Component-based Software Protocol Approach

Jin Baek Kim[1] Gregory E. Kersten[2] Stefan Strecker[3] Ka Pong Law[2]

[1] Concordia Institute for Information Systems Engineering, Concordia University
[2] John Molson School of Business, Concordia University
1455 de Maisonneuve Blvd. W. Montreal, Quebec H3H 1M8, Canada
{jbkim, kersten, kplaw}@jmsb.concordia.ca

[3] Information Systems and Enterprise Modelling, University Duisburg-Essen
Universitaetsstr. 9, D-45141 Essen, Germany
strecker@uni-duisburg-essen.de

Abstract

A major challenge in developing an e-negotiation system (ENS) is that the context of negotiations such as negotiators' characteristics, negotiation processes, negotiation rules, and social implication are different case-by-case. This context dependency makes it difficult to develop a general ENS applicable to wide variety of negotiation problems. In this paper, in order to mitigate the context dependency issue, we propose to adopt the component-oriented software protocol approach to e-negotiation systems and present a framework for e-negotiation protocols that implements this approach. According to this framework, an ENS is developed by designing a high level e-negotiation protocol which specifies the rules on allowed activities at a certain state and the rules on how to change them depending on the activities performed. Then, this designed e-negotiation protocol is executed by a general purpose ENS platform, which integrates software components and executes the protocol. This approach allows one to easily develop or modify ENS so that it can best fit into the context. We prove validity of our framework by redeveloping two existing ENS's - SimpleNS and Inspire – using the framework for e-negotiation protocol model and an ENS platform that understands and executes the defined protocol.

Keywords: e-negotiation systems, context dependency, negotiation protocol, negotiation process design, component-based approach, software protocol

1. Introduction

Activities involved in a negotiation can be classified into intrapersonal and interpersonal. Negotiators may have a great degree of freedom in choosing intrapersonal activities (e.g. preparing a negotiation, creating and evaluating the offers), while they should follow the rules explicitly or implicitly agreed with the counterparts in selecting intrapersonal activities (e.g. sending the offers and messages). In face-to-face (F2F) negotiations, the models of intrapersonal activities and the rules of interpersonal activities are often implicit. In system supported negotiations, however, these models and rules of activities should be explicitly considered and specified so that the system can effectively and efficiently support decision making and communications in the negotiation.

Every negotiation which involves the use of information and communication technologies distributes the work between its people and the technology. This allocation of work has to be agreed upon by the parties. Even the use of a simple solution like email requires the negotiators' prior agreement that they communicate via email rather than face-to-face, snail-mail, fax or telephone. In other words the parties agree on the use of certain rules governing some or all aspects of the negotiation. What rules need to be agreed upon and what is their potential impact on the negotiation processes and outcomes becomes one of the problems that researchers and practitioners alike are concerned. One reason for the failure of the first generation of firms providing e-negotiation services which were established in the late 90s (e.g., Ozro, Prowess and Tradeaccess) may be due to the rejection of the rules governing the division of labour between people and software.

Negotiation protocol is a set of rules governing the intrapersonal and interpersonal activities in negotiations. E-negotiation protocol, a negotiation protocol for e-negotiation systems (ENS), is a set of the rules that control the interactions between the negotiators and ENS as well as the behaviour of the ENS. E-negotiation protocols should contribute to achieve a better agreement by encouraging negotiators to employ verified methods, to follow best practices, and to provide partial or full automation (Vetschera et al. 2003).

Behavioural research posits that relevant negotiation activities depend on the negotiators' characteristics and the negotiation context (e.g. problem structure, process of negotiation, relationship between negotiators, etc.). These characteristics and context determine negotiators' approaches, strategies and tactics leading to the selection of specific activities in the different phases of a negotiation.

Although a great deal of insight is provided by behavioural research, incorporating all these insights into the ENS is difficult because of the huge number of possible combinations of the negotiator's characteristics, dependence of the negotiators' behaviour on external factors (e.g., relationship with other stakeholders, competing decision problems and the consideration of future situations), as well as the complexity of the problem and process (Kersten 2005).

One possible approach to mitigate this context dependency issue is to separate conceptual e-negotiation protocol model from the system platform that executes it. Through this way, the most appropriate e-negotiation protocols can be created new or by modifying existing e-negotiation protocols. In order to achieve this, the conceptual protocol model should be abstract enough to help users focus only on essentials while it should contain the details enough to be deployed on ENS as far as it can understand the protocol model.

Such a separation is especially desirable if the contexts are rather significantly different. The ENS should be applicable and appropriate to many different contexts without changes as far as the difference is minor. In order to accommodate minor difference in contexts, the conceptual e-negotiation protocol should allow defining protocols that give some degree of freedom in selecting activities. For example, during the process, the negotiators may wish to review the problem, modify their preferences, and add or remove issues. The freedom of performing these activities should be allowed to fully fit into the context. On the other hand, however, the e-negotiation protocol should be able to force the negotiators to undertake certain activities in order to reach a better outcome. For example, forcing the negotiators to learn about the negotiation problem, consider their own objectives and preferences, and evaluate the counterpart's offer before making their own offers can improve the negotiation outcome by encouraging the negotiators to make informed decisions. In these situations, the selection of a particular activity opens a new path of activities which have to be contiguous; every possible path selected by the user and/or ENS has to be connected and geared towards the desired negotiation outcomes.

The dilemma between restricting the bargainers' possible activities and providing the flexibility in activities is the main issue of negotiation protocol design. The more activities the system mediates and the more activities the system undertakes autonomously, the more restrictions are imposed on the negotiators' own choices. The framework for e-negotiation protocols should allow both considerations to be incorporated when designing the e-negotiation protocols.

In this paper, we focus on the framework for modeling e-negotiation protocols that specify both intrapersonal and interpersonal activities. The framework is conceptual enough to separate the protocol design task from system development task, but contains the details enough to be deployed and executed on a general purpose ENS that understands the framework. The framework allows to model e-negotiation protocols that provide certain degree of freedom in user's activities while offering mechanisms to force activities. The formal representation of e-negotiation protocols provided in the framework is a further refined version of authors' earlier work on e-negotiation protocols (Kim and Segev 2003; Kersten 2004; Kersten and Lai 2005).

The organization of this paper is as follows. In the next section, related literature is reviewed. Then, we present a framework for modeling deployable e-negotiation protocols which adopts component-oriented software protocol approach. In Section 4, we validate the framework by illustrating redevelopment of two existing e-negotiation protocols using the framework. Finally, we summarize our contributions and discuss future work.

