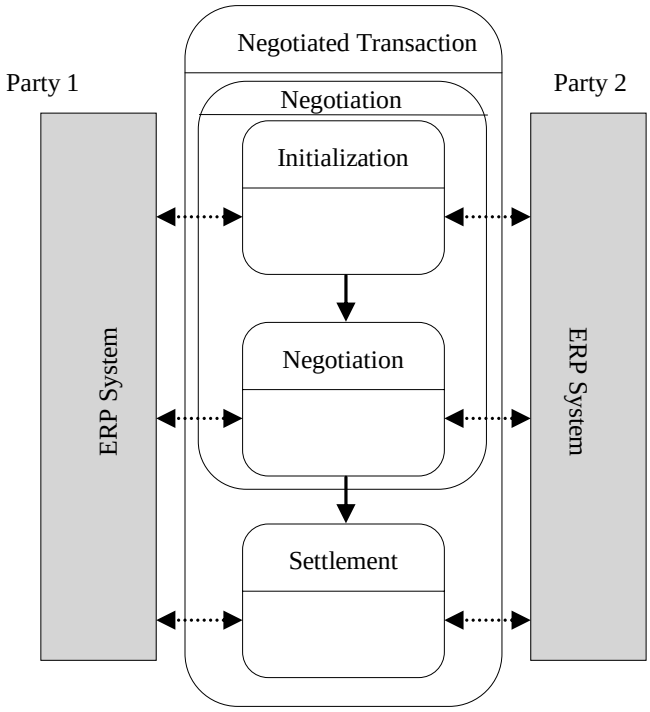

Figure 1: Generic Template for Business Transactions

As shown in

Figure **1**, a negotiated transaction between two or more parties usually goes through three distinct states. In the initialization state participating parties register on the e-marketplace and the e-marketplace performs the necessary checks on the parties.

The initialization can be as simple as signaling one party's intent to participate in the transaction, in case the e-marketplace is set up for a closed circle of registered or invited members. If the e-marketplace is open, then the initialization can be as complex as one party completing rigorous registration procedures, and the e-marketplace performing the necessary verifications such as credit checks. If electronic links exist between the e-marketplace and the ERP systems of the participating parties, the initialization can be greatly facilitated by granting protected and restricted access to each other's internal databases.

The negotiation state is where the parties negotiate by exchanging messages (offers, counter offers, etc.) according to the precise protocol of the negotiation implemented on the e-marketplace. A negotiating party can require an electronic link to its own ERP system to automatically check stock levels, manufacturing schedules, etc. The link makes it easier and quicker to respond to offers made by the other parties and to formulate counter offers.

A successful negotiation leads to the drafting of a contract that needs to be executed by the agreeing parties. This is the settlement state. If a certain level of contract automation is achieved, then electronic links to the parties' ERP systems are necessary to carry out certain transactions such as sending

purchase orders, receiving payments, etc.

Example: Here we present the Dutch auction as an example of e-negotiation protocols. Dutch auctions are often used to sell perishable goods such as tulips or airplane seats where the seller starts with a certain price and gradually decreases it until a buyer signals his/her interest, in which case a deal is concluded [KuFe98a]. The Statechart depicted in Figure 2 reflects a Dutch auction for an arbitrary number of items (i.e., buyers can specify how many items they will purchase at the current price).

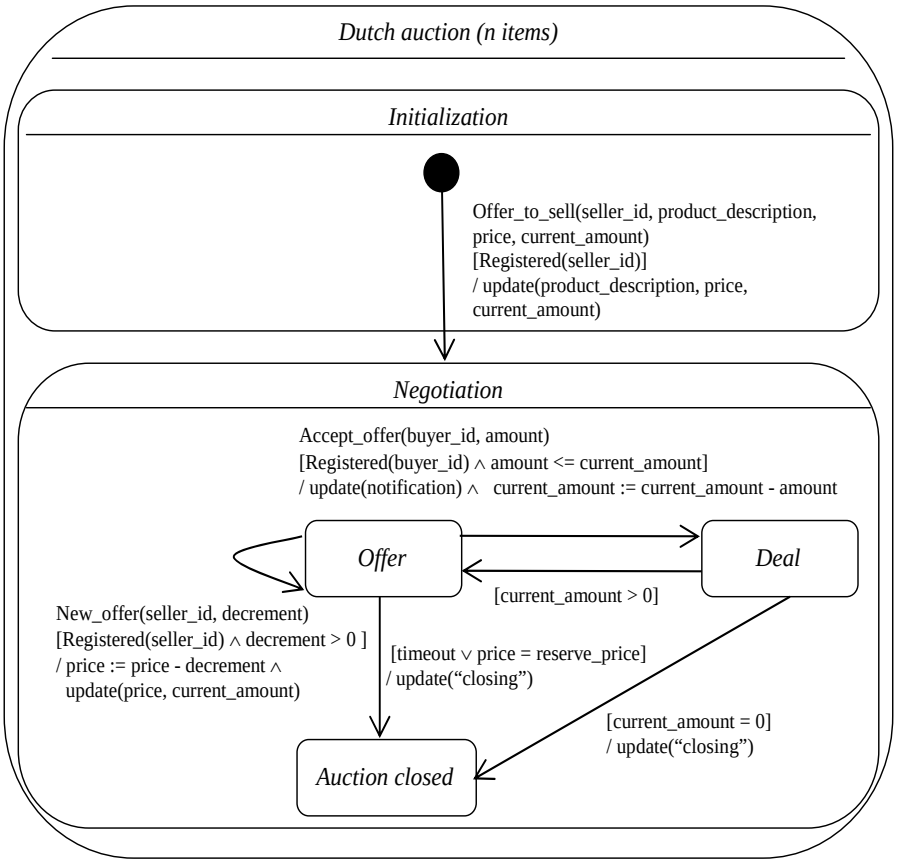

Figure 2: Statechart for the Dutch Auction

3. Service-oriented e-negotiation framework

Businesses are rapidly moving towards exposing their services on the web (giving way to web services), hoping to interact more efficiently with their partners and to achieve higher levels of automation at lower cost. For that reason our framework is based on a service oriented architecture (SOA). We believe that web services are appropriate for deploying e-negotiation systems because: (1) relationships between negotiating partners are dynamic; (2) negotiation is part of procurement therefore interoperability between internal and external IT systems is important; and (3) web services provide a standardized and flexible integration technology that no organization can afford to ignore if it wants to interact with its partners [KiSe05]. Simply put, web services provide the means for software components to communicate with each other on the web using XML. A web service describes itself (using WSDL), can be located (using UDDI), and invoked (using SOAP).

