Combined Negotiations in E-Commerce: Concepts and Architecture[1]

Morad Benyoucef, Hakim Alj, Mathieu Vézeau, and Rudolf K. Keller

Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, C.P. 6128 Succursale Centre-ville, Montréal, Québec, H3C 3J7, Canada

{benyouce, aljh, vezeau, keller}@iro.umontreal.ca

Abstract. Combined Negotiations are a novel and general type of negotiation, in which the user is interested in many goods or services and consequently engages in many negotiations at the same time. The negotiations are independent of each other, whereas the goods or services are typically interdependent. Using currently available technology for electronic negotiations, the user conducts each negotiation separately, and has the burden of coordinating and reconciling them. The inherent complexity of combined negotiations in B2C as well as B2B e-commerce calls for software support.

In our research, we aim to devise a Combined Negotiation Support System (CNSS) to help the user conduct all the negotiations at the same time. The CNSS enables the user to control and monitor the progress of the negotiations, makes sure that the specified dependencies are respected, and applies user-defined strategy rules. We have designed such a CNSS which we call CONSENSUS. The architecture of CONSENSUS relies on workflow technology, negotiating software agents, and rule engine technology. The originality of this architecture lies in the fact that the user of CONSENSUS models the combined negotiation at build time using a workflow that captures the sequencing of the individual negotiations and the dependencies between them. At runtime, software agents are assigned to individual negotiations, and they participate in the combined negotiation as actors in the workflow. The user can monitor the progress of the combined negotiation as a whole, and the progress of individual negotiations via dedicated graphical user interfaces. We rely on rule engine technology to enable the agents to use negotiation strategies.

The paper introduces combined negotiations with a usage scenario. Then, combined negotiations are detailed, along with the approach taken to cope with their complexity. Afterwards, we describe the functionality a CNSS should provide, and present the architecture of CONSENSUS, together with a discussion of the underlying concepts and technologies. Furthermore, we report on our prototype implementation of CONSENSUS and illustrate it with an example. A discussion of related and future work concludes the paper.

Keywords: electronic commerce, electronic negotiation, combined negotiation, auction, commitment, workflow, software agent, negotiation strategy, rule engine.

1. Introduction

Electronic negotiations (e-negotiations) are becoming an important research subject in the area of electronic commerce (e-commerce). The AMEC laboratory of MIT [1], for instance, puts e-negotiations at the center of its Consumer Buying Behavior (CBB) model for e-commerce [2]. The model identifies six steps in an e-commerce transaction: (1) the need identification step in which the buyer is stimulated through product information, (2) the product brokering step in which information is retrieved to help the consumer determine what to buy, (3) the merchant brokering step in which the consumer determines who to buy from, (4) the negotiation step where the price and possibly other aspects of the deal are settled, (5) the purchase and delivery step, and finally (6) the product and service evaluation step where the product and service are evaluated by the consumer.

[1] This research is supported by Bell Canada, BCE Emergis, NSERC (National Sciences and Engineering Research Council of Canada), and CIRANO (Centre Interuniversitaire de Recherche en ANalyse des Organisations).

The most basic form of e-negotiation, as described by Kumar et al. [3], is no negotiation at all (also called fixed-price sale) where the seller offers her goods or services through a catalogue at take-it-or-leave-it prices. This is the way most electronic retailers (e-tailers) operate. Auctions are a bit more complex, and they are at present the most visible type of e-negotiations on the Internet as conducted by eBay [4], OnSale [5], Yahoo [6] and hundreds of other auction sites. A comprehensive description of the different auction types can be found in [7]. E-negotiations can take an even more complex form called bargaining. This involves making proposals and counterproposals until an agreement is reached [8]. The Object Management Group (OMG) sees bargaining as bilateral and multi-lateral negotiation depending on whether there are two parties (one-to-one bargaining) or many parties (many-to-many bargaining) involved in the negotiation [9]. Bargaining can become challenging if the object of the negotiation has more than one negotiable attribute (e.g., the price, the quality, the delivery date, etc.). In this case we talk of multi–attribute negotiations. A negotiation is said to be distributive (of win-lose nature) if a gain for one agent is necessarily a loss for the other agent [10]. It is said to be integrative (of win-win nature) if the parties do not necessarily have opposing interests, and they try to optimize different attributes. The buyer for instance can hope to pay a small price if she can live with a poorer quality and/or can stand a long delivery time. Combinatorial auctions [8] are another form of negotiations that involve making bids on combinations of goods or services, and one of the main challenges is for the auctioneer to determine the winning bid.

Jhingran [11] proposes a three-dimensional negotiation space (see Figure 1, solid lines). The first dimension is for the case where *multiple copies* of the same item (good or service) are available for negotiation, and the bids (i.e., offers) take the form of pairs (quantity, price-per-unit). The second dimension addresses the case where *multiple items* are subject to one negotiation. In this case, the participants make bids on combinations of these items. The third dimension is for *multiple attribute* negotiations. Jhingran's model obviously addresses only one aspect of a negotiation, which is the item being negotiated. A more complete classification of e-negotiations can be found in [10]. The dscription identifies three more aspects of a negotiation: the participants (the "people" aspect), the type of the negotiation (the "process" aspect), and the criteria that can be used to evaluate the process (the "evaluation criteria" aspect).

We talk of a Combined Negotiation (CN) when a consumer is interested in many items, and consequently engages in many negotiations at the same time. The negotiations can be of any type (fixed-price sale, Dutch auction, bilateral bargaining, combinatorial auction, etc.). Each negotiation is for a separate bundle of copies, items, and attributes and thus corresponds to one point in the 3D negotiation space (see Figure 1, solid lines). The negotiations are in general totally independent of each other. The goods and services of the CN, however, are typically interdependent. To capture this new type of negotiation, where we have *multiple negotiations* going on at the same time, we introduced a fourth dimension in Jhingran's model (see Figure 1, dashed line).

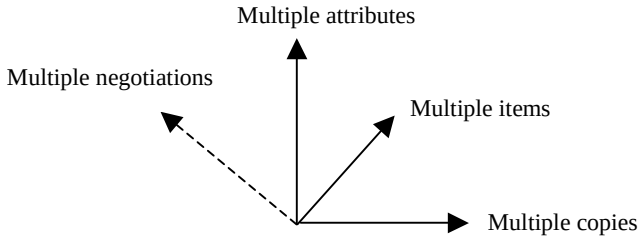

Figure 1: The four-dimensional negotiation space.

As an example of a CN, let us take a vacation package consisting of three items: a transportation ticket, a hotel room, and a ski trip. The three items are obviously interrelated since the consumer would have to travel to the location where the ski trip journey initiates (or at least near it) on the date of the trip (or before it). In addition to the places and dates, there can be other constraints and dependencies between the three individual items, as we will see later on. Let us suppose that the three items are negotiable (keeping in mind that a fixed-price sale is a special case of a negotiation) and that they can be negotiated on different negotiation servers. Let us suppose also that the negotiations practiced on each single server can be of different types. Clearly, the individual negotiations are independent of each other. Using currently available negotiation technology, the consumer would have to conduct each negotiation separately, and she would have the burden of coordinating and reconciling the various negotiations. It can happen for instance that the

consumer makes a deal on a plane ticket and a hotel room, and then, while negotiating the ski trip, finds out (through bargaining for example) that she is missing out on a very interesting deal only because she arrives a few hours late.

We therefore see a pressing need for a software system that supports the user in conducting all the negotiations at the same time, i.e., in carrying out CNs. We call such a system a Combined Negotiation Support System (CNSS). A CNSS is a tool that enables the user to track and monitor the progress of many negotiations efficiently and to respect all the constraints, dependencies and preferences of the given context. Moreover, a CNSS will support the user in taking decisions.

Before we go further, and to avoid any confusion, we shall first clarify the concept of "negotiation server" which will be heavily used in this paper. It is possible that many negotiations are created and conducted on one single server. These negotiations are independent and therefore can run in parallel. An example of such a server is eBay [4]. It is also possible that a server conducts only one negotiation. This is the case where a company makes a dedicated server available for its consumers to negotiate a particular item. In both cases, we will use the term "negotiation server". Another type of negotiation server accepts connections from users, takes a description of their needs and preferences, and then negotiates on their behalf [12]. We will call this type a "proxy negotiation server".