2. Literature review

Negotiation support systems (NSS) are a class of group decision support systems (GDSS) designed to support the negotiation activities of two or more parties in reaching an agreement in situations of contradicting interests (Jelassi and Foroughi 1989). On a conceptual level, NSS consist of an individual decision support systems (DSS) for each party and an electronic communication channel between the parties (Lim and Benbasat 1992). Interactive, session-oriented (comprehensive) NSS simultaneously support the entire negotiation process of all parties and enable the parties to communicate directly with each other (Foroughi 1995). Functionalities of comprehensive NSS include decision support in negotiation preparation (e.g. formulation of the negotiation problem, modeling of preferences), in negotiation execution (e.g. evaluation of offers, graphical representation of information), and in post-settlement activities (e.g. analysis of outcome efficiency) in addition to the underlying electronic communication channel (Kersten and Noronha 1999).

Electronic negotiation systems or e-negotiation systems (ENS) are NSS which use internet technologies for communication such as web-based NSS (Kersten and Noronha 1999) as well as NSS based on email and other internet technologies, e.g. chat, streaming audio and video (Bichler 2003).

The use of internet technologies has increased the popularity and diffusion of NSS in many applications domains from personal negotiations to corporate procurement to legal dispute resolution (Neumann et al. 2003, Yuan, 2004 #9).

Despite wide variety in approaches, most ENS's have been developed with a specific negotiation protocol in mind. Often, the protocol has been informally described as a part of cases or system instructions. The lack of explicit and formal modeling of negotiation protocols eventually restricted system's capability to allow users to create new protocols or modify existing protocols (Kim and Segev 2003).

Methodologies for modeling various business processes and developing systems supporting these processes have been the main research issues of the workflow research community for more than a decade (Stohr and Zhao 2001). Commercial vendors, consortia, and academic society of workflow and business process management system provided various methods of defining workflows and business processes. Workflow Management Coalition (WfMC) proposed XPDL (XML Process Definition Language) for defining the processes(WfMC). Microsoft proposed XLANG and IBM proposed WSFL as the business process definition language of their products – BizTalk and MQSeries. The most recent version of BPEL4WS was proposed by IBM, Microsoft, BEA, SAP, and Siebel Systems in 2003. Its main focus is to define Web Services-based executable business processes (BPEL4WS 2003). According to Wohed et al. (2002), BPEL4WS has expressive power which is the union of XLANG and WSFL. Most of them are based on Petri-nets or block-based language with control flows (Aalst and Hofstede 2005). Aalst (2005) provides the survey of expressive power of standards and commercial systems from the perspective of control-flow, data, and resource.

Benyoucef et al. (2001) and Bassil et al. (2002) approached ENS from the workflow management perspective. They studied negotiation systems that help negotiators to coordinate interdependent negotiations for two or more items. They used workflow management systems for building systems for such combined negotiations. Negotiations that run on this system are based on auction type protocol in which workflow management system is used to deals with interdependency.

Despite remarkable achievements of process models in the workflow management area, the unique nature of negotiation processes prevents us from applying the models, techniques and algorithms to ENS development. First, while ENS should deal with unstructured cyclic activities, workflow systems have focused on repetitive execution of a highly structured flow of activities and on the data and resources for performing the activity. Second, a process in a workflow system usually assumes cooperative relationships among users while in most negotiations, the relationships between negotiators are somewhere in-between competition and cooperation. Third, multi-party negotiations require complex interleaving of many instances whose number is not known. Aalst et al. showed that such type of processes are not well supported in many (commercial) workflow management systems as well as standards.

Holsapple et al. (1998) and Bichler et al. (2003) provided a generic framework for modeling negotiations and developing negotiation support systems developing negotiation support and e-negotiation systems. However, these frameworks do not cover the details required for deployable e-negotiation protocols.

Kim and Segev (2005) investigated Web Services as implementation technology of ENS and BPEL4WS as modeling tool for negotiation processes. They showed a very structured negotiation process—alternating offer negotiations—can be represented by BPEL4Ws and proposed market-based process management architecture, but found problems in modeling less structured negotiations using the BPEL4WS and other workflow oriented technologies.

Kersten and Lai (2005) proposed a formal representation of e-negotiation protocols as well as properties and desiderata of the protocols. As stated in the previous section, this work has been a starting point of this paper while it is refined based on the insights and lessons we acquired while modeling various e-negotiation protocols and developing a generic e-negotiation platform called Invite system.

From the research methodology perspective, in the sense that the work in this paper is an outcome of designing ENS and efforts to improve the development process, the proposed framework is science of the artificial or design research discussed by Vaishnavi and Kuechler (2005). March and Smith (1995) propose four general outputs of design research: constructs, models, methods, and instantiations. Constructs are the conceptual vocabulary of a problem/solution domain. A model is a set of propositions or statements expressing relationships among constructs. A method is a set of steps used to perform a task. An instantiation is an operationalization of constructs, models, and methods. We adopt this perspective and provide constructs, models, methods, and instantiations for designing deployable e-negotiation protocols.

3. Framework for e-negotiation protocols

This paper focuses on how to model and represent e-negotiation protocols: negotiation protocols which are deployable and executable on ENS. The ENS considered in this paper are the ones used by humans where the key negotiation activities such as sending and reading offer are not automated, although auxiliary activities which traditionally have been conducted by software, such as computation of utilities, simulations, scenario generation, remain conducted by software. In future research, this assumption may and will be relaxed. At this stage, however, we consider systems which allow and help people to conduct negotiations on the web.

The proposed framework of e-negotiation protocols adopts the perspective of component based systems and software protocol discussed in Section 3.1. The constructs defining the terms and notations for e-negotiation protocols, and the model describing the relationships among the constructs are presented in Section 3.2. In Section 3.3, common patterns in the negotiation protocol and methods of implementing them are given. In Section 3.4, the table-based representation and instantiation of constructs, models, and methods are explained.

3.1 Component-based systems and software protocol

The main purpose of e-negotiation protocol is to organize interactions between the negotiators and ENS, i.e., negotiation activities performed via web-based negotiation systems. Activities can be modeled at various level of abstraction. Therefore, it is important to define the level of activity that is used as the basic unit when modeling e-negotiation protocols.

The World Wide Web is designed based on the metaphor of pages and hyperlinks (Berners-Lee and Cailliau 1990). Therefore, page becomes an easily identifiable unit of activity in web-based applications such as e-negotiation systems, and we view the page as a unitary medium or an identifiable interaction element. Adopting this perspective, we model the e-negotiation protocol by decomposing the negotiation into the level of activities that can associate with a page. In other words, the atomic unit of activity in the e-negotiation protocol model is the page-level activity and e-negotiation protocol is concerned with organizing pages.

Developing e-negotiation systems based on the component-oriented approach allows us to take advantage of many well-known merits in component-based systems such as flexibility in the system, easy maintenance and upgrade, reducing the development cycle time by reusing existing components, and so on. We focus on the e-negotiation protocols for component-based ENS.