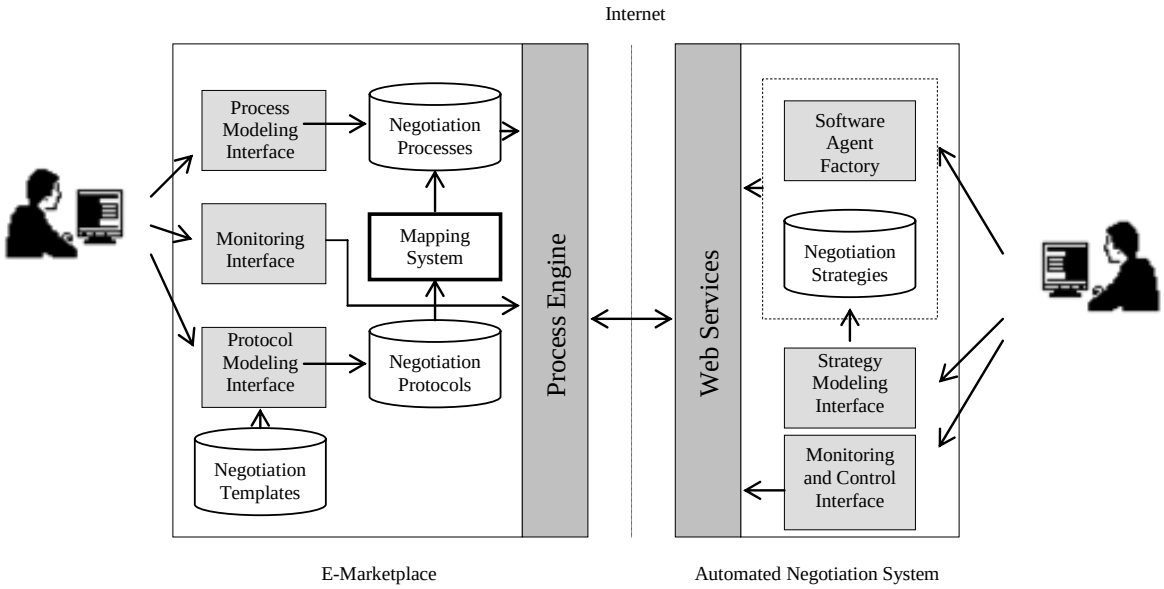

Figure 3: Service Oriented e-Negotiation Framework

It is important to remember that the e-marketplace is deployed by a negotiation facilitator (usually a third party) and its main role is to implement a negotiation protocol. An automated negotiation system, on the other hand, is deployed by participants in the negotiation (i.e., negotiators). It can be seen as the interface between the negotiator's internal IT systems (mainly the ERP system) and the e-marketplace. The framework presented in Figure 3 can be interpreted as follows.

The e-marketplace: a designer on the e-marketplace side uses the *Protocol Modeling Interface* to design *Negotiation Protocols*. These protocols are usually represented using a formal specification such as Statecharts. The designer can also use *Negotiation Templates* available on the e-marketplace, and eventually modify them into negotiation protocols that answer the needs of the moment. A *Mapping System* will automatically convert the protocols into *Negotiation Processes* ready to be executed on the e-marketplace. A web services orchestration language such as BPEL4WS [ACD+03] will be used to represent the processes. The designer can choose to directly design the processes using a *Process Modeling Interface* which is a web services orchestration authoring tool. A *Monitoring Interface* will be used to monitor the execution of the negotiation process. The process engine runs the process.

The Automated Negotiation System: In manual or automated negotiation, we have to distinguish between negotiation protocols which are the rules of the game (e.g., how the exchange of offers and counter offers takes place between the participants) and the negotiation strategies which are used by the participants to maximize their benefit (and to minimize their loss). Negotiation tactics are the small steps taken by participants towards achieving their strategies. Protocols are made public to every participant, but strategies and tactics are kept secret. Nothing keeps a participant from trying to discover its opponent's strategies and tactics by observing the opponent's present and past behavior. In the framework of Figure 3 a designer on the participant's side uses a *Strategy Modeling Interface* to design *Negotiation Strategies*. Based on the negotiation protocol implemented on the e-marketplace, a *Software Agent Factory* generates the components of the system that automates the exchange mechanism between the participant and the e-marketplace. A *Monitoring and Control Interface* is used to monitor and control the behavior of the automated negotiation system. The Web Services

component groups all the interactions of the Automated Negotiation System and makes them available in the form of web services.

4. On mapping negotiation protocols to web service orchestrations

In this section we provide an algorithm for mapping Statechart models of e-negotiation protocols to processes described within a web service orchestration language.

4.1 Statecharts

One particularity of our service oriented e-negotiation framework is the fact that it is based on the separation of negotiation protocols from the e-negotiation media (i.e., the implementation of the e-marketplace). This separation is achieved by using Statecharts to specify the protocols. Based on [JaSu04] we define a Statechart as a tuple ST= (S, T, E, C, A, $s_{initial}$, s_{final}) where

- S is a finite set of states
- E is a finite set of events
- C is a finite set of conditions
- A is a finite set of actions
- $T \subseteq S \times E \times C \times 2^A \times S$ is a finite set of transitions where 2^A denotes the power set of A
- $s_{initial} \in S$ is the (unique) initial state (i.e., $s_{initial}$ has no incoming transitions)
- $s_{final} \in S$ is a final state (s_{final} has no outgoing transitions)

From different case studies we derived several assumptions which hold for the Statechart models of e-negotiation protocols. Firstly, the event and action parts of the transitions comprise message sending and receiving events, i.e., the e-marketplace either waits for an incoming message sent by one of the participants or the e-marketplace itself sends a message to one or more of the participants.[2] Secondly, we assume that the Statechart models are hierarchically decomposed (i.e., except for concurrent flow super-states there are no hierarchical states). For example, super-state *Auction closed* (cf. Figure 2) can be decomposed and removed without losing basic information. As mentioned, one exception is the use of concurrent execution where sub-states may be ordered in parallel but must not be hierarchically decomposed themselves. Thirdly, the use of cyclic structures is restricted to loops with length less than three (i.e., only short loops of length one and mutual calls between two states of length two are allowed).

The above assumptions can be formalized as follows:

Let *EType* be the set of all possible event types and let *AType* be the set of all possible action types. Let further *eType: E* → *EType* be the function which maps events to their specific event type and *aType: A* → *AType* the function which maps actions to their specific action types. Then:

$\forall$ $e \in E$: eType(e) = msg_receiving

[2] In addition, the action part of a transition may include the assignment of process variable values.