The work presented in this paper is part of the TEM (Towards Electronic Marketplaces) project, a joint industry-university project started in 1999 involving researchers from economic science, software engineering, and operations research. The project addresses market design issues in respect to resource allocation and control and reward mechanisms, investigates open protocols for electronic marketplaces, and explores concepts and tools for e-negotiations. Two key tools being investigated in TEM are the CNSS described in this paper, and a generic negotiation server infrastructure, which we call GENESIS [13]. GENESIS is already functional and we are using instances of it to implement the servers involved in a CN. GENESIS is generic as it can support a great variety of negotiation types. It also offers the possibility to the participants to negotiate via a web browser or via software agents.

The contribution of this paper is twofold. First, we define the concept of CNs and present the problems they generate along with the approach taken to solve them. Second, we present an architecture for a CNSS we call CONSENSUS, which is based on workflow technology, negotiating software agents and rule engine technology. The originality of CONSENSUS stems from the fact that the user models her CN at build time using a workflow. The workflow itself captures the sequencing of the individual negotiations and the dependencies between them. At runtime, software agents are assigned to individual negotiations and participate in the CN as actors in the workflow. The user of CONSENSUS can track and monitor the progress of the CN via the Workflow Monitoring and Control Tool of a Workflow Management System (WfMS). She can also monitor the work of the agents via an Agent Monitoring and Control Tool. The four-component model proposed by the Workflow Management Coalition (WfMC) [14] serves as the core of our CNSS architecture. We use Rule Engine technology to enable the agents to use negotiation strategy rules. These rules can be edited before the negotiations start and/or during the process of the negotiations. The formal description of negotiation rules[2] is also investigated, as it is necessary for CONSENSUS to transfer the negotiation rules from the corresponding negotiation servers to its Negotiation Rules Repository.

This paper[3] is organized as follows. In Section 2, we briefly describe a CN usage scenario. The concept of combined negotiation is detailed in Section 3, and in Section 4, we present our approach to solving the problems related to CNs. The architecture of CONSENSUS as well as an overview of its underlying concepts are covered in Section 5. Section 6 presents an example of a CN. Section 7 reports on the implementation of CONSENSUS, and Section 8 discusses related work. Finally, Section 9 concludes the paper.

[2] Negotiation rules mean the rules (i.e., the valid actions) of the game. A participant in the negotiation uses strategy rules to maximize her utility.

[3] This paper is a revised and extended version of [41].

2. Usage Scenario

Let us go back to our vacation package example. The consumer uses CONSENSUS to register, say, in one negotiation for the hotel room, two different negotiations for transportation, and one for the ski trip. CONSENSUS is then used to construct a workflow that models the CN (we will see the details later on). The workflow reflects the sequencing of the individual negotiations and the dependencies between them. The consumer might want, for example, to run all the four negotiations in parallel, or run all or some of them sequentially. The workflow also reflects the dependencies between individual negotiations such as the amount of money already committed and thus the remaining amount to be spent. After the consumer finishes modeling the CN, four software agents are instantiated and assigned to the four different negotiations. The agents are created according to the type of the negotiation practiced on the corresponding servers. The consumer can provide the agents with negotiation strategies such as: "if your bid is always beaten by the same opponent then be less aggressive in your bidding " or "if you have little chance of making a deal on the ski trip then don't commit yourself on the other two items of the package". Note that the first strategy applies to one individual negotiation whereas the second one applies to the CN as a whole. The negotiation strategies can be communicated to the agents before the negotiation starts or during the course of the negotiation. This scenario is an example of B2C e-commerce. In case one or more items in the package are offered by consumers (i.e., a rare ticket to a rock concert auctioned on an auction site), we would face a C2C transaction.

CONSENSUS can also be used at the B2B level. A travel agency, for instance, can use CONSENSUS to negotiate travel packages on behalf of its clients. The more items there are to be negotiated and the more providers of such items there are, the more there is a need for a support tool. We might imagine different means of transportation (by air, by bus, etc.), different types of accommodations (hotel, motel, pension, etc.), and different recreational activities (ski trip, camping, hiking, etc.). We might also imagine that there are many providers of such services and that each provider might practice a different type of negotiation. As another example at the B2B level, consider a company that wants to import some merchandise from a foreign country. The attributes and issues to be negotiated are: the price as well as other attributes of the merchandise, the transportation, the insurance, the customs, etc. These examples suggest that at the B2B level, CNs may become highly complex, and that automated support by a CNSS is even more important.

CONSENSUS offers the possibility to find the products and their providers (steps 2 and 3 of the CBB model) along with complete information to help the user decide which products to negotiate and where to negotiate them. The negotiation types practiced by the corresponding servers should be known to the user as it is an important factor in choosing one provider over another. She might for example prefer a bargaining type negotiation to an English auction if she is a good haggler. Once the choice is made, the CN can be modeled. A high-level specification tool is provided by CONSENSUS to specify the workflow and an English-like syntax is used to specify the negotiation strategies. When the modeling phase is over, the CN is launched by running an instance of the workflow, which in turn launches the software agents. CONSENSUS provides the user with the possibility to track the individual negotiations as well as the CN as a whole. The user can also edit negotiation strategies at run time.

We see CONSENSUS as a support tool, where the automation provided by the workflow engine (to run the CN) and the software agents (to run the individual negotiations) should allow the user to intervene in the important aspects of the negotiation whenever a need should arise. Our CNSS is not meant to replace the user, but to provide her with a powerful tool.

3. Issues in Combined Negotiations

In this section, we address seven important issues related to CNs. We first define CN failure, and then present the notions of AND-Negotiation and OR-Negotiation. Thereafter, we discuss the negotiation types covered by a CN. Then, constraints in a CN are explained. We then present the notion of commitment in negotiation protocols followed by a comparison of the CN and the synchronized auction types. Finally, we classify the information involved in a CN.

3.1. Failure in Combined Negotiations

When we engage in a negotiation, there is no guarantee that we will end up as a winner, and even if we do win, there is no guarantee that we will make a good deal by paying the real value of the good (this last case is referred to as the *winner's curse* meaning that the winner has possibly over-evaluated the item and ends up paying more than the item is worth).
In a CN the risks for the user are even higher because she has to negotiate several items that form a package. Whether the individual negotiations are conducted in parallel or in sequence, there is always the risk that some individual negotiations will be successful and some will not be. Since the user wants the whole package or nothing, she might want to break (if allowed) the commitments she made in the successful negotiations. Breaking a commitment evidently has a price. We talk of failure (also referred to as exposure) in a CN, when we need two items A and B, engage in negotiations on both items, and end up winning on A but loosing on B.

3.2. AND-Negotiation and OR-Negotiation

The package that is the object of the CN is in general made up of many items. If we engage in many negotiations for the same item we call these OR-Negotiations. In the case of a vacation package, for example, we can engage in more than one negotiation for the plane ticket. Negotiations for different items that make up the package are called AND-Negotiations. We might, for example, engage in one negotiation for the plane ticket (possibly an OR-Negotiation), another one for the hotel room, etc.

OR-Negotiations. OR-Negotiations will usually run in parallel. This means that, in order to save time and also to maximize the chances of a good deal, the negotiations for one item are started at the same time. This will enable the user to see the progress of the negotiations and choose to bid (make offers, counteroffers) in the most promising one. We may have to ensure that no more than one commitment (bid, offer, etc.) is made at the same time, that is, commitments must be mutually exclusive. If we do not take this precaution, we might end up making more than one successful deal on the same item. An alternative scenario would be to run OR-Negotiations sequentially. Whenever one fails (the bids are too high, the negotiation ended, etc.), we start another one for the same item.

AND-Negotiations. Here again the user has the choice of running her individual negotiations in parallel or sequentially. Running them (or some of them) in parallel will be more interesting for the user. She can make decisions in one negotiation based on what is happening in the other ones. The type of the negotiation can also make it appealing to run the AND-Negotiations in parallel, as we will see in the next subsection.