Component is a relatively defined term. Sometimes a whole system may mean a component while a component may refer to a small piece of code. The granularity of components to be used by the e-negotiation protocol depends on the activities modeled in the e-negotiation protocol because the foremost goal of the e-negotiation protocol is to support negotiation activities. Considering the page-level as the granularity of the components considered in the protocol, e-negotiation protocol can be viewed as an example of a software protocol that controls page-level components.

A software protocol is a set of rules that determine what or how unrelated objects or components communicate with each other (Wikipedia). The software protocol for component-oriented systems determines which component is executed and in what order. The software protocol also decides what is to be done when a component's execution results in success or in failure; it invokes another component to be executed or—if it cannot determine a component—chooses one of the "fall-back alternatives" (e.g., it terminates the system's execution). In the system architecture separating the data from the logic through the database, the role of the component is to read data from and to write data to the database, to present data to the user, or to perform processing (e.g. image generation), and to inform the protocol whether it executed its task with success or failure. An example of the software protocol controlling the execution of four components (A, B, C, and D) is presented in Figure 1.

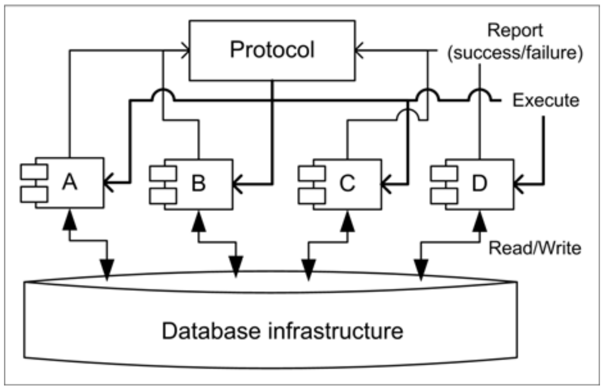


Figure 1. Protocol, components and database infrastructure

Viewing the e-negotiation protocol as a software protocol, the protocol is the set of rules allowing, forbidding, and forcing page-level negotiation activities, which are specified by the rules on enabling, hiding, and invoking page-level components.

It should be noted that activities and components defined at the page-level can be decomposed into minuscule ones. In other words, an activity on one page may be broken into several smaller elements (i.e. actions), and likewise, a component for a page may be decomposed into smaller components (i.e. sub-components). For example, the construct offer page may comprise the form component which writes the submitted offer to the database and the display component that reads the offer from the database and presents it. On such a page, a user can both read the most recent offer received and construct a counteroffer. This case shows that the page-level construct offer activity can be decomposed into the minuscule actions of read offer and write offer, and the component for offer construction page should contain the offer display and the offer construction form sub-components.

By modeling the page-level activity and making the page-level component as an atomic unit of the protocol, the framework leaves the detailed composition of the page as the task of component developers. There are advantages in doing so. First, modeling detailed actions composing each page can be left out of the scope when designing e-negotiation protocol. Therefore, the protocol designer can focus on high level coordination among page-level activities and components. Second, the features on individual pages can be upgraded or changed without changing defined e-negotiation protocol using them. Third, component designers can have more flexibility in designing and implementing components without deep knowledge and concern about the e-negotiation protocol.

3.2 Constructs and model

As explained in the previous section, we view the e-negotiation protocol as a software protocol and a triple of activity-page-component (i.e. the atomic units of activity and component are at the page-level) as the basic concept in e-negotiation protocols. More formally, we define components as follows.

Definition 1. Component cj, ($j \in J$, J – set of component indices) is the software module associated with composing a page that supports activity of the user on a page.

As can be seen in the definition of the component, we are not concerned here with such issues as software decomposition for the purpose of its reusability, and the specification of separable entities such as objects. Instead, the e-negotiation protocol in our perspective is based on the natural and intuitive unit of activity and component in web-based applications – the page.

Although there is no one process model that fits every negotiation, negotiations generally progress in phases. Gulliver (1979) proposed a model of negotiation composed of eight phases which Kersten (1997) suggested to reduce to the following five phases: (1) planning, (2) agenda setting and exploring the field, (3) exchanging offers and arguments, (4) reaching agreement, and (5) concluding the negotiation. In negotiations, of course, one phase is often revisited from a later phase. For executable e-negotiation protocols, these phases should be divided into smaller entities we call activities. For example, the planning phase includes activities such as gathering information, choosing potential negotiation partner, designing strategies, and others.

The phase models suggest that activities can be clustered naturally. Consider for example, the offer exchange phase. When a user makes an offer, he may need to read the transcript of the previous offers in order to refresh his memory. In order to support this, when the user is reading the transcript page (i.e. the component for displaying the transcript page is invoked), the component for the write_offer page should be accessible, so that after the user reviewed the transcript he can return to the write_offer page. In the offer exchange phase, it is also natural to allow the read_offer page if there is a received offer. These allowed activities in the offer exchange phase are very different from earlier phases (e.g., setting the agenda) or later phases (e.g., after reaching an agreement)

To account for the possibility of grouping related negotiation activities, we introduce the concept of a sequence which allows clustering components for those activities together.

Definition 2. Sequence $\sigma_i = \{c_j , j \in J_i)\}$, (where $i \in I$; I is the set of sequence indices) is a set of interrelated components which together determine all possible activities that can be undertaken in a given negotiation situation.

The case depicted in Figure 2 gives one possible example. By clustering components into a sequence, a component becomes accessible whenever one of the other components in the same sequence is invoked and being executed. By clustering components, the protocol need not be concerned with ordering every component. Instead, as shown in Figure 2, the protocol generally tries to operate on sequences. It executes only one component from the sequence and only one component in the sequence has to report the success or failure of the sequence execution. Other components in the sequence report only failure; this case is indicated in Figure 2. Alternatively, they may report failure to one selected component in the sequence, which, however, has to have a capability to redirect the execution flow.

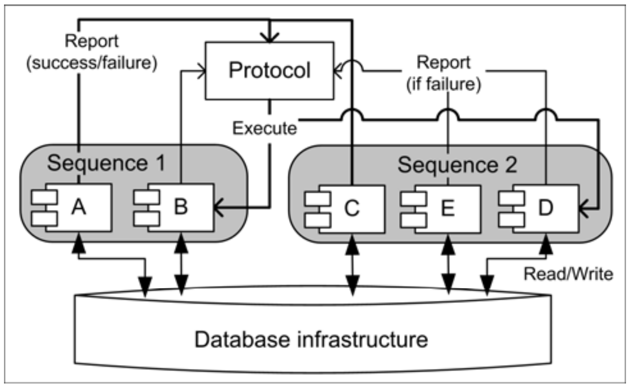


Figure 2. Protocol, sequences, and components

Because the same component (or page) may be required and used at different steps of the negotiation, a component should be allowed to use in more than one sequence.