- ∀ e ∈ E: e = rec_message(sender, [paramList]) where sender is one of the e-marketplace participants

- ∀ a ∈ A: aType(a) ∈ {msg_sending, assign_values}

- ∀ a ∈ A with aType(a) = msg_sending:

  a = end_id(sender, [paramList]) where sender is one of the participants the e-marketplace

- no loops with lengths > 2: Let ST = (S, T, E, C, A, i, f) and ST' = (S, T', E, C, A, i, f) be Statecharts where ST' is obtained by reducing transition set T of ST in the following way:

  - if ∃ s ∈ S with ∃ (s, e, c, a, s) ∈ T: T' := T \ {(s, e, c, a, s)}

  - if ∃ s, s' ∈ S with ∃ (s, e, c, a, s'), (s', e', c', a', s) ∈ T:

    T' := T \ {(s', e', c', a', s)}

    Then ST' has to be acyclic.

- ST is hierarchically decomposed except for concurrent execution
- Concurrent execution contains no hierarchically decomposed sub-states
- ∃ e ∈ E: eType(e) = msg_receiving with ∃ T = (i, e, c, a, s), i.e., there is a transition condition containing a message receiving event[3].

4.2 Web Service Orchestration Language (BPEL4WS)

As a web service orchestration language we use a subset of BPEL4WS. Within this paper a BPEL4WS process P is defined as a tuple P = (A, AT, S, ST, V, VL, PT, PL, i) where

- A is the set of simple activities
- AT is the function which assigns to each simple activity its particular type, i.e.:

  AT: A → {<assign>, <empty>, <terminate>, <receive>, <invoke>, <pick>}
- S is the set of structured activities
- ST is the function which assigns to each structured activity its particular type, i.e.:

  ST: S → {<sequence> … </sequence>, <switch> … </switch>,

  <flow> … </flow>, <while> … </while>}
- V is the set of variables
- VL ⊆ (A × V) ∪ (V × A) is the set of read / write accesses linking activities to variables and vice versa

[3] This is important in order to generate the first `receive` activity within the process that drives the e-marketplace which initiates a new process instance.

- PT is the set of port types
- PL is the set of partner links
- i is the initial state of P.

4.3 Pattern-wise mapping

In general, finding a complete mapping from Statecharts to a web service orchestration language such as BPEL4WS is a difficult task. Some approaches face this challenge in order to provide a model-driven development of web services [BBCT04, BDS05]. However, these approaches are based on some restrictions and assumptions (e.g., restricting transitions conditions to the conditional part). In our approach we start from a different direction by exploiting the assumptions which result from the specificity of Statechart models of e-negotiation protocols (cf. Section 4.1). Furthermore, we exploit the idea of identifying commonly used *patterns* within the Statechart models (comparable to workflow patterns in the process management domain [AHKB03]). For all of these Statechart patterns the corresponding BPEL4WS patterns have been elaborated. In the following we present and describe a selection of the most important patterns. We provide an algorithm that constructs the complete BPEL4WS process based on these patterns in Section 4.4.

Figure 4 depicts the Statechart pattern for a sequence of states `S1` and `S2` connected by transition `(S1, msg1(s1, pL1), cond1, msg2(s2, pL2), S2)`. The associated BPEL4WS pattern is shown as sub-pattern `SP1_2`: the e-marketplace waits for receiving message `msg1` from sender `s1`. If `msg1` is received and condition `cond1` is true then the e-marketplace invokes the associated operation within the port type connected with sender `s2` by sending message `msg2`.

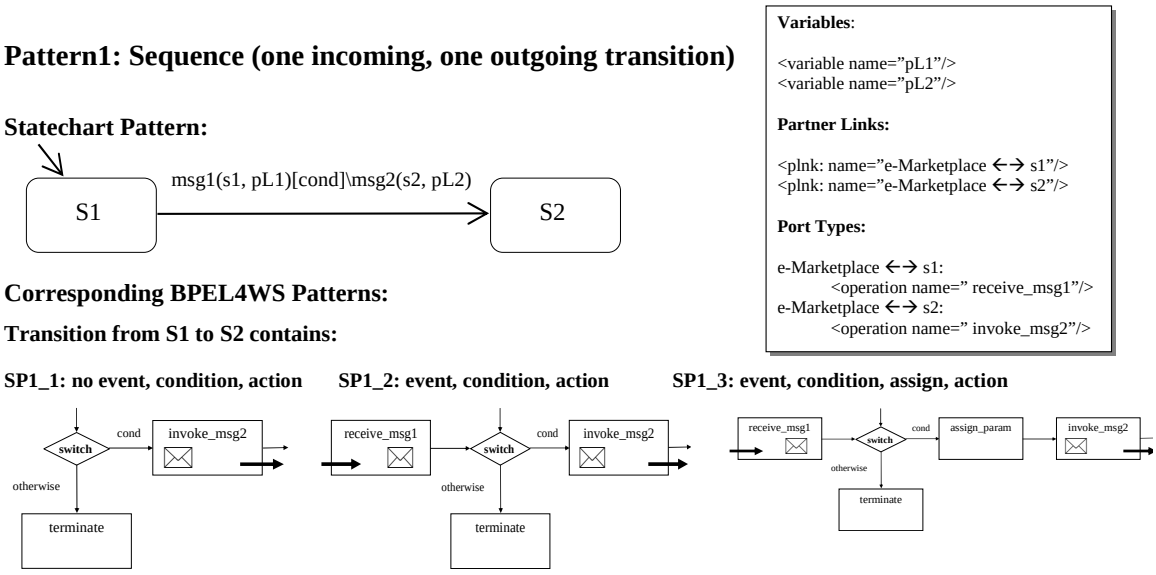

Figure 4: Sequence with Different Sub-Patterns

The parameter lists of message `msg1` and `msg2` constitute the variables of the e-Marketplace. Note that a more precise definition of the variables is conceivable by splitting the parameter lists into single parameters and specifying the associated variables of the e-marketplace. The data flow is specified

within the implementation of the receive and invoke operations (e.g., operation `invoke_msg2` has input variable `pL1` and output variable `pL2`). There may be also variations like sub-pattern `SP1_1` (cf. Figure 4) depending on whether the event or the condition part of transition `(S1, msg1(s1, pL1), cond1, msg2(s2, pL2), S2)` is specified or not. Sub-pattern `SP1_3` reflects the case where, within the action part of the transition, not only a message is sent but also a certain value is assigned to a particular process variable.