3.3. Negotiation Types covered by a Combined Negotiation

The individual negotiations for the different items in the package can be of any type. The consumer might engage in a bargaining type negotiation for the ski trip, an English auction for the plane ticket, and a Dutch auction for the hotel room. Each negotiation type evidently will require different rules and strategies. Therefore, the CN depends on the types of the individual negotiations. If the consumer is interested in an item that is offered in a sealed-bid auction that ends in 10 hours, she might for example try to finalize deals on the other items in the package and use the remaining amount of money to make a bid in the sealed-bid auction.
The fact that more than one attribute of an item is negotiable in general complicates the dependencies among the individual negotiations. Let us go back to the vacation package and suppose that it is made up of three items: a plane ticket (price, date, week-day/week-end, first-class/second-class, destination, etc.), a hotel room (price, date, type, location, etc.), and a ski trip (price, date, location, etc.). The outcome of one negotiation will be crucial in the other ones (the date of the flight from the first negotiation will be a necessary input to the other two negotiations).
These examples suggest that multi-attribute negotiations, possibly of different types, make CNs flexible. Yet, such negotiations may incur complex dependencies among the items that form the CN package and thus warrant software support.

3.4. Constraints in a Combined Negotiation

Let us consider a CN for three different items i1, i2 and i3. The individual negotiations are N1, N2 and N3. Suppose that the attributes for each item are the price, the date, and the place. We classify CN constraints as intrinsic or procedural.

Intrinsic constraints: These constraints concern the dependencies between the items, copies, and attributes of the CN. They may involve just one single individual negotiation, such as:

- price1 * number-of-copies <= threshold.
- date1 in RANGE.
- place1 in (X, Y, Z).

Other constraints may involve more than one of the individual negotiations forming the CN. Examples of such constraints include:

- price to pay for the package <= threshold.
- date3 = function(date1, date2).

Procedural constraints: These constraints indicate the sequencing in time of the individual negotiations and the control flow between them. Examples of such constraints include:

- Sequential: N2 is launched after N1 is finished.
- Parallel: N1 and N2 are launched at the same time.
- Choice: depending on a condition, either N1 is launched or N2 is launched.
- Wait for: N3 waits for N1 and N2 to finish or waits for either one to finish (we suppose that N1 and N2 both start before N3 and that N3 is aware of the status of N1 and N2).
- Repeat: repeat N1 until a condition is met.

3.5. Commitment in Negotiation Protocols

A further issue of importance in CNs are commitments in negotiation protocols. It is obvious that, given the possibility of breaking a commitment (even at a certain cost to the user), the CN will be more flexible and the role of the CNSS even more important. According to Sandholm and Lesser [15], commitment means that one agent (human or software agent) binds itself to a potential contract while waiting for the other agent to either accept or reject its offer. If the other party accepts, both parties are bound to the commitment. When accepting, the second party is sure that the contract will be made, but the first party has to commit before it is sure. As an example, let us consider an English auction with the possibility to break a commitment. If your bid is the winner (a commitment from you) and you decide to leave the auction (break the commitment), then you will have to pay at least the difference between your bid and the second highest bid (or be subject to another penalty agreed upon in advance).

3.6. Combined Negotiations versus Synchronized Auctions

A CN bears some resemblance to a well-known auction type called the synchronized auction, and for that reason, we would like to point out the differences and similarities between the two. The following definition is taken from [10]. In a synchronized (also called simultaneous or parallel) auction there are *n* items. A participant can make *m* distinct bids on *m* distinct items (m not greater than n). The *m* bids are made simultaneously. The auction usually runs multiple rounds of sealed bids, announcing the bids after each round. The Federal Communication Commission (FCC) of the U.S. uses such a format to auction licenses for Personal Communication Services (PCS). Note that a variation of the synchronized auction is the so-called combinatorial (or bundled) auction [8], which allows the participant to make a single bid for *m* possibly distinct items (*m* not greater than *n*).

The similarity between a CN and a synchronized auction is the fact that the user can make many distinct bids (offers or counteroffers) on many distinct items, and that the items might be interrelated. However, a CN may involve many different individual auctions and/or negotiations, whereas the synchronized auction is one single auction with a set of known rules. Moreover, a CN is not synchronized. A CN should not be

confused with a combinatorial auction either. In a CN, the user cannot make a single bid on all the items of the package.

3.7. Information Involved in a Combined Negotiation

The negotiation process is an exchange of information between the participants and the negotiation server. It "consists of a number of decision-making episodes, each of which is characterized by evaluating an offer, determining strategies and generating a counteroffer" [16]. In order to evaluate an offer, the participant needs all the information she can get from the negotiation server. Based on that information, she can choose a negotiation strategy that helps her shape a counteroffer in response to the offer made by the server. We define a negotiation strategy as the way in which a human negotiator would react in her best interest in a given situation and given the information that is available to her. Two examples of negotiation strategies in an English auction are: "when the highest bid reaches 1000 dollars, then make a bid by doubling your bid increment" and "if the frequency of bids is low, then be less aggressive in your bidding." Here "the highest bid" is equivalent to the "offer" coming from the server, and "make a bid" is equivalent to the counteroffer to be made by the participant. How much the participant bids is decided by applying negotiation strategies that rely on information that we classify as follows:

- Internal versus external: the information can be internal to the CN (the total amount the user is willing to pay, her reserve price, etc.) or external (the current quote or highest bid, the identity of the other participants (if available), the profile of the other participants (if available), etc.).
- Individual versus combined: it can be related to an individual negotiation (the actual quote, the time remaining, etc.) or to the CN at large (the number of negotiations where the consumer is leading, the total amount of money already committed, etc.).
- Static versus dynamic: it can be static (the total amount the user is willing to pay, etc.) or dynamic (the frequency of bids, the average bid increment, etc.).

4. Tool Support for Combined Negotiations

We are aiming at a tool (CONSENSUS) that helps manage and possibly minimize the risks that the consumer faces in a CN. We see the tool as a support system for the consumer and not as a system that can replace her. Furthermore, note that CONSENSUS is quite different in scope from Negotiation Support Systems (NSS) [17]. The main idea behind these latter systems is to help the user evaluate offers made by her opponents, and evaluate her counteroffers before she submits them to her opponents. All this happens in a bargaining type negotiation. This does not preclude, of course, the use of NSSs on top of CONSENSUS, in order to assist the agents in their evaluation of offers and counteroffers.
The functionality of CONSENSUS can be summarized in the following four points:

- Model the CN using a workflow definition formalism to specify its intrinsic and procedural constraints. We realize that in general there may be a need to change both the intrinsic and the procedural CN constraints dynamically, that is, during execution of CONSENSUS. Accordingly, CONSENSUS should support such dynamism.
- Use software agents for automation. Software agents are assigned to individual negotiations.
- Use a WfMS to coordinate, track, and monitor the work of the negotiating software agents.
- Use negotiation strategy rules to make the agents more autonomous. The user should have the possibility to enter strategy rules and edit them at runtime.

As seen earlier, there is a CN failure when we need two items A and B, we engage in negotiations on both items, and end up winning on A but loosing on B. We propose the following three solutions to address CN failure:

- Provide negotiation rules that allow the user to break a commitment. In this case the user backs off and breaks her commitment on item A and (eventually) pays a price for that. This is a CN failure resolution solution.
- Engage in many OR-Negotiations so that whenever we loose on item B we engage (or resume) a negotiation on item B on another server. This, of course, is not a guarantee against exposure. This is a CN failure prevention solution.

- Use strategies to minimize the risk of exposure. An example would be: if you notice that the competition is high on item B (and that you have little chance of winning), then do not make a commitment on item A. Here, once more, we see the need for coordination between individual negotiations. This is again a CN failure prevention solution.

Finally, a commitment can be different from one negotiation type to another. In an English auction, to bid means that you make a commitment to buy in case you stay leader until the closure of the auction. In a Dutch auction, however, by bidding you make a commitment that is binding and final. In a "bilateral bargaining" type negotiation, offers and counteroffers are exchanged between the user and the negotiation server until an agreement is reached or until the negotiation is aborted. Offers become "binding" only when they are "final" (i.e., if accepted they must be honored). CONSENSUS will take into account the different types of commitments. If the rules of the negotiation allow for commitments to be broken, then this fact should be reflected in the rules describing the negotiation and in the workflow that models the CN.

5. Architecture of CONSENSUS and Underlying Concepts

In this section, we first describe the architecture of CONSENSUS. Then, we give a brief overview of the concepts and technologies it is based upon, that is, the formalism for describing negotiation rules, software agents, workflow management, as well as negotiation strategies and rule engines.