Relationship 1. Every sequence has at least one component, that is, min $|\sigma_i|=1$, $i \in I$. It is possible that one component belongs to more than one sequence, that is, $c_j \in \sigma_i$ and $c_j \in \sigma_k$, where $j \in J$; $i, k \in I$; $i \neq k$.

The e-negotiation protocol also needs to control routing from one sequence to another. Possible moves among sequences are described by directed links between sequences, called exit link. These links indicate the possible sequences a user can move when the user is in a specific sequence. From the web-based system perspective, on a page, the exit link should show up as a link in addition to the links that make other components in the sequence accessible.

Definition 3. Exit link ρ_{ij}: $\sigma_i \rightarrow \sigma_j$ ($i, j \in I$), is a binary relation between two sequences σ_i and σ_j, indicating a user can move from σ_i to σ_j.

Examining many patterns in the protocols, we found that a user should have option to move to more than one sequence. Also, a sequence that does not have any exit link should be allowed because such dead-end sequences are useful for modeling the final steps of the negotiation. We thus distinguish three types of sequences:

1. One sequence that determines the beginning of the negotiation: σ_i, ($i \in I_1$; $|I_1| = 1$; $I_1 \subset I$). There can be only one such sequence so that the negotiation starting point is deterministic;

2. Sequences which lead to the end of the negotiation: σ_i, ($i \in I_2$; $I_2 \subset I$); which can be agreement or disagreement, termination imposed by one side, or some other final point; and

3. Sequences which are in-between the beginning sequences and ending sequences: σ_i, ($i \in I_3$; $I_3 \subset I$).

Note that there are no other types of sequences than the beginning, in-between and ending, that is,

$$I_1 \cup I_2 \cup I_3 = I \text{ and } I_1 \cap I_2 = \emptyset;\ I_1 \cap I_3 = \emptyset;\ I_2 \cap I_3 = \emptyset.$$

Considering a directed graph $G(V,E)$ where the vertex set $V=\{\sigma_i\}$ and edge set $E=\{\rho_{ij}\}$, there is a following relationship between sequences and exit links.

Relationship 2. The relationship between sequences and exit links has the following two properties:

- in-degree of any σ_i, ($i \in I$), i.e. the number of exit links coming into σ_i, is greater than 0 except for the sequence σ_j, ($j \in I_1$);
- out-degree of any σ_i, ($i \in I$), i.e. the number of exit links coming out of σ_i, is greater than 0 except for the sequences σ_k, ($k \in I_2$).

In other words, a sequence may have zero, one, or many exit links.

Modeling e-negotiation protocols by sequences and components require consideration on the roles of the components. For example, in a sequence, there is a component that should be invoked when the user first enters the sequence. While the components should be accessible from any component in the same sequence in general, there could be some components accessible only after certain conditions are met. We describe these roles by introducing states of a sequence. Components in a sequence are classified into initial, mandatory, optional, and hidden optional states depending on their roles as follows.

Definition 4. Initial, mandatory, optional, and hidden optional states are defined as follows:

- *Initial state* $e(\sigma_i)$: A component which a user is forwarded to when entering a sequence is the initial state of the sequence.
- *Mandatory state* $m(\sigma_i)$: A component which the user has to enter in order to exit to another sequence is the mandatory state of the sequence.
- *Optional* states $O(\sigma_i)$: A set of components accessible to the user is optional states of the sequence.
- *Hidden optional states* $H(\sigma_i)$: A set of components not accessible to the user until some conditions are met is hidden optional states of the sequence.

The needs for initial states and optional states are clear. When there are many components in a sequence, the component to be invoked when a user first enters a sequence should be specified. Also, the reason for having optional states are evident – components that can be visited from any other component from the sequence – from the reason to cluster components into a sequence. Because a user should access the initial state, the initial state must be an optional state.

After trying to model different types of protocols by components and sequences, we found in general there is a component that should be invoked before leaving a sequence. For example, it is desirable to allow moving from the exchange_offer sequence to the agreement sequence only when the read_offer component is being executed, because this will ensure that the user knows which offer he agrees. Mandatory state of a sequence models such a component that should be visited before leaving a sequence. The mandatory state may not be an optional state because sometimes it is desirable not to allow the user to leave the sequence until some conditions are satisfied. Modeling this will be explained later.

We made a design choice of allowing one mandatory state in a sequence, because by doing so the system can consistently apply rules of displaying all exit links at a mandatory state, without considering which links to display on which mandatory state. This rule of the unique mandatory state affects the granularity of the sequences in an e-negotiation protocol. After trials of modeling e-negotiation protocols, we found it leads to a reasonable level of granularity.

A sequence has one initial state and one mandatory state. Because the minimum number of components in a sequence is one, the initial and mandatory states can be the same component. The following statement summarizes the relationships between sequences and states.

Relationship 3. The relationship between sequences and states is given by

$$e(\sigma_i) \in O(\sigma_i) \text{ and } m(\sigma_i) \in O(\sigma_i) \vee H(\sigma_i).$$

Figure 3 visually describes the relationship between components, sequences, exit links, and states.

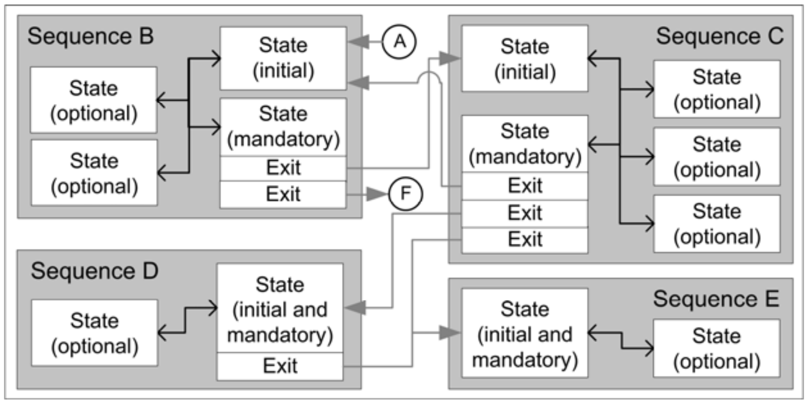


Figure 3. An example of six sequences (A,B,C,D,E,F)

The sequence-state-exitlink model specifies the change of permissible activities through controlling component from the single party's perspective. However, negotiation is an interactive decision making and communication process where the activities to perform depend on the activity of the counterpart as well as the user. For example, when the counterpart sends an offer, the user should be forwarded to read_offer page. Depending on information received from the counter-part, the set of allowed activities modeled by the sequence may change too. For example, when the counterpart agrees to the most recently sent offer, both the user and counterpart should be forwarded to the agreement phase. In order to support these interpersonal activities, e-negotiation protocol should also specify how to adjust component configuration in response to information arrived from the counterpart.