When mapping a Choice Statechart pattern (i.e., transitions from one to different other states are possible) the resulting BPEL4WS process pattern depends on whether the choice is based on different conditions (then the corresponding BPEL4WS process is a `switch` construct) or if it is triggered by different incoming messages (then a `pick` construct is used instead, cf. Figure 5).

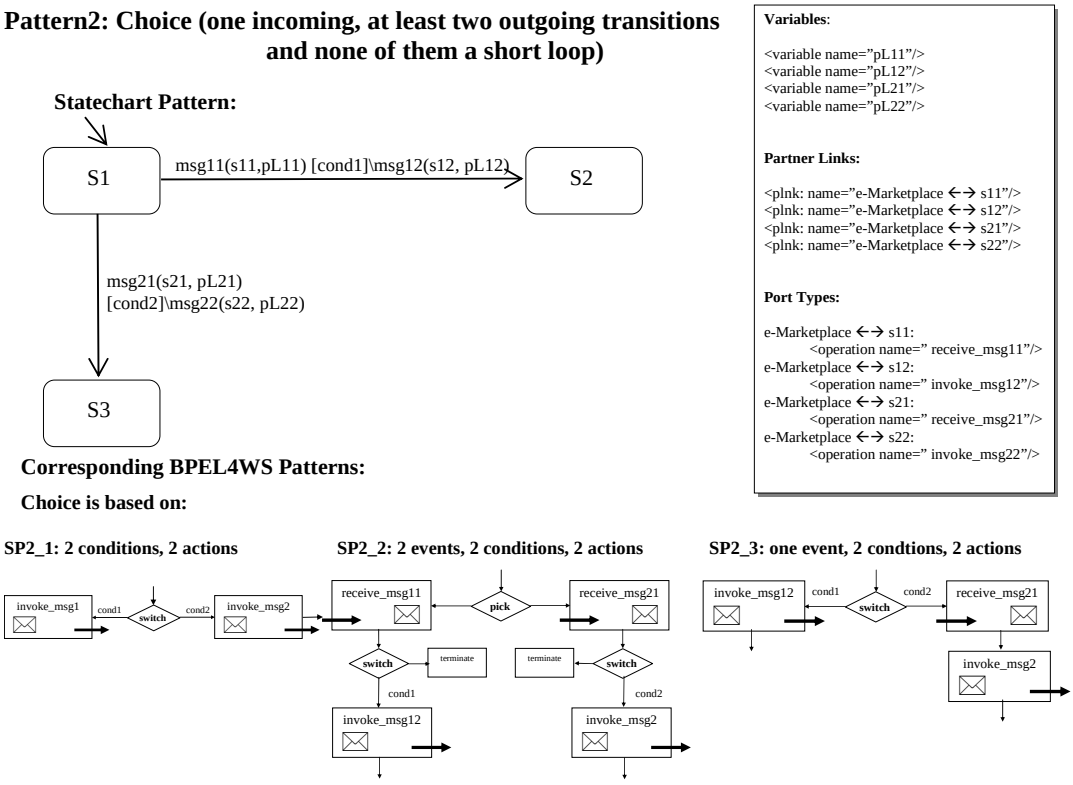

Figure 5: Choice with Different Sub-Patterns

Figure 6 depicts the mapping of the Short Loop Statechart pattern to a `while` construct in BPEL4WS. If the loop condition `cond2` becomes `TRUE` then the while loop is terminated and message `msg22` is sent by invoking the associated operation.

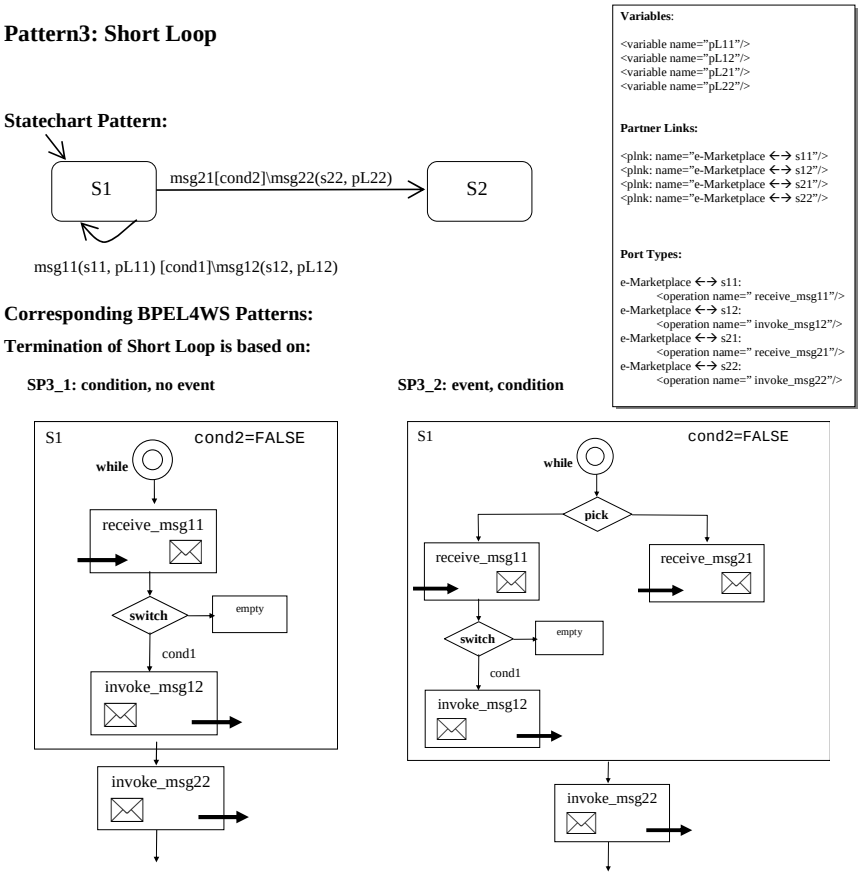

Figure 6: Short Loop with Different Sub-Patterns

The Mutual Call pattern shown in Figure 7 is also mapped to a `while` construct in BPEL4WS. Similar to the Short Loop pattern, the loop condition for the Mutual Call is either a conjunction of conditions `cond3` and `cond4` of the transitions from `S1` to `S3` or `S2` to `S4` or it is based on receiving messages `msg31` or `msg41`. Short Loop and Mutual Call patterns can also be combined (cf. sub-pattern `SP4_3` in Figure 7).