5.1. The Architecture:

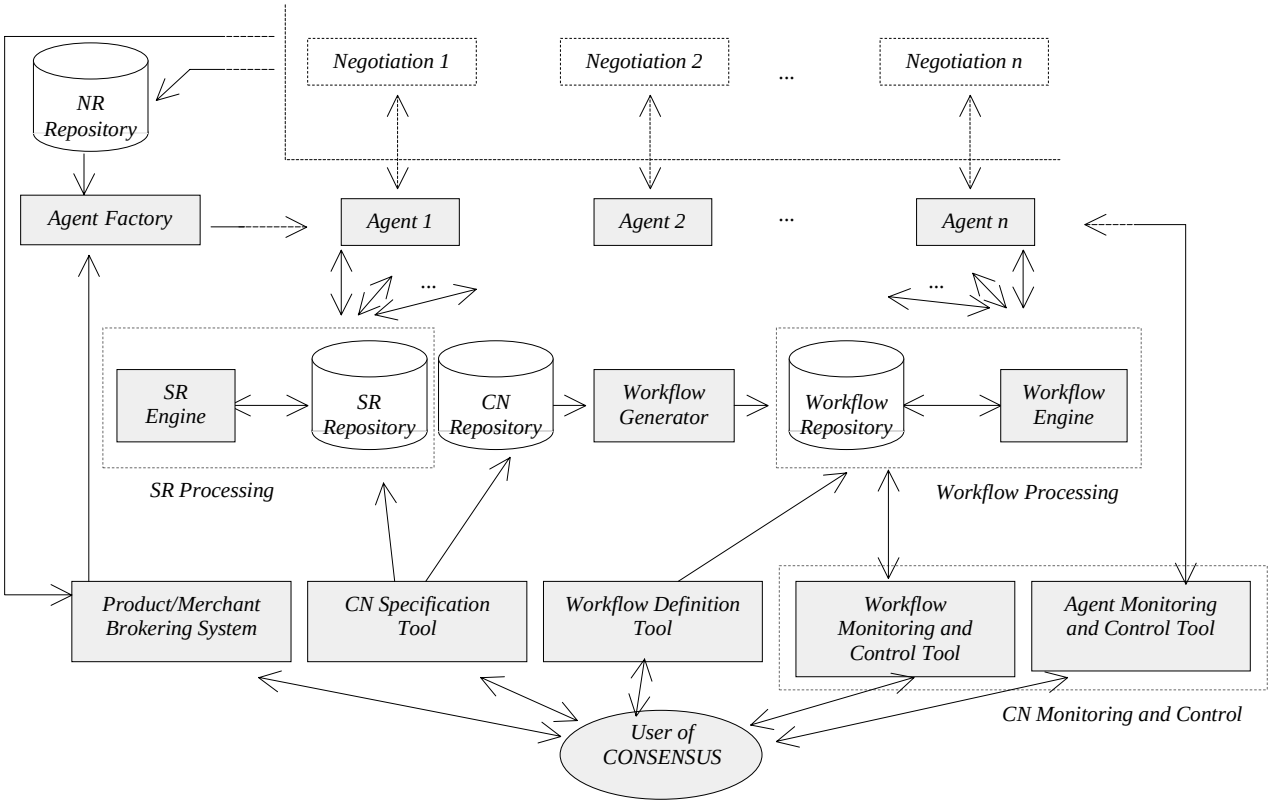

Figure 2: Architecture of CONSENSUS.

Figure 2 shows the architecture of CONSENSUS. The *Product/Merchant Brokering System* (responsible for phases 2 and 3 of the CBB model) is used by the consumer to select the products and the providers she is willing to negotiate with. The system will return a list of candidate negotiations, and the user selects the ones she is interested in. This list is passed on to the *Agent Factor*y. The Negotiation Rules (NR) are then downloaded from the corresponding negotiation servers. We are using XMI [18] as a vehicle for such a

transfer. The rules are stored in the *NR Repository* and are used by the *Agent Factory* to instantiate the *Agents* that will be responsible for the individual negotiations. Each agent will be instantiated according to the negotiation type practiced by the negotiation server assigned to it. After they are instantiated, the agents connect to and register on the corresponding servers. The consumer uses the *CN Specification Tool* (a GUI tool) to specify the CN. A constraint language will be used to represent, manipulate and store the constraints of the CN. The constraints are stored in the *CN Repository*. Another part of the specification deals with negotiation Strategy Rules (SR). These go into the *SR Repository*. When the CN specification task is completed, the workflow modeling the CN is generated and stored in the *Workflow Repository*. The consumer can use the *Workflow Definition Tool* to see and debug the resulting workflow. This ends the modeling part. At runtime, an instance of the workflow is started. The workflow can be monitored via the *Workflow Monitoring and Control Tool*. The *Workflow Engine* executes the workflow by dispatching work to the agents (which are actors in the workflow) and enforces the sequencing of the activities in the workflow. The user can monitor the work of the agents (what is specific to individual negotiations) using the *Agent Monitoring and Control Tool*. Strategy rules can be entered and/or edited at run-time. Each *agent* instantiates an *SR Engine* to make inferences based on the rules found in the *SR Repository*. Note that *Negotiation 1, Negotiation 2, … Negotiation n* are the negotiations in which the agents participate. They can be conducted on one single negotiation server or on different ones as long as they are independent of each other.

5.2. Formal Description of Negotiation Rules

The transfer of the negotiation rules from the servers into the *NR Repository* of CONSENSUS as well as the instantiation of the agents by the *Agent Factory* calls for a formal description of these rules. We strongly believe that the rules describing the negotiation are as important as, or perhaps even more, than the other information describing the good or service that is the object of the negotiation. For that, they should be known to the participants (human or software) before they engage in the negotiation. We agree with Wurman et al. [19] that "implementing auction mechanisms in a larger context will require tools to aid agents in finding appropriate auctions, inform them about auction rules, and perform many other market facilitation functions". Therefore there is a need for a mechanism to formally describe the rules governing a negotiation, visualize the description when necessary, and serialize it in order to transfer it over the network. This mechanism should also make it possible to separate the description of the negotiation process from the other parts of the negotiation software, for the purpose of efficiency, reuse, and ease of testing. A review and an evaluation of the methods used to describe negotiations can be found in [20], [21].

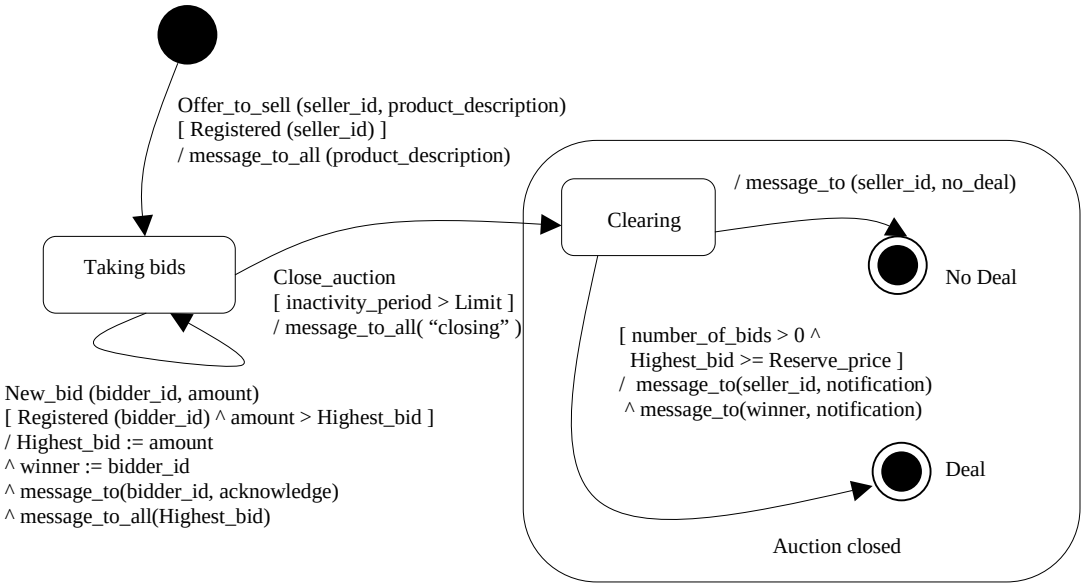

Figure 3: Statechart diagram of an English auction.

Inspired by Kumar and Feldman's Finite State Machine approach [3], we have adopted the UML's statechart diagrams [22] to describe the negotiation processes. An important feature of statechart diagrams

is the possibility to be serialized in XML [23] and XMI [18]. Moreover, simulation and analysis tools are available for statecharts, such as Statemate [24], which help validate and render the descriptions being investigated. Figure 3 shows our statechart description of an English auction. The main states of the English auction are "Taking bids" and "Auction closed". When the auction is closed, it goes to the "Clearing" state where the auctioneer has to determine if there is a deal or not. The two possible final states are "Deal" and "No Deal." In the first case the seller and the buyer are notified. In the second case only the seller is notified. The transitions are labeled with the string *event[guard-condition(s)]/action(s)*.