First, we consider sending information to the counter part, which ignites the adjustment of component configuration of the counterpart. We call this intervening.

Definition 5. *Intervening* is to send information to the counterpart and change the status of the counterpart's system.

E-negotiation protocols should consider how to handle intervening or the intervening rules. Well defining how to handle intervening (i.e. intervening rules) is critical in the e-negotiation protocol because intervening is a key activity in negotiations. Intervening rules are the function of information type received. Common information types found in most negotiations are offer, message, agreement, and termination.

Definition 6. *Intervening rules R* change the system through executing, enabling, or invoking components in response to the received information type τ. In other words, intervening rules define the way to change the current activity and/or allowed activities when a specific type of information is received.

We find intervening can be started by executing a component in the sequence (e.g. by sending offer or message) or by exiting to a different sequence linked through an exit link (e.g. by moving to agreement or termination sequence). When the execution of a component c_i or exit link ρ_{ij} starts intervening by sending information type τ, we call c_i or ρ_{ij} is associated with τ. However, not all the components and exit links are associated with an information type because there are also many activities in negotiation that do not affect the status of the counterpart. (e.g. viewing history). It should be also noted that invoking and execution should be discriminated. For example, invoking the send offer component (and thus displaying the send offer page) does not intervene the counterpart because displaying does not sends an offer, while executing the send offer component does cause intervening by sending the offer.

Relationship 4. If the execution of the component or exit link sends a specific type of information to the counterpart, the component or exit link is associated with the information type.

It is important to consider intervening rules based on the type of received information. This received party's perspective on intervening rules makes the ENS more interoperable and even enables the ENS to interact with other ENS's that do not follow the protocol model, because it does not require change in the counterpart's system and by having an interpreter or adapter translating received information into the type of information defined in the intervening rule, the ENS can make proper changes in component configuration in response to the message sent by the counterpart's system.

After trials of modeling different protocols, we identify three types intervening rules described below are most commonly required.

Proposition 1. The three types of intervening rules are most commonly used in e-negotiation protocols:

1. *Activate* hidden optional states in sequence into optional states R_A: $\tau \rightarrow (c_j, \sigma_i)$: when received the information type τ, add c_j to $O(\sigma_i)$ by activating hidden optional states.

2. Update initial state of sequence R_U: $\tau \rightarrow (c_j, \sigma_i)$: when received the information type τ, set $e(\sigma_i)$ as c_j.

3. Forward *a user to a specific sequence* R_F: $\tau \rightarrow (\sigma_i, \sigma_j)$: when received the information type τ, the user is forwarded from σ_i to σ_j.

Activation of hidden optional states allows changing a component from inaccessible from accessible. Update of initial state as well as activation of hidden optional states indicates the mapping between states and components are dynamic and change due to information exchange. Forwarding to a sequence indicates the system may force the user to leave a sequence although the user is not at the mandatory state of the sequence.

For an example of activating a hidden optional state, consider the user receiving an offer from the counterpart for the first time. Allowing the read_offer activity when there is no offer, does not make sense. It is especially so when the accept_offer activity is allowed on the read_offer activity, because this will lead to the situation where the user agrees with the offer that is not even arrived. In order to prevent this non-sense, the read_offer activity should be allowed only if there is at least one offer received. This can be modeled by the intervening rule activating the read_offer component from the hidden optional state into the optional state when the information type offer is received.

For an example of updating the initial state, consider a user received an offer before reaching the sequence modeling offer exchange activities. When he reaches the exchange offer sequence, he should be forwarded to the read_offer page, because there is an offer. On the other hand, it does not make sense to forward him to the read offer page when he did not receive an offer. If there is no offer, displaying the send_offer page makes more sense. This can be modeled by the intervening rule updating the initial state of the exchange offer sequence from the send offer to the read_offer component when the information type offer is received.

For an example of forwarding to a sequence, suppose the counterpart agreed with the last offer sent by a user. The user should be forwarded to the agreement sequence from the exchange offer sequence. This can be modeled by the intervening rule forwarding the user to the agreement sequence from the exchange_offer sequence when the information type agreement is received.

By combining the rule of updating the initial state and forwarding the sequence, it is possible to forward a user to a specific state in a specific sequence. For example, consider the user is in the exchange offer sequence and the counterpart sends an offer. The user should be forwarded to the read offer page if he is not. This can be modeled by the intervening rule updating the initial state of the exchange offer sequence to the read_offer component and another intervening rule forwarding the user to exchange_offer sequence.

Using the constructs and models defined so far, we define e-negotiation protocol as a set of components, sequences, states, exit links, and intervening rules. In other words, e-negotiation protocol is a set of rules controlling components to be used, how they are clustered (i.e. sequence), how those clusters are related (i.e. exit links), how components are invoked (i.e. states), and dynamic change of the description in response to receiving information from the counterpart (i.e. intervening rules).

Definition 7. E-negotiation protocol P is composed of sequences, exit links, and intervening rules:

$$P = \{\sigma, \rho, R\},$$

where $\sigma=\{\sigma_i\}$ and $\rho=\{\rho_{jk}\}$

Since the e-negotiation protocol specifies both intrapersonal and interpersonal activities, different e-protocols may be adopted by the negotiating parties. In order to do so, the protocols adopted by the parties should be compatible. Well designed e-negotiation protocols are compatible with many other e-negotiation protocols including themselves.

In the run time environment, the designed e-negotiation protocol should be instantiated. In a protocol instance, the initial, mandatory, and optional states of sequences change while executing the negotiation according to the intervening rules. In other words, the sequences, states, and exit links should change in run-time in response to the type of information received.

3.3 Patterns and methods

In this section, we illustrate the common patterns of e-negotiation protocols and how to implement these patterns using the constructs and models presented in the previous section. As stated earlier, the proposed protocol model considers the level of details that can be deployed as a web application. The patterns presented in this section are patterns of user's moves among pages and their effects on the counterpart's moves.

As shown in Table 1, we classify some common patterns based on whether user's activity involves interpersonal activities or not. It should be noted that the patterns illustrated in the table are not exhaustive and the described methods are not unique ways of implementing the patterns using the constructs and models.