Associated with the Mutual Call pattern, additional sub-patterns exist, e.g., if there are more outgoing transitions from states `S1` or `S2` (cf. Figure 7). We have studied the possible sub-patterns but due to lack of space we abstain from giving further details here. Note that in case there are more mutual calls starting from a state, the mutual call sub-pattern `SP4_2` can be combined with the first `pick` construct.

Figure 8 depicts the mapping from the Statechart Concurrent Flow pattern to the associated BPEL4WS `flow` construct. Sub-pattern `SP5_1` shows the general mapping with sub-states `S21, …, S2n` which could be any other patterns. Sub-pattern `SP5_2` depicts the special case where leaving the concurrent execution is based on a choice, i.e., either there are loop backs within sub-states `S21` or `S22` or the transition condition from concurrent execution state `S2` to `S3` becomes true. There are more sub-patterns for Pattern 5. For example, if for sub-pattern SP5_2 the transition from S2 to S3 also contains a triggering event. In this case we need to add a `pick` construct to the BPEL4WS pattern.

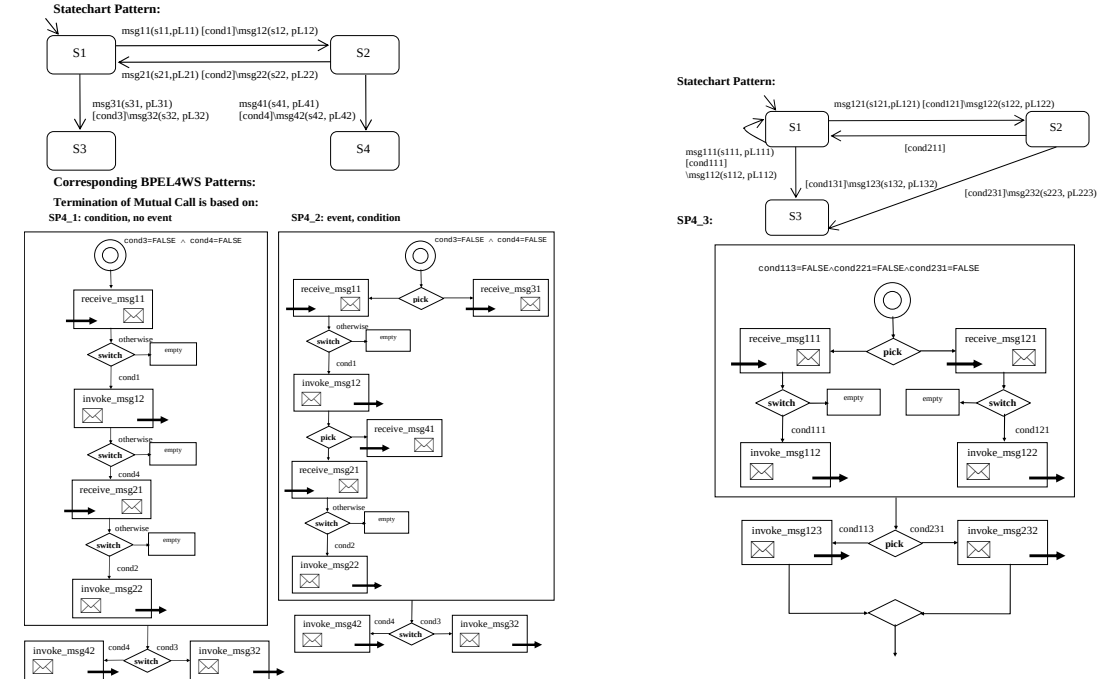

Figure 7: Mutual Call with Different Sub-Pattern

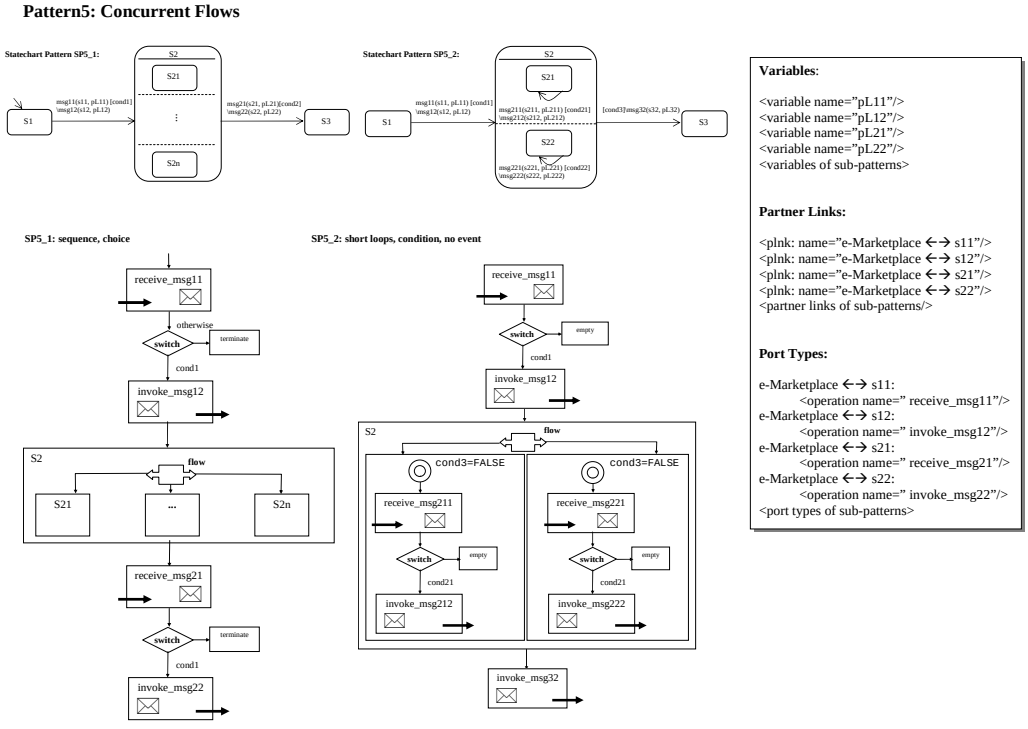

Figure 8: Concurrent Flow with Selected Sub-Patterns

4.4 Mapping Algorithm

The following algorithm maps Statechart models of e-negotiation protocols to the associated BPEL4WS processes. It is based on the pattern-wise mapping introduced in the previous section and proceeds in the following steps. Starting from the initial state the algorithm analyzes the Statechart patterns by traversing the Statechart graph and maps them to the associated BPEL4WS patterns as described in Section 4.3. There a Statechart pattern is determined by the currently analyzed state, its outgoing transitions, and its direct successor states.