5.3. Negotiating Software Agents

The agents in CONSENSUS are instantiated by the *Agent Factory* and will act as participants in the workflow. Intelligent agents can be defined as software entities that execute functionalities in an autonomous, proactive, social, and adaptive fashion. These functionalities include searching, comparing, learning, negotiating, and collaborating [25]. This makes software agents particularly useful for the information-rich and process-rich environment of e-commerce [26]. They are well suited for information filtering and retrieval, personalized evaluation, complex coordination, and time-based interaction. This last feature is vital to e-negotiations. The agents in CONSENSUS are autonomous and proactive, that is, they do not require the intervention of the user as long as they are provided with strategy rules. Furthermore, they are intelligent by making inferences using the *SR Engine*. Recall that the *SR Engine* processes strategy rules, in order to decide, for instance, whether or not to make an offer and on the amount of the offer. Our agents reside on the machine that runs CONSENSUS. In contrast, the agents of the AMEC laboratory, for instance, can be mobile, which enables them to go from site to site to negotiate deals on behalf of their creator [27]. Our agents as well as those of the AMEC laboratory are used in the negotiation step (Step 4 of the CBB model), whereas several commercial products use agents in the preceding steps, such as Amazon [28] in the need identification step (Step 1), PersonaLogic [29] in the product brokering step (Step 2), and Bargain Finder [30] in the merchant brokering step (Step 3).

Parkes et al. distinguish autonomous and semi-autonomous agents: "a fully autonomous agent requires a complete set of preferences in order to represent the user correctly in all situations that it might encounter," and "a semi autonomous agent will bid on behalf of the user when it has enough knowledge to proceed, and query the user when its best action is ill-defined given the current information" [31]. A further distinction is being made between a reservation-price agent and a progressive-price agent. The first type is an autonomous agent that places bids up to the value of a fixed reservation price. The second type, on the other hand, is initialized with a lower bound and an optional upper bound on the true reservation prices of the user [31]. We intend to support in CONSENSUS semi-autonomous and fully autonomous agents, as well as the two facets of price agents.

5.4. Workflow Management

The architecture of CONSENSUS is based on three WfMS modules: the *Workflow Definition Tool*, the *Workflow Engine,* and the *Workflow Monitoring and Control Tool*. These modules correspond to three of the four components suggested by the WfMC [14]: (1) The Process Definition Tool which is used to enter the workflow into the computer, (2) the Workflow Engine which executes and tracks the workflow, (3) the Administration and Monitoring Tool used to administer and track the status of the workflow, and (4) the Workflow Client Application through which the participants interact with the workflow. We do not introduce a Client Application component, since in our solution the participants are software agents.

We rely on workflow technology because it provides support in three broad functional areas [32]: (1) workflow definition: capturing the definition of the business process; (2) workflow execution: managing the execution of the workflow processes in an operational environment, sequencing the various activities to be performed and; (3) workflow monitoring: monitoring the status of workflow processes and dynamically configuring the runtime controller. Note that these functional areas are reflected in the WfMC components and the CONSENSUS modules mentioned above.

In the architecture of CONSENSUS, the workflow captures the logic of the CN (i.e., its intrinsic and procedural constraints), whereas the agents capture the logic of the individual negotiations. The agents, by participating in the workflow, can share information and cooperate in conducting the CN. Using a workflow also makes it easier for the user to track and monitor the progress of the CN at runtime.

Furthermore, the user can adjust certain intrinsic and procedural constraints at runtime. Examples include adjusting the total price she is willing to pay at runtime, or changing the range of acceptable dates for her flight. We are also investigating the possibility to generate a workflow from a higher-level specification so that the user need not learn how to model a workflow in order to use CONSENSUS.

5.5. Negotiation Strategies and Rules Engines

According to Wong at al. [33], most current e-commerce systems use predefined and non-adaptive negotiation strategies in the generation of offers and counteroffers during the course of the negotiation. Commercial auction sites such as eBay, for instance, still require that consumers manage their own negotiation strategies over an extended period of time [34]. There is however a possibility to relieve the user from managing the negotiation. EBay, for instance, offers the possibility of "proxy-bidding" which is actually a simple agent-negotiation that uses a straightforward strategy: bid until you reach a certain amount, by going up each time with a certain increment (called the bid-increment). As another example, the buying agents in the well-known KASBAH system [27] can chose between three negotiation strategies: anxious, cool-headed and frugal, corresponding to linear, quadratic, and exponential functions, respectively, for increasing their bid for an item over time.
In the architecture of CONSENSUS, we use rule engine technology to represent and exploit negotiation strategy rules. A rule such as: "if the bidding gets too high, then abandon the negotiation" can be nicely coded as a declarative statement. According to McClintock et al. [35], "rules are declarative statements that drive activity in a software application by describing the action or actions to take when a specified set of conditions is met". The rules can be coded as stand-alone atomic units, separate from and independent of the rest of the application logic. This makes the rules easier to develop and maintain. Moreover, rule engines have special languages for writing rules.
A rule engine is a software component designed to evaluate and execute rules [35]. Rule engines are already been used in consumer profiling (also referred to as e-commerce personalization). The idea is to use collected information about consumers visiting a web site in order to trigger rules that tailor the content of a provider's site to the profiles of the consumers. The information can be collected during all steps of the CBB model (see Section 1) and, in the above example, it is used in Step 1 of the model.
A similar declarative approach is taken by Su et al. [12]: "Each strategy is expressed in terms of an Event-Trigger-Rule (ETR); which specifies the condition to be checked and the actions to be taken by the negotiation server. Rules are activated upon the occurrence of specific events during the negotiation." Instead of a rule engine, they use an ETR server [36]. The server manages events and triggers rules that are relevant to the posted event.

6. An Example of a Combined Negotiation

In this section, we illustrate how a CN is modeled using a workflow and strategy rules. Going back to the vacation package example, we start with a simple CN with no parallel negotiations. The CN is for a vacation package consisting of two items: a hotel room and a transportation ticket. Depending on the price the consumer will pay for the hotel, she either negotiates a plane ticket or a boat ticket (we suppose that it is cheaper to travel by boat). We use IBM's MQSeries WfMS [37] build-time client for modeling. Figure 4 presents the resulting workflow. There are 5 nodes that represent activities. Three activities represent three negotiations: the hotel room, the plane ticket, and the boat ticket. These activities will be carried out by software agents (note that in other domains, workflow activities are typically assigned to human agents). The two other activities deal with the success and failure of the CN, and specific programs written for that purpose will carry them out. The five activities are connected by control connectors (solid arrows) and data connectors (dashed arrows). Transition conditions are associated to control connectors. Exit conditions are associated to the three negotiation activities, which means that the activities are to be repeated until these conditions are verified. The workflow shown in the figure states the following: "Start negotiating the hotel room. Keep negotiating until you win or you lose. If you win by paying less than 100 units then start negotiating a plane ticket. If you win by more than 100 units then negotiate a boat ticket. If you lose, stop the CN."