Table 1 Patterns, examples, and methods of implementation

Name	Description	Example	Implementation
		User-ENS interaction patterns involving only intrapersonal activities	
Free move	A user is allowed to move among a set of activities	When the user exchanges offers, he should be able to do the following five activities: {send offer, read offer, send message, read message, view history} without affecting other accessible activities.	Set the components for these activities c_{SO}, c_{RO}, c_{SM}, c_{RM}, c_{VH} as optional states of exchange offer sequence σ_{EX}, i.e., set $O(\sigma_{EX}) = \{ c_{SO}, c_{RO}, c_{SM}, c_{RM}, c_{VH} \}$.
Sequential move	A user performs activities step by step	In the hybrid conjoint method for eliciting preferences, the user should first rate importance of issues being negotiated, then rate available options in each issue, and finally rate packages (i.e. alternatives)(Kersten and Noronha 1999). In other words, at the page for rating issues, the user should go to the page for rating options next. At the page for rating options, the user should go to the page for rating packages	Model sequences rate_issues(σ_{RI}), rate_options(σ_{RO}), and rate_packages(σ_{RP}) as individual sequences and place exit points ρ_{RI-RO} directed from σ_{RI} to σ_{RO}, and ρ_{RO-RP} from σ_{RO} to σ_{RP} so that activities included in σ_{RI}, σ_{RO}, σ_{RP} can be executed sequentially
An activity allowed only at a specific component	A user can move to a different page only at a specific page	A party can click agree only at the page that displays the most recent offer from the counterpart in order to avoid confusion.	Model two separate sequences exchange_offer(σ_{EX}) and agreement (σ_A) and set up an exit link from σ_{EX} to σ_A (i.e. ρ_{EX-A}). Set the read_offer component c_{RO} as the mandatory state of σ_{EX} (i.e. $m(\sigma_{EX}) = c_{RO}$).
		User-ENS interaction patterns involving interpersonal activities	
Exchanging offers/ messages until reaching an agreement or end	Negotiators interact with each other until reaching an agreement or an end. When one party sends information, the other party is forced to read it. If one of them wants to agree or terminate, they are forwarded to a different stage.	User A and B exchange offers as the following sequence. A sends an offer $\rightarrow$ B reads the offer $\rightarrow$ B sends a counter offer (i.e. reject A's offer) $\rightarrow$ A reads the offer$\rightarrow$ … $\rightarrow$ A or B accepts the received offer $\rightarrow$ Both A and B are forwarded to the read_agreement page	Model two sequences σ_{EX} and σ_A When the send_offer component $c_{SO} \in O(\sigma_{EX})$ is executed the information type offer (i_O) is sent to the counterpart. i_O is associated with the intervening rule updating the initial state of the exchange_offer sequence $e(\sigma_{EX})$ into the read_offer component c_{RO} and forwarding to the exchange_offer sequence. Executing the exit link ρ_{EX-A} from σ_{EX} to σ_A sends the information type agreement i_A. i_A is associated with the intervening rule forwarding to the agreement sequence (σ_A).

Name	Description	Example	Implementation
Addition of an allowed activity	An action is allowed only after the counterpart sends information of a specific type.	The read_offer activity should not be available when there is no offer received. It should be activated when the first offer from the counterpart is arrived.	Model the read offer component c_{RO} as a hidden optional state of σ_{EX} (i.e. $c_{RO} \in H(\sigma_{EX})$). Set an intervening rule activating c_{RO} as an optional state of σ_{EX} when the information type offer (i_O) is received.
Change of the steps to follow	The steps to be followed by a user changes if the counterpart sends information.	After finishing the steps for preparing the negotiation, it is desirable to display the user send_offer page when there is no offer arrived from the counterpart. However, if the counterpart sent an offer, displaying the read_offer page is more desirable.	Set the send_offer component c_{SO} as the initial state of the exchange_offer sequence σ_{EX} (i.e. $e(\sigma_{EX}) = c_{SO}$). Add an intervening rule that updates $e(\sigma_{EX})$ from c_{SO} to c_{RO} if the information type offer (i_O)is received.
Proposal and wait	A negotiator proposes something. If the counter part accepts the proposal both parties move to another step X. If it rejects the proposal they move to another step Y.	A user proposes to add a negotiation issue then wait for reply. The counterpart is forwarded to the read_proposal _for_adding_an_issue page and accepts the proposal. Both parties move to the page showing that the proposal is mutually agreed.	Model sequences for exchange offer(σ_{EX}), proposal(σ_{PR}), and proposal_agreement(σ_{AP}). Set an exit link from σ_{EX} to σ_{PR} (i.e. $\rho_{EX\text{-}PR}$) so that the user can enter the sequence σ_{PR} from σ_{EX}. The initial state of σ_{PR}, $e(\sigma_{PR})$, is the send proposal component c_{SP} and the mandatory state of σ_{PR}, $m(\sigma_{PR})$, is the read_proposal component c_{RP}. Set $H(\sigma_{PR}) = c_{RP}$. Associate c_{SP} with the information type proposal (i_P). Set the intervening rule activating c_{RP} in σ_{PR}, updating $e(\sigma_{PR})$ to c_{RP}, and forwarding to the sequence σ_{PR} when receiving the information type proposal (i_P). Set an exit link between σ_{PR} and σ_{AP} ($\rho_{PR\text{-}AP}$) which is associated with the information type accept_proposal (i_{AP}). Set an intervening rule that forwards from σ_{PR} to σ_{AP} when received i_{AP}.

3.4 Representation of e-negotiation protocols

Software protocol can be viewed as a description of how to integrate components to build a system. Haines et al. (2004) propose component integration through a data infrastructure, such as a database, a blackboard, a message bus, or an object request broker. This data-level component integration allows decoupling of the control of the component execution from communication. Adopting this data-level integration approach, we see the e-negotiation protocol as a software protocol stored in the database, which controls page-level components.

The model for e-negotiation protocol presented in Section 3.2 can be represented by tables and consequently, stored in a relational database. In order to describe the e-negotiation protocol, first we need tables representing the initial setting of sequences, initial, mandatory, optional, and hidden optional states of these sequences, and exit links. Also, to describe the invocation of intervening, for each optional state and exit point, associated the type of information should be specified if the execution of it causes intervening. In addition, three tables for describing three kinds of intervening rules are required: (1) activating hidden optional states, (2) updating initial states, and (3) forwarding to a sequence. All of these intervening rule tables should have the type of received information as the condition for applying the rule. The hidden optional state activation table should specify the state and sequence to activate, the initial state update table should describe the sequence and the component to become its initial state, and the sequence forwarding table should define the origin (i.e. from sequence) and the destination (i.e. to sequence) to forward the user.