Algorithm MapStateChartToBPEL4WS
input: Statechart model $ST = (S, T, E, C, A, s_{initial}, s_{final})$
output: BPEL4WS process
initialization
$CurrentStates := \{s_{initial}\}$; $VisitedStates := \emptyset$;
begin
// parse Statechart structure
while $VisitedStates \neq S$ do
  forall $s \in CurrentStates$ do
    if s is super-state of concurrent sub-states
      //Pattern5: Concurrent Execution
      add corresponding sub-pattern of Pattern5;
      VisitedStates := VisitedStates ∪ {s};
    determine set of outgoing transitions $tOut_s = \{t_s^1, \ldots, t_s^n\}$ of
    S with $t_s^k = (s, e_s^k, c_s^k, a_s^k, s^k)$, $k = 1,\ldots, n$;
    Pattern P is determined by s, $tOut_s$,
    $Succ_S = \{s^k \mid s^k \in S, k = 1, \ldots, n, s^k \neq s\}$,
    and $conT_s = \{t \mid t \in T, t = (s^{k1}, e^{k1}, c^{k1}, a^{k1}, s^{k2}), s^{k1,} s^{k2} \in Succ_s\}$;
    switch
    case1: $\exists\ t_s^{k1} = (s, e_s^{k1}, c_s^{k1}, a_s^{k1}, s^{k1}) \in tOut_s \wedge \exists\ t_s^{k2} \in T$ with $t_s^{k2} = (s^{k2}, e^{k2}, c^{k2}, a^{k2}, s)$
      if $\neg(\exists\ t_s^{k3} = (s, e^{k3}, c^{k3}, a^{k3}, s))$
        //Pattern4: Mutual Call
        concatenate corresponding sub-pattern of Pattern4 at dangling edge;
          if s^{k1} is super-state of concurrent sub-states
          //Pattern5: Concurrent Execution
          add corresponding sub-pattern of Pattern5;
          else // if $\exists\ t_s^{k3} = (s, e^{k3}, c^{k3}, a^{k3}, s)$
          //combined sub-pattern short loop and mutual call
          concatenate corresponding combined sub-pattern of Pattern4 at dangling edge;
    case2: $\exists\ t_s^k \in tOut$ with $t_s^k = (s, e, c, a, s)$
      //Pattern3: Short Loop
      concatenate corresponding sub-pattern of Pattern3 to dangling edges;
    case3: if $|tOut_s| > 1$
      //Pattern2: Choice
      concatenate corresponding sub-pattern of Pattern2 at dangling edge;
    case4: $|tOut_s| = 1$
      //Pattern1: Sequence
      concatenate corresponding sub-pattern of Pattern1 at dangling edge;
      CurrentStates := (CurrentStates ∪ $\{s^k \mid k = 1, \ldots, n\}$) \ {s};
      VisitedStates := VisitedStates ∪ {s};
end

Figure 9: Mapping Algorithm

We start with pattern Mutual Call which may be also combined with a Short Loop pattern. If the current pattern is neither a Mutual Call nor a Short Loop we check the number of outgoing edges. One outgoing edge indicates the Sequence pattern, more than one outgoing edge leads to the Choice pattern. Already "visited" states are stored within set `VisitedStates`. The states to be analyzed next are determined as the direct successors of the currently treated state (set `CurrentStates`). The sets of variables, partner links, and port types can be determined by merging the corresponding

sets of the particular patterns.

4.5 Examples

Applying the algorithm introduced in the previous section to the Statechart model of the Dutch auction protocol results in the BPEL4WS process depicted in Figure 10.

Starting with the outgoing transition of the initial state the first pattern to be inserted is the Sequence pattern (cf. Figure 10). The parameter list of the incoming message `Offer_to_sell` is described by the variable `offer`. The partner links comprise links to the seller and the buyers. The port types turn out as a `receive Offer_to_sell` operation within the partner link for the seller and an `invoke` of an `update` operation within the partner links of the seller and the buyers. The next state to analyze is `Offer`.

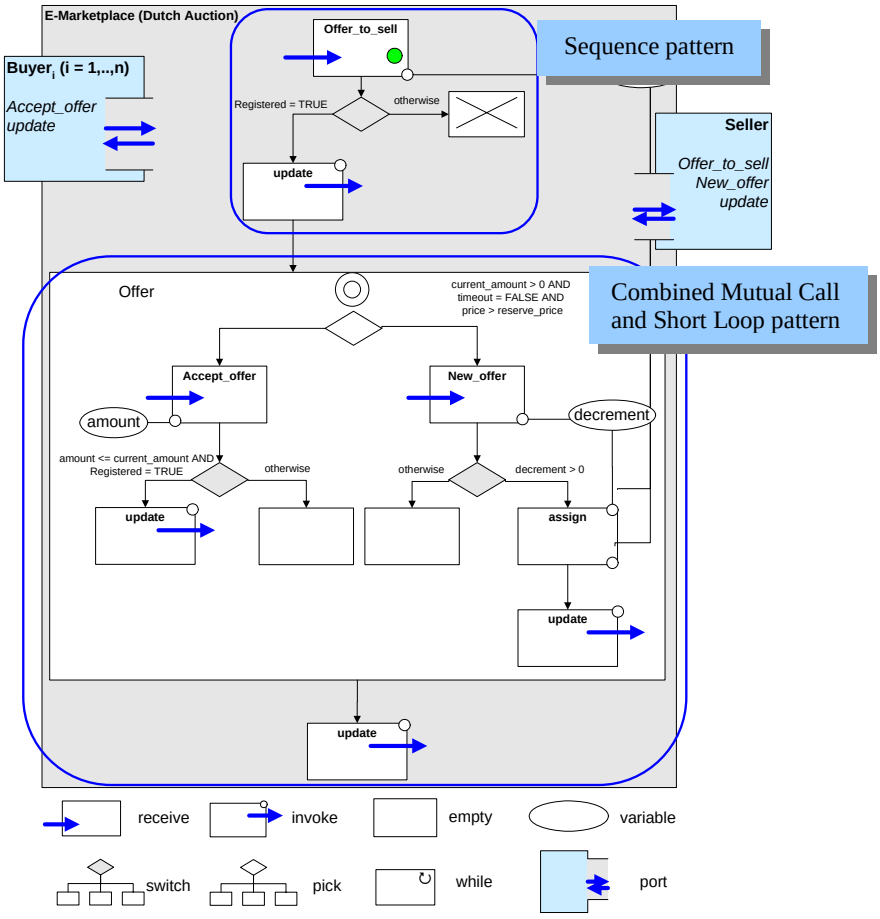

Figure 10: Web Service Orchestration of the Dutch auction Protocol

The associated pattern comprises the set of outgoing transitions $tOut_s$ = {(`Offer, …, Offer`), (`Offer, …, Deal`), and (`Offer, …, Auction Closed`)}, the set of direct successor states $Succ_s$= {`Deal, Auction Closed`}, and $conT_s$ = {`Deal, …, Auction Closed`} and corresponds to the combined (Mutual Call, Short Loop) sub-pattern `SP4_3` (cf .Figure 7). Note that assign activities are inserted corresponding to the action parts of the corresponding transitions.