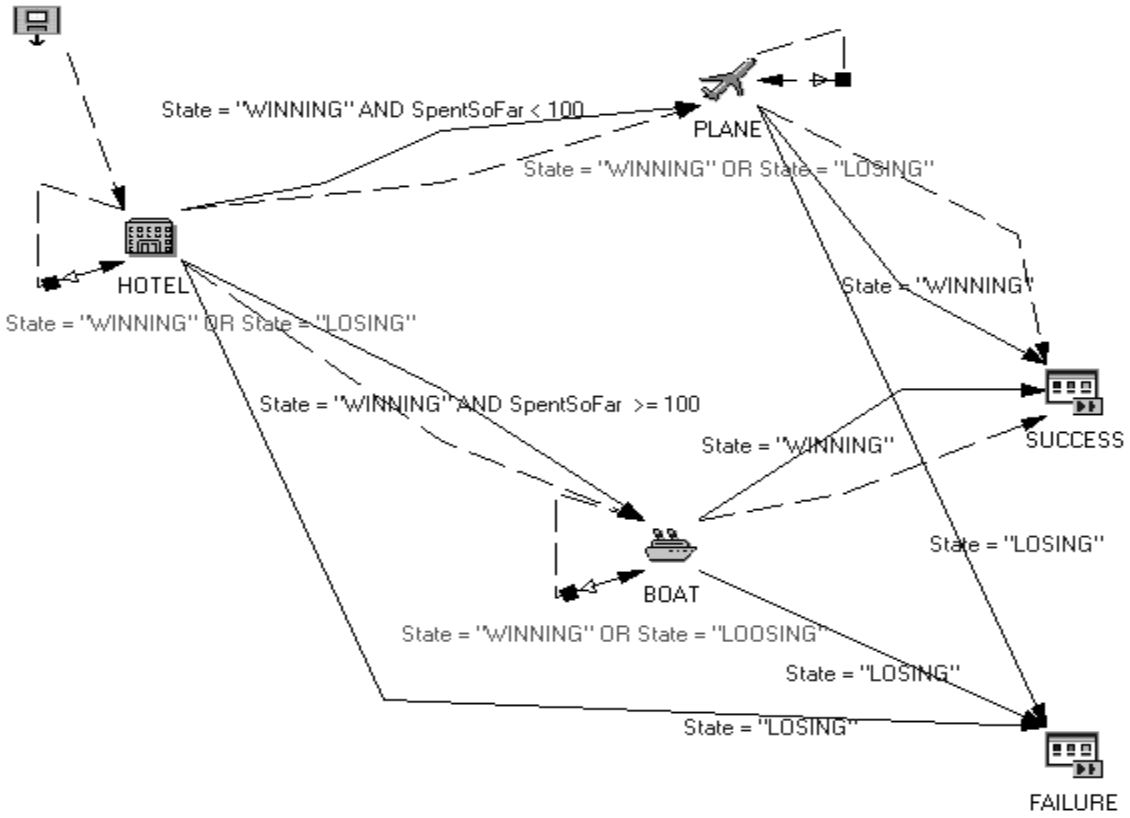

Figure 4: Workflow model 1 for the travel package.

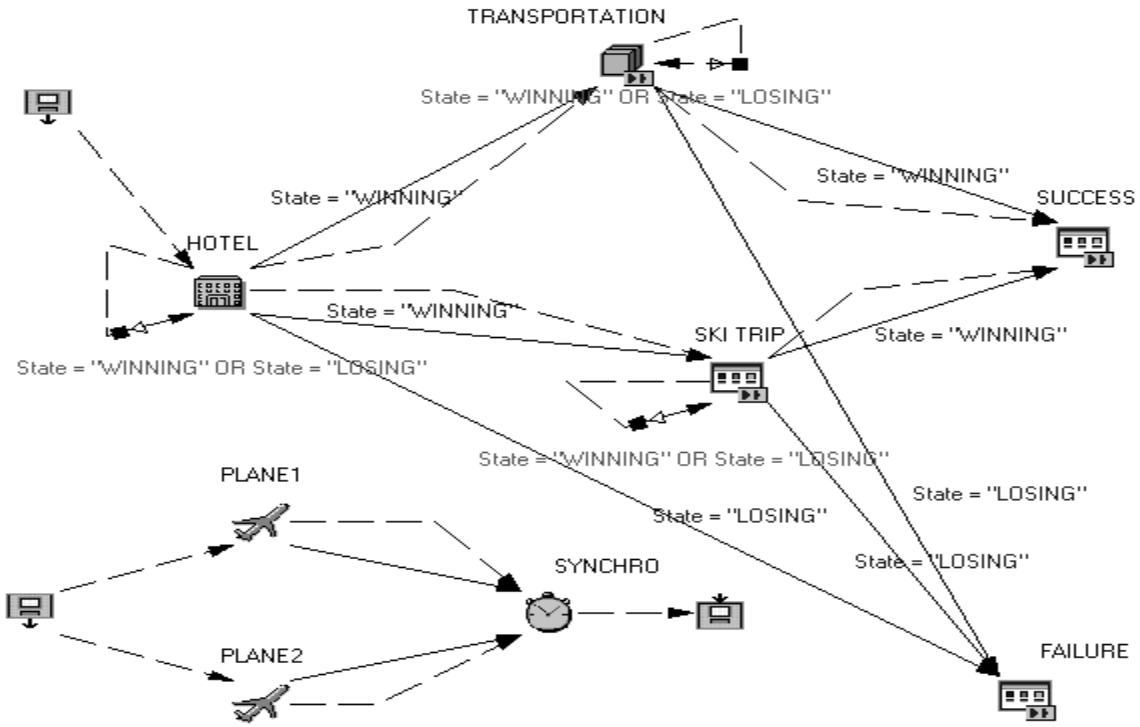

Figure 5: Workflow model 2 for the travel package.

Let us add a third item to the vacation package (a ski trip). Let us also choose to travel only by plane and introduce parallel negotiations. In Figure 5, the modeling is the same as before except for the node "TRANSPORTATION" which, in this case, is not an activity but a block of activities. A block represents a sub-process (sub-workflow). The block "TRANSPORTATION" in the figure is equivalent to the sub-workflow on the lower left corner of the figure. The main workflow states: "start negotiating a hotel room and if you win, then start the negotiations for the transportation and the ski trip at the same time. The sub-workflow "TRANSPORTATION" and the activity "SKI TRIP" will be started at the same time. That gives three negotiations going on at the same time (the sub-workflow will launch two parallel negotiations for plane tickets). Obviously, we have to be careful not to win two plane tickets. To solve this problem, we intend to use strategy rules. An obvious rule would be: "never make more than one commitment for a plane ticket at the same time." Another rule could be: "only bid in the negotiation with the smaller bid." Since the hotel room is already booked, the place and date attributes are also fixed for the other negotiations. The total amount of money to spend for the package will have to be respected. The sum of the bids in the parallel negotiations should not exceed that amount. The activity "SYNCHRO" in the sub-workflow is used to synchronize the two negotiations for the plane ticket, taking control after the two agents make a pulse.
The workflow by itself in not sufficient to model the CN. To complete it, we need strategy rules such as the one in Figure 6. The notation used is that of ILOG JRules [38] which is based on an English-like syntax. Rules have a "WHEN part" which specifies the conditions that must be met in order for the "THEN part" to be executed. The rule in Figure 6 applies to an English auction, and it states: if the participant is not leading (i.e., her bid is not the highest bid) and her reserve-price is about to be reached (i.e., the difference between the reserve-price and the highest bid is less than a certain limit) and she can reduce her increment while still respecting the minimum permitted, then she should reduce the increment by half.