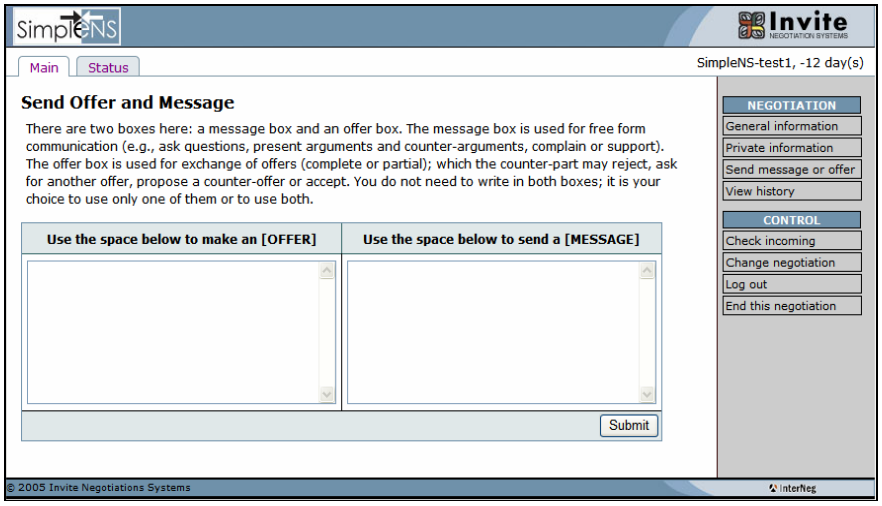
For the operationalization or instantiation of the e-negotiation protocol, the tables for recording the run-time protocol are also necessary because the initial setting of sequences and components changes as negotiation being executed according to the intervening rules. Therefore, for instantiation, the run-time tables for sequences (their states) and exit links are necessary as well as tables storing the current status of the negotiation instance such as the current sequence, user and counter-part information, etc.

4. Examples

In this section, we validate proposed framework for e-negotiation protocol by illustrating how two previously developed ENS's, SimpleNS and Inspire, can be modeled by constructs and models presented in the previous sections.

SimpleNS is a communication and process support oriented ENS which does not offer analytical support to the negotiators. It has been developed for teaching and comparative studies on the use and effectiveness of different ENS's (http://mis.concordia.ca/*SimpleNS*). It provides a virtual negotiation table which allows its users to exchange offers and messages. This system displays the negotiation case and other information required to conduct the negotiation, presents a form in which users write messages and offers, and shows the negotiation history in which all messages and offers are displayed in one table with the time they were made. It has been used in teaching at the University of Ottawa, Concordia University, Vienna University, Austria and National Sun-Yat Sen University, and Taiwan.

The table-based representation of the underlying e-negotiation protocol model emulating the SimpleNS system is presented in Appendix 1. Figure 4 shows the screen shot of the SimpleNS system implemention on the Invite platform, an e-negotiation platform running the protocol defined by the constructs and models presented in this paper. The screen is the page for constructing an offer and/or message. The user is in the exchange_offer sequence, where the read_public_case, read_private_case, construct_offer_and_message, and view_history components are optional states. The main content of the page in Figure 4, is the result of invoking the send_offer_and_message component, and the links on the right column indicate accessible components (i.e. optional states of the current sequence) from the current page, send_offer_and _message.



Send Offer and Message

There are two boxes here: a message box and an offer box. The message box is used for free form communication (e.g., ask questions, present arguments and counter-arguments, complain or support). The offer box is used for exchange of offers (complete or partial); which the counter-part may reject, ask for another offer, propose a counter-offer or accept. You do not need to write in both boxes; it is your choice to use only one of them or to use both.

Use the space below to make an [OFFER]	Use the space below to send a [MESSAGE]

Figure 4 Screen shot of the SimpleNS system implemented on the Invite platform

Inspire is a bilateral ENS developed by the InterNeg research group based on decision and negotiation analysis theory (Kersten and Noronha 1999). Its main purpose is to investigate cross-cultural negotiations and to provide a teaching tool in negotiation courses. Inspire views a negotiation as a process that occurs in a particular context. The system uses a simplified 3-stage process model: pre-negotiation analysis, negotiation, and post-settlement analysis.

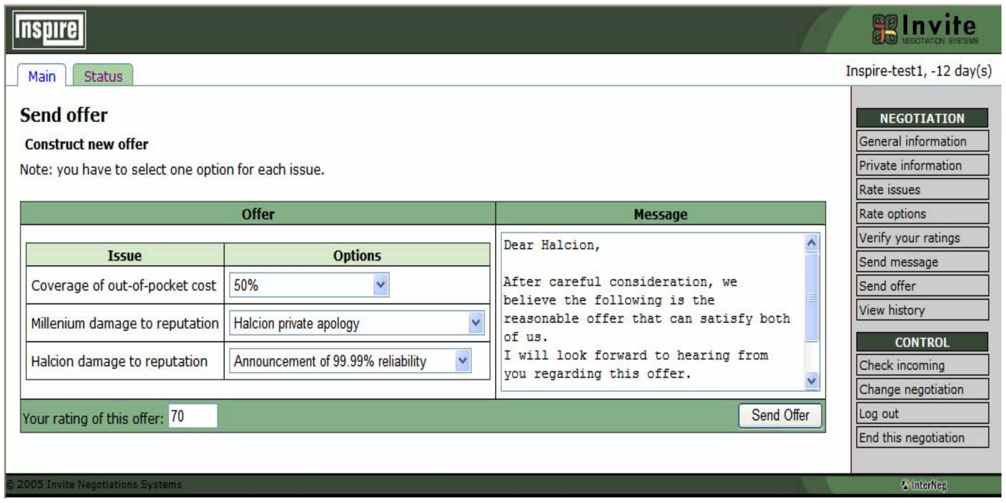
In the pre-negotiation phase, Inspire provides tools for preference elicitation based on the additive utility model. The preference elicitation is performed in three steps. First, the user specifies his preferences over pre-defined issues by distributing 100 points among issues. Once the user assigns preference over the issues, he proceeds to assign scores on the options in each issue. The next step of the pre-negotiation is generation of packages followed by the calculation of ratings for these packages and by the user's verification of these ratings. If the user changes the displayed rating values the least-square procedure propagates these changes to all remaining ratings.

The negotiation phase involves exchange of messages and offers, evaluation of offers, and the review of the progress of the negotiation. In support of the offer exchange activity, the system presents a drop down menu for each issue, so that user can select for each issue only one option. Once an offer has been constructed, the rating is displayed based on the rating function, constructed from user's preferences earlier.

In the post-settlement phase, the system first determines the rating values of the achieved compromise for both users. Then it checks whether the compromise is a Pareto efficient outcome. If not, the system searches for up to five efficient packages and display them to the user so that they can re-negotiate and improve inefficient compromises.

A negotiation in the Inspire system is terminated when (1) an agreement has reached among both parties, (2) one party terminates in any phase during the negotiation, or (3) the period allocated for talk is expired.