5. Implementation

As a first step towards implementing the proposed framework (cf. Figure 3), we developed a prototype based on Oracle JDeveloper 10g (www.oracle.com) which includes the *BPEL*[4] *Process Manager Server* (the authoring tool) and the *BPELConsole* (the monitoring and control tool). The *BPEL Process Manager Server* is a Web Server which can execute SOAP messages between two or more of the interacting web services. These web services can be implemented using *JDeveloper 10g* whereas the WSDL itself can be browsed trough the *BPELConsole*. The *JDeveloper 10g IDE* allows us to graphically model BPEL processes using constructs such as *invoke*, *receive*, *switch*, or *assign*. Furthermore it is possible to modify the XML directly whenever necessary. Figure 11 depicts the (Mutual Call, Short Loop) pattern of the Dutch Auction processes modeled by using Oracle JDeveloper 10g.

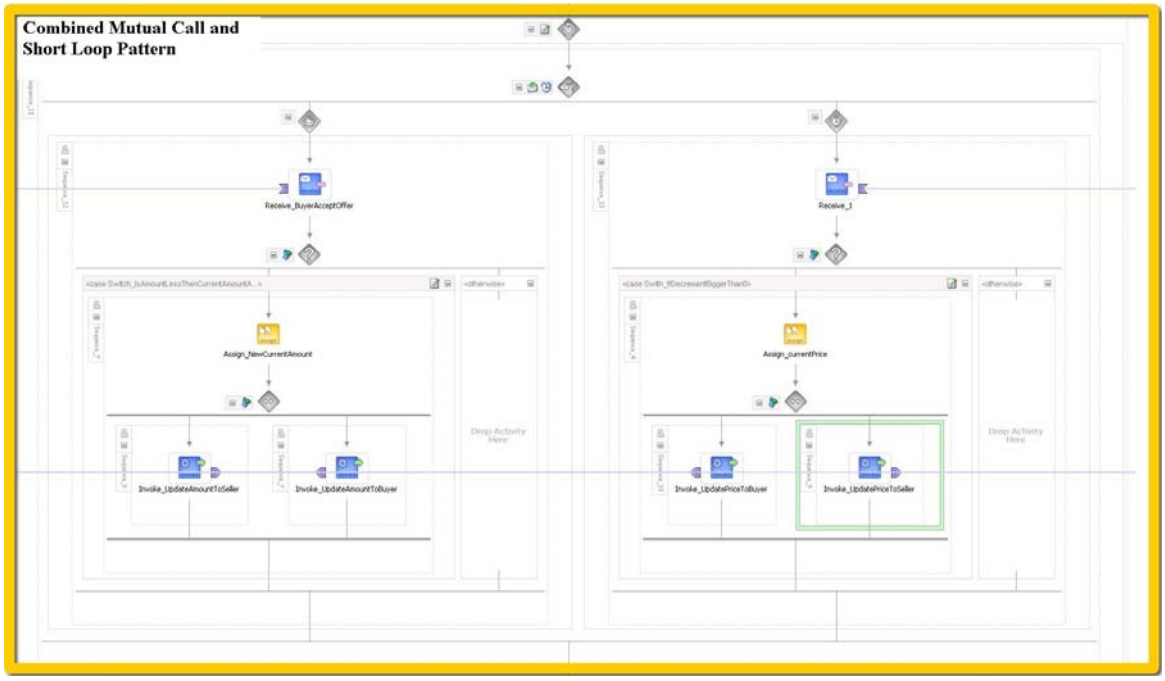

Figure 11: Section of the Dutch Auction Process Using Oracle JDeveloper

6. Discussion

In [KuFe98] different price negotiation protocols such as the Dutch auction are described using finite state machines. Though finite state machines are a formally founded formalism, Statecharts provide additional constructs (e.g., hierarchical states) which make them better suited for the modeling of e-negotiation protocols.

Rolli and Eberhart [RoEb05] propose a reference model for describing and running auctions as well as an associated three-layered architecture which consists of the auction data, the auction mechanism, and the auction participants. Apparently the auction protocols are modeled manually using BPEL4WS

[4] Nowadays BPEL is used instead of BPEL4WS for short.

which might be a complex task for users in general.

Kim and Segev [KiSe05] also follow an approach for establishing a web-service enabled e-marketplace. The authors provide a Statechart description for one e-negotiation protocol and the corresponding BPEL4WS process. In this paper we adopt the idea of providing understandable models for e-negotiation protocols and to automate them within a service-oriented architecture. However, we extend the approach of Kim and Segev towards a systematic description of generic e-negotiation protocols and by providing an automatic mapping of the Statechart models to the corresponding web service orchestrations.

In [SiRe04] e-negotiation protocols are modeled using the Petri Net formalism. Special focus is put on the modeling of attributes which reflect the different strategies the participants in the e-negotiation might adopt.

Chiu et al. [CCH+05] present an interesting approach for developing e-negotiation plans within a web services environment. The authors provide meta-models for e-contract templates and e-negotiation processes which can be used to set up the concrete e-negotiation processes within a web service environment. Though this approach is generic we believe that providing (generic) e-negotiation templates (i.e., Statechart models) to users which can be individually modified and immediately mapped onto executable web service orchestrations is more intuitive and user-friendly.

In [BDS05] the authors present a model-driven approach for developing web services. They model web services using the Statechart formalism and provide a mapping procedure from Statecharts to web services. Their paper presents a general and systematic mapping approach which has been implemented within a prototype called SelfServ. Due to the generality of the approach there are certain restrictions imposed on the Statechart models. One example is that transitions can be solely labeled by conditions (i.e., the authors do not consider the event and action part of the ECA rules). In our paper we introduce another way of addressing the challenge of mapping Statecharts to a web service orchestration language by exploiting the specificity of the e-negotiation domain.