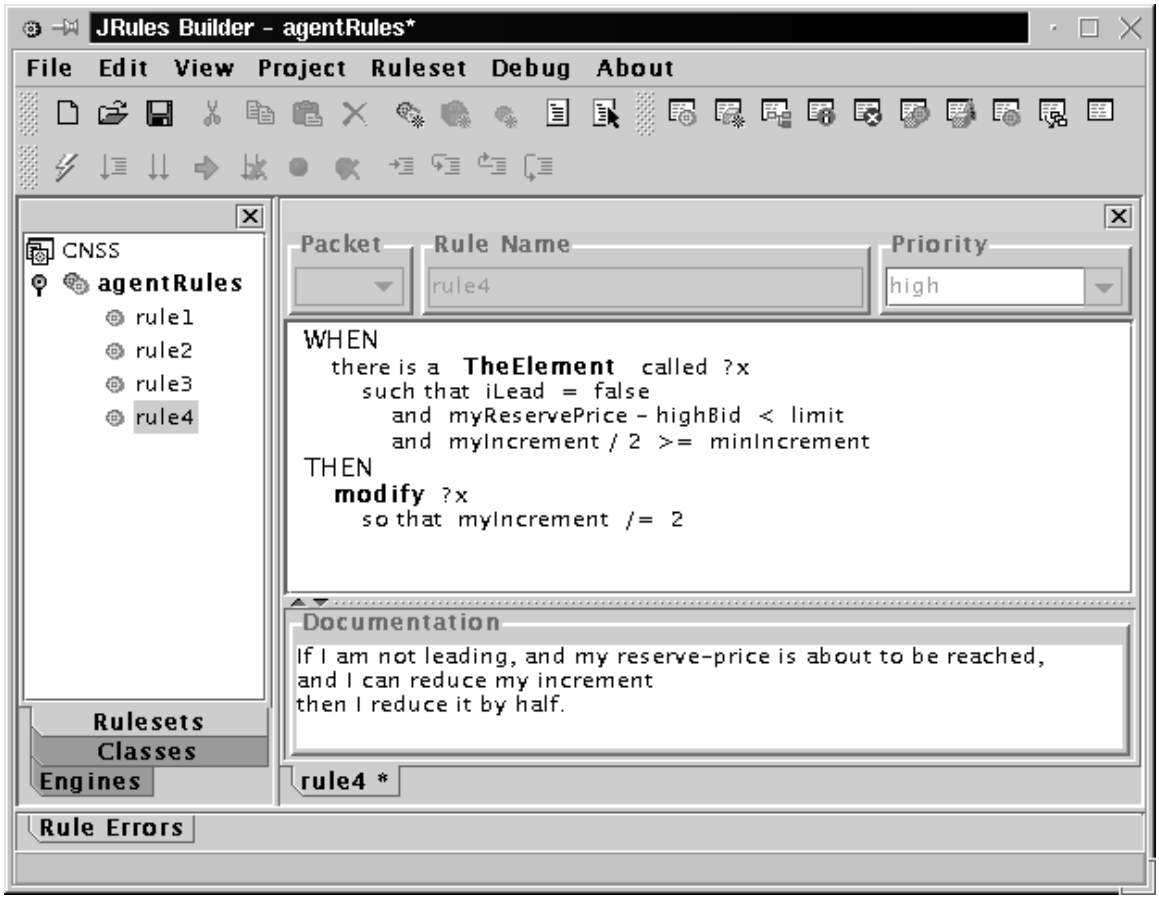

Figure 6: A sample strategy rule.

7. Implementation of CONSENSUS

Part of the presented CONSENSUS architecture has already been implemented as CONSENSUS version 0.1. This version includes a WfMS, software agents and an Agent Monitoring Tool. Instances of our negotiation server GENESIS are used by CONSENSUS to conduct individual negotiations. Only one negotiation type is supported in this version (the English auction). It is easy to implement new negotiation types on GENESIS since all there is to do is to write a new script describing the new negotiation type (called the auctioneer in GENESIS terminology). The agents function in pulses (also called episodes). A pulse is an atomic action by the agents, which includes getting the information from the server, deciding what to do, and finally taking action. The concept of pulse adds transparency to the negotiation process. The WfMS gets the control back after each pulse by the agents. The exchange of information between the agents and the servers is done via XML documents (the order, the quote, the adjudication, etc.). We use the IBM MQSeries WfMS for this version and rely on its build time client for modeling and on its run time client for executing instances of the workflow model and for monitoring. There is no use of negotiation strategies in this version, and consequently, the second action of the pulse (deciding) will be performed by the user.

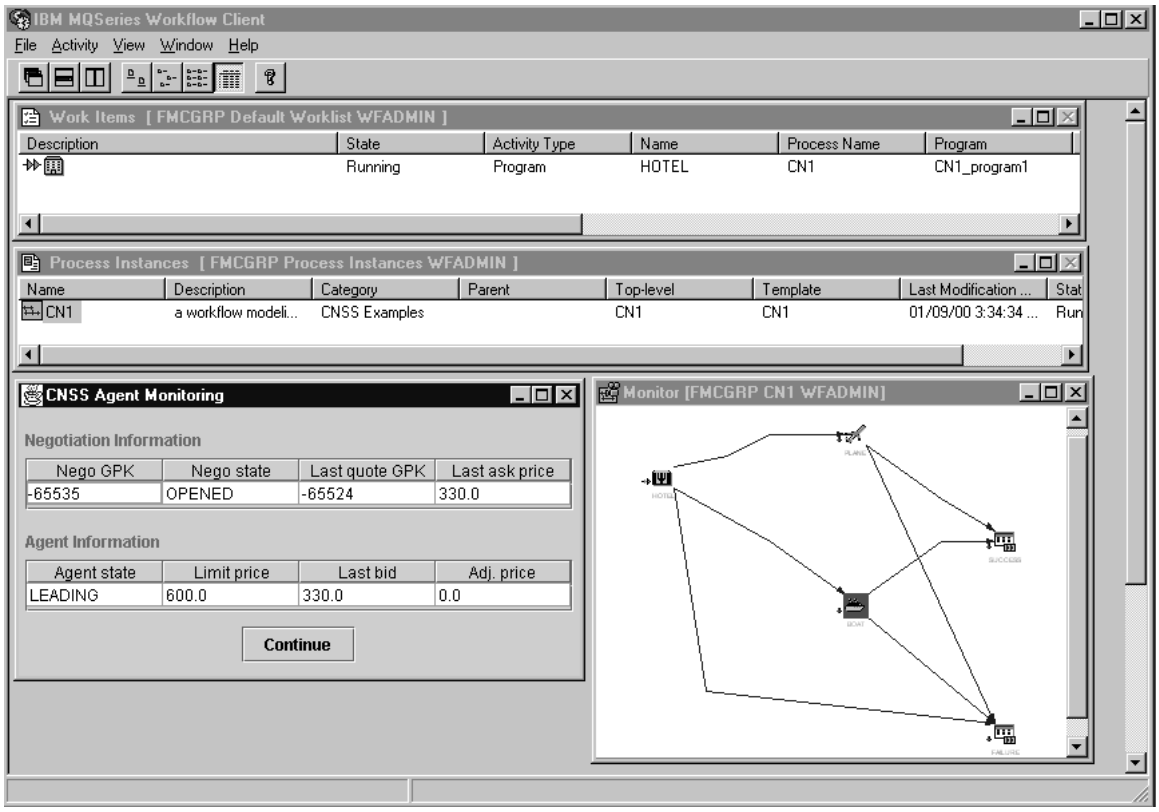

Figure 7: Screenshot of CONSENSUS version 0.1.

In a standalone fashion, we have incepted integrating rule engine technology into our negotiating agents. As a rule engine, we are using ILOG's JRules. The integration with CONSENSUS will come later, once we are satisfied with the results of using negotiation strategies in individual negotiations. Since strategies for whole CNs are much more complex than those for individual negotiations, we will deal with them in later versions.

In Figure 7, we present a screenshot of CONSENSUS version 0.1. The main window is that of the IBM MQSeries WfMS runtime client. The window at the top shows the work items (In case of manual processing, the human will have to consult this window to know what to do. In our case, the agents are activated automatically). The second window shows the instance of the workflow that is running. The third

window (bottom left) shows the Agent Monitoring and Control Tool displaying information about one of the individual negotiations that is going on. The last window (bottom right) is the monitoring window of the IBM MQSeries WfMS.
CONSENSUS is designed such that the currently used WfMS may be substituted for another WfMS. To validate this design, we are currently working on a new version that uses BEA's WLPI system [39].

8. Related Work

In this section, we present the two approaches that we think are most closely related to the CONSENSUS approach.
The approach by Su et al. [12] is based on the idea that a consumer registers on a proxy negotiation server by giving a description of the goods or services she wants, her preferences, and a negotiation strategy. Then the server takes over and looks for a supplier that matches the consumer. When it finds one, it starts a bargaining type negotiation until eventually a deal is reached. The server uses the negotiation strategy supplied by the consumer. Notice that in this case, the merchant brokering phase is the responsibility of the server, and that the user has no control over the negotiation. Many negotiations can be started on the same proxy server, and they will be stored as persistent objects in a database.
With this solution, the user will have no control over the negotiations once they are started. She can only see their progress, but cannot intervene in them. We prefer the user to be in control and intervene whenever she wants in the CN. The workflow representing the CN would have to be constructed, stored, and executed at the negotiation server (a feature that is not supported by the proxy server as it is described in [12]). We prefer the workflow representing the CN to be constructed, stored, and executed on the client side rather than on the negotiation server side. In contrast to our solution, the proxy negotiation server approach does not explicitly support CNs. Furthermore, our solution is decentralized and supports any type of negotiation, whereas the server solution supports only bilateral bargaining.
When we compare the services expected from CONSENSUS to those offered by the KASBAH system [27], we see some similarities. The agents can be monitored by their creator via GUI tools. They also can have a negotiation strategy. For KASBAH, the strategy is rather simple and is chosen from a set of predefined strategies. In CONSENSUS, however, the strategy is entered rule by rule. Consequently, it can be more sophisticated and may not be guessed easily by the server or the other participants. In the KASBAH system, for instance, it is possible to guess the negotiation strategy of an agent just by observing it for a while. An agent for which the strategy is known by its opponents is at a disadvantage [40]. The agents in KASBAH are mobile, whereas ours are not. The agents in KASBAH conduct only one style of negotiation, which is bargaining, whereas ours will practice any kind of negotiation.