7. Summary and outlook

This paper presented our current research on providing quick and elegant ways to develop e-negotiation systems. We proposed a service oriented framework for developing e-marketplaces and automated e-negotiation systems. The e-marketplace component implements a negotiation protocol and represents the virtual place where organizations and/or individuals meet to negotiate deals. The automated e-negotiation system component is the interface between the participant in the negotiation and the e-marketplace. Based on the negotiation protocol implemented on the e-marketplace, this component creates an automated entity capable of negotiating based on strategies and tactics provided by a human. The framework is based on (1) the separation of negotiation protocols from the e-marketplace; (2) the formal specification of these protocols using Statecharts; (3) an algorithm that transforms these Statecharts into web service orchestrations; and (4) the separation of negotiation strategies from the automated negotiation entity.

The framework is built on the assumption that e-marketplace participants are invited to join the negotiation beforehand. The number of participants is therefore known in advance, enabling us to fix the number of partner links in the BPEL4WS process before the negotiation starts. In B2B scenarios this assumption is realistic. Businesses usually choose their partners as well as the virtual markets where they negotiate very carefully, and most importantly they join the negotiation before it starts.

However in C2C scenarios (e.g., on the eBay marketplace) the number of participants is not known at the beginning as participants dynamically register and interact. In this situation the number of partner links and port types is unknown at the beginning of the negotiation. One future research direction is to enable the framework to enable new participants to enter the negotiation after it is started.

References

[ACD+03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana.: BPELWS - Business Process Execution Language for Web Services – Version 1.1., 2003.

[AHKB03] W.M. P. v.d. Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros: Workflow Patterns. Int'l Journal on Distributed and Parallel Databases 14(1):5-51 (2003)

[BBCT04] K. Baïna, B. Benatallah, F. Casati, and F. Tournani: Model-Driven Web Service Development. In Proc. 16th Intl Conf. on Advanced Information Systems Engineering, pages 290-306, Riga, June 2004

[BSS96] C. Beam, A. Segev, and J. G. Shanthikumar. Electronic negotiation through internet-based auctions. Tech. Rep. 96-WP1019, Haas School of Business, UC Berkeley, December 1996

[BDS05] B. Benatallah, M. Dumas, and Q.S. Sheng: Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services. Int'l Journal on Distributed and Parallel Databases 17(1):5-37 (2005)

[BKL+00] M Benyoucef, R. K. Keller, S. Lamouroux, J. Robert, and V. Trussart. Towards a Generic E-Negotiation Platform. In Proceedings of the Sixth International Conference on Re-Technologies for Information Systems, pages 95-109, Zurich, Switzerland, February 2000

[BKS03] M. Bichler, G. Kersten, and S. Strecker. Towards a Structured Design of electronic Negotiations, Group Decision and Negotiation 12: 311–335, 2003

[CCH+05] D.K.W. Chiu, S.C. Cheung, P.C.K. Hung, S.Y.Y. Chiu, A.K.K Chung: Developing e-Negotiation Support with a Meta-Modeling Approach in a Web Services Environment. Int'l Journal on Decision Support Systems. Special Issue on Web Services and Process Management 40(1): 51-69 (2005)

[DuHo01] M. Dumas and A.H.M. ter Hofstede: UML Activity Diagrams as a Workflow Specification Language. In Proc. 4th Int'l Conf. on The Unified Modeling Language, pages 76-90, Toronto, Canada, 2001.

[Eau02] University of Washington. The eAuctionHouse. 2002

[Hare87] David Harel: Statecharts: A Visual Formulation for Complex Systems. Int'l Journal Scientific Computer Programming 8(3): 231-274 (1987)

[Hur00] Hurwitz Report. Negotiated Trade: the Next Frontier for B2B e-commerce. Technical Report. 2000.

[KLS04] G. Kersten, K. P. Law, and S. Strecker. A Software Platform for Multi-Protocol E-Negotiations. An InterNeg Research Report 04/04 (available at http://interneg.org), April 2004

[KiSe05] J. B. Kim and A. Segev: A Web Services-Enables Marketplace Architecture for Negotiation Process Management. Int'l Journal on Decision Support Systems. Special Issue on Web Services and Process Management 40(1):71-87 (2005)

[KuFe98] M. Kumar and S.I. Feldman: Business negotiations on the Internet. Technical Report, IBM Research Division, New York, 1998

[KuFe98a] M. Kumar and S.I. Feldman: Internet Auctions. Technical Report, IBM Research Division, New York, 1998

[Mal87] T. W. Malone, et al., Electronic Markets and Electronic Hierarchies. Communications of the ACM, 1987, 30(6): p.483-494.

[Muth98] P. Muth, D. Wodtke, J. Weißenfels, A. Kotz Dittrich, and G. Weikum: From Centralized Workflow Specification to Distributed Workflow Execution. Int'l Journal of Intelligent Information Systems 10(2): 159-184 (1998)

[NBBV03] D. Neumann, M. Benyoucef, S. Bassil, and J. Vachon. Applying the MTL Taxonomy to State of the Art E-Negotiation Systems. Int'l Journal on Group Decision and Negotiation, 12 (4): 287-310, July 2003

[OMG99] Object Management Group (OMG) Negotiation Facility final revised submission. Technical report. March 1999.

[RoEb05] D. Rolli and A. Eberhart: An Auction Reference Model for Describing and Running Auctions. In Proc. of the Wirtschaftsinformatik, Bamberg 2005.

[ReDa98] M. Reichert and P. Dadam: $ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Losing Control. Int'l Journal of Intelligent Information Systems 10(2):93-129 (1998)

[RRD04] S. Rinderle, M. Reichert, and P. Dadam: Flexible Support of Team Processes by Adaptive Workflow Systems. Int'l Journal on Distributed and Parallel Databases 16(1):91-116 (2004)

[RiBe05] S. Rinderle, M. Benyoucef: Towards the Automation of E-Negotiation Processes Based on Web Services – A Modeling Approach. Working Paper, School of Management, University of Ottawa, Canada (2005).

[SiRe04] C. Simon and M. Rebstock: Integration of Multi-attributed Negotiations within Business Processes. In Proc. 2nd Int'l Conf. on Business Process Management, pages 148 – 162, Potsdam, Germany, June 2004

[San99] T. Sandholm. An Algorithm for Optimal Winner Determination in Combinatorial Auctions. In International Joint Conference on Artificial Intelligence, pages 542-547, Stockholm, Sweden, 1999.

[WWW98] P. Wurman, M. Wellman, and W. Walsh. The Michigan Internet AuctionBot. In 2nd Intl Conf on Autonomous Agents, pages 301-308, Minneapolis, MN, May 1998