9. Conclusion

In this paper, we stated the problem of a CN and showed the need for a CNSS to solve it. A CN involves negotiating many interdependent items by engaging in different and independent negotiations. CONSENSUS is a CNSS which helps the user model, track and monitor a CN. Its architecture is based on agent technology, as well as workflow management and rule engine technology. CONSENSUS can be used by a consumer from her home computer to negotiate, for instance, a vacation package (B2C), or it can be used in B2B, for example, by a travel agency to negotiate travel packages for its customers.
We claim that a CN can be specified using a workflow. We are certain that a workflow can capture the sequencing of the negotiations and some dependencies between them (we call this the CN know-how), and currently, we are experimenting with more complex examples to further substantiate our claim. The software agents know the rules of the individual negotiations in which they are involved (we call this individual negotiation know-how). But knowing the rules of the negotiation does not make you a good negotiator. What is needed are negotiation strategies to be used by the agents in order to make them better negotiators (we call this negotiation strategy know-how).
As future work, we aim to further investigate several research issues that are central to our solution. One such issue is how to instantiate software agents based on a description of the negotiation rules. Another

issue is the high-level specification language we would like to define and implement in order to free the user from the task of modeling a workflow. The question of breaking commitments in a negotiation will also be investigated along with its effect on CNs. Negotiation strategies are an important research area for us. We will have to come up with strategies that apply to individual negotiations and strategies that apply to a CN. We will also have to see to what extent the negotiation strategies depend on the negotiation type. Our conceptual work will be completed by proof-of-concept prototypes, as incepted with CONSENSUS version 0.1.
In conclusion, we believe that a CNSS such as CONSENSUS is a much-needed tool for coping with the ever-increasing scope and complexity of e-negotiations.

Acknowledgment: We would like to thank our colleagues Robert Gérin-Lajoie, Jacques Robert, and Vincent Trussart for their contributions to the CONSENSUS research.

References

[1] The AMEC Laboratory. http://ecommerce.media.mit.edu/.
[2] P. Maes, R. H. Guttman, and A. G. Moukas. Agents that Buy and Sell: Transforming commerce as we know it. Communications of the ACM, March 1999.
[3] M. Kumar and S. I. Feldman. Internet Auctions. Technical Report, IBM Institute for Advanced Commerce, Yorktown Heights, NY, November 1998.
[4] EBay Auctions. http://www.ebay.com.
[5] Onsale Auctions. http://www.onsale.com.
[6] Yahoo Auctions. http://www.auctions.yahoo.com.
[7] http://www.agorics.com/new.html.
[8] T. Sandholm. An Algorithm for Optimal Winner Determination in Combinatorial Auctions. In IJCAI, pp. 542-547, Stockholm, Sweden, 1999.
[9] OMG Negotiation Facility. http://www.oms.net/ecdtf.html.
[10] The Enegotiations Web Page: http://enegotiations.wu-wien.ac.at/.
[11] A. Jhingran. The Emergence of Electronic Market Places, and other E-Commerce Directions. Handout at Workshop on Electronic Marketplaces held at Cascon'99, Toronto, ON, Canada, November 1999.
[12] S. Y. Su, C. Huang, and J. Hammer. A Replicable Web-based Negotiation Server for E-Commerce. In Proceedings (CD_ROM) of 33rd International Conference on System Sciences, Hawaii, 2000.
[13] M. Benyoucef, R. K. Keller, S. Lamouroux, J. Robert, and V. Trussart. Towards a Generic E-Negotiation Platform. In Proceedings of the Sixth Conf. on Re-Technologies for Information Systems, pp. 95-109, Zurich, Switzerland, February 2000.
[14] The WfMC. http://www.aiim.org/wfmc/mainframe.htm.
[15] T. Sandholm and Victor Lesser. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. In International Conference on Multi-Agent Systems, pages 328-335, San Fransisco, CA, 1995.
[16] W. Y. Wong, D. M. Zhang, M. Kara-Ali. Negotiating with Experience. In Proc. of KBEM'00. Austin, Texas. July 2000.
[17] G. Lo and G.E. Kersten. Negotiation in Electronic Commerce: Integrating Negotiation Support and Software Agent Technology. In Proceedings (CD-ROM) of the 29th Atlantic Schools of Business Conference, Halifax, NS, 1999.
[18] XMI. ftp://ftp.omg.org/pub/docs/ad/98-10-05.pdf.
[19] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The Michigan Internet AuctionBot. In Second Intl. Conference on Autonomous Agents, pp. 301-308, Minneapolis, MN, May 1998.
[20] A. G. Cass, H. Lee, B. Staudt Lerner, and L. J. Osterweil. Formally Defining Coordination Processes to Support Contract Negotiation. Technical Report UM-CS-1999-039, University of Massachusetts, Amherst, MA, June 1999.
[21] M. Benyoucef and R. K. Keller. An Evaluation of Formalisms for Combined Negotiations in E-Commerce. In Proceedings of the Workshop on DCW, pages 45-54, Quebec City, QC, Canada, June 2000.Springer. LNCS 1830.

[22] J. Rumbaugh, I. Jacobson, and G. Booch. The Unified Modeling Language Reference Manual. Addison-Wesley. 1999.
[23] XML. http://www.w3.org/TR/1998/REC-xml-19980210.
[24] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, and M. Trakhtenbrot. STATEMATE: A Working Environment for the Development of Complex Reactive Systems. IEEE Transactions on Software Engineering. 16(4):403-414, 1990.
[25] K. Jonkheer. Intelligent Agents, Markets and Competition. Electronic publication. http://www.firstmonday.com. 1999.
[26] A. Moukas, R. Guttman, and P. Maes. Agent-mediated Electronic Commerce: An MIT media laboratory perspective. In International Conference On Electronic Commerce, ICEC'98, Seoul, Korea, April 1998.
[27] A. Chavez and P. Maes. KASBAH: An Agent Marketplace for Buying and Selling Goods. In 1st Intl Conf on Practical Appli. of Intelligent Agents Technology, London, UK, April 1996.
[28] Amazon Website. http://www.amazon.com.
[29] Personalogic Website. http://www.personalogic.com.
[30] Bargainfinder Website. http://www.bargainfinder.com.
[31] D. C. Parkes, L. H. Ungar, and D. P. Forster. Agent Mediated Electronic Commerce, chapter Accounting for Cognitive Costs in On-line Auction Design, pages 25-40. Springer-Verlag, 1999.
[32] The Workflow Automation Corporation. New Opportunities for Dramatic IT Results, a white paper available at http://www.workflow.ca. 1998.
[33] W. Y. Wong, D. M. Zhang, M. Kara-Ali. Negotiating with Experience. In Proc. of KBEM'00. Austin, Texas. July 2000.
[34] P. Maes, R. H. Guttman, and A. G. Moukas. Agents that Buy and Sell: Transforming commerce as we know it. Communications of the ACM, March 1999.
[35] C. McClintock and C. A. Berlioz. Implementing Business Rules in Java. In Java Developers Journal, pages 8-6, May 2000.
[36] H. Lam and S.Y.W. Su. Component Interoperability in a Virtual Enterprise Using Events/Triggers/Rules. In Proceedings of OOPSLA '98 Workshop on Objects, Components, and Virtual Enterprise, Oct. 18-22, 1998, Vancouver, BC, Canada.
[37] IBM MQSeries Workflow. http://www-4.ibm.com/software/ts/mqseries/workflow/.
[38] ILOG JRules. http://www.ilog.com/products/jrules/.
[39] BEA WebLogic Process Integrator. http://www.beasys.com/products/weblogic/integrator/.
[40] C. Beam and A. Segev. Automated Negotiations: A Survey of the State of the Art. Technical Report 97-WO-1022, Haas School of Business, UC Berkeley, 1997.
[41] M. Benyoucef and R. K. Keller. A Conceptual Architecture for a Combined Negotiation Support System. In Proceedings of the Eleventh International Workshop on Database and Expert Systems Applications (DEXA 2000), pages 1015-1019, Greenwich, London, Britain, September 2000. IEEE. Presented in W17: Workshop on Negotiations in Electronic Markets - Beyond Price Discovery - E - Negotiations.