The e-negotiation protocol model for the Inspire system deployable on the Invite platform can be found in Appendix 2. Figure 5 shows the screen shot of the Inspire system implemented using the Invite platform. Like in SimpleNS, links on the right column are generated by the optional states defined in the current sequence (i.e. exchange_offer).



Inspire-test1, -12 day(s)

Send offer

Construct new offer

Note: you have to select one option for each issue.

Issue	Options
Coverage of out-of-pocket cost	50%
Millenium damage to reputation	Halcion private apology
Halcion damage to reputation	Announcement of 99.99% reliability

Message:

```
Dear Halcion,

After careful consideration, we
believe the following is the
reasonable offer that can satisfy both
of us.
I will look forward to hearing from
you regarding this offer.
```

Your rating of this offer: 70

Figure 5 Screen shot (the offer construction page) of the Inspire system implemented on the Invite platform

Note that some components (e.g. read_public_case, read_private_case) are reused in both SimpleNS and Inspire protocols. A component used in the e-negotiation protocol can be reused in another protocol as far as it provides necessary functionality and satisfies the intent of both protocols.

5. Conclusion

Context dependency is one of the key issues that hinder development of general purpose ENS and adoption of ENS in practice. In this paper, we propose to adopt the component-oriented software protocol approach to ENS in order to mitigate the context dependency issue. We present a framework providing a detailed methodology of developing ENS following this component-oriented software protocol approach. The core of the framework is the e-negotiation protocol model. The e-negotiation protocol controls interactions between users and ENS by integrating the page-level components, executing them, routing the user to a specific page, and generating links to relevant activities. The protocol also specifies the rules on how to handle information received from the counterpart.

The framework tries to solve the dilemma of context independence of the system and context-aware support to the user by allowing design of e-negotiation protocols that provide certain degree of freedom in activities and by letting users design new or modify existing e-negotiation protocols. In addition, by reusing software components, the framework improves efficiency in ENS development.

In the database arena, a breakthrough improvement in productivity was achieved by separating the conceptual data model, usually represented by the ER-diagram, from physical implementation of the database. In the workflow arena, similar productivity enhancement was achieved by separating the conceptual process model from the physical implementation layer. Our framework is on the similar path, in the sense that the conceptual e-negotiation protocol model provides enough details for implementation but is separated from the implementation of ENS.

The separation of the conceptual model from physical implementation of database and workflow was possible only because it is supported by the lower level infrastructure systems, DBMS and WfMS, which bridge them and execute the conceptual model. In order to support the e-negotiation protocol model, we developed a generic purpose e-negotiation platform called Invite platform which can run e-negotiation protocols conforming to the proposed framework.

The two examples given in this paper are bilateral negotiations. We also found that multi-bilateral negotiations or multi-attribute auctions are also supported by the framework, by simply allowing multiple and selective intervening. Although our work focuses on negotiations, the contribution is not limited to negotiation systems only. It can give a useful model for developing other general web-based applications supporting collaborative and interactive group decision making processes. We will continue to test and refine our framework for various multi-lateral negotiations and collaborative group decision making processes.

Table-based representation of e-negotiation protocols are database and development friendly, but not very user friendly. One of the important future work to be done is to develop tools for visual representation and modeling of e-negotiation protocols. We are studying methods to map visual representation into the table-based representation. GUI tools for visual design of e-negotiation protocols will be implemented based on the method.

We will also perform a comparative study of workflow theory and the proposed framework for e-negotiation protocols. Recently workflow arena has been focusing on expressive power of modeling tools and methodologies for validating defined workflow model. We will approach the e-negotiation protocols from the similar perspective. Desirable properties of the framework and e-negotiation protocols will be identified, and methodologies to check if the defined protocols satisfy the properties will be studied. Text (single spacing)

References

Aalst, W. M. P. v. d. (2005). Workflow patterns, http://is.tm.tue.nl/research/patterns/documentation.htm
Aalst, W. M. P. v. d. and A. H. M. t. Hofstede (2005). "YAWL: Yet Another Workflow Language." Information Systems 30(4): 245-275.
Bassil, S., M. Benyoucef, R. Keller and P. Kropf (2002). Addressing dynamism in e-Negotiations by workflow management systems. Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA2002), Aix-en-Provence, France.
Benyoucef, M. and R. Keller (2001). "Combined negotiation in e-commerce: concepts and architecture." Electronic Commerce Research 1: 277-299.
Berners-Lee, T. and R. Cailliau (1990). World Wide Web: Proposal for a HyperText Project, http://www.w3.org/Proposal.html.
Bichler, M., Kersten, G., Strecker, S. (2003). "Toward a structured design of electronic negotiations." Group Decision and Negotiation 12(4): 311-335.
BPEL4WS (2003). version 1.1 standard specification, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.
Foroughi, A. (1995). "A Survey of the Use computer Support for negotiation." Journal of Applied Business Research.
Gulliver, P. H. (1979). Disputes and Negotiations: A Cross-Cultural Perspective. Orlando, FL, Academic Press.
Haines, G., D. Carney and J. Foreman (2004). Component-Based Software Development / COTS Integration. Software Technology Roadmap. C. M. S. E. Institute, http://www.sei.cmu.edu/str/descriptions/cbsd.html.
Holsapple, C. W., H. Lai and A. B. Whinston (1998). "A Formal Basis for Negotiation Support System." Group Decision and Negotiation 7(3): 203-227.
Jelassi, M. T. and A. Foroughi (1989). "Negotiation Support Systems: An Overview of Design Issues and Existing Software." Decision Support Systems 5(2): 167-181.
Kersten, G. E. (1997). Support for Group Decisions and Negotiations. An Overview. Multicriteria

Analysis. J. Climaco. Heilderberg, Springer Verlag: 332-346.

Kersten, G. E. (2004). "E-negotiation systems: interaction of people and technologies to resolve conflicts." The Magnus Journal of Management 1(3): 71-96.

Kersten, G. E. and H. Lai (2005). "Satisfiability and completeness of protocols for electronic negotiations." European Journal of Operational Research, To appear.

Kersten, G. E. and S. J. Noronha (1999). "WWW-based Negotiation Support: Design, Implementation, and Use." Decision Support Systems 25: 135-154.

Kim, J. B. and A. Segev (2003). A Framework for Dynamic eBusiness Negotiation Processes. IEEE Conference on e-Commerce, Newport Beach, California.

Kim, J. B. and A. Segev (2005). "A Web Services-enabled marketplace architecture for negotiation process management." Decision Support Systems 40(1): 71-87.

Lim, L. and I. Benbasat (1992). "A Theoretical Perspective of Negotiation Support Systems." Journal of Management Information Systems 9(3): 27-44.

March, S. and G. Smith (1995). "Design and Natural Science Research on Information Technology." Decision Support Systems 15: 251 - 266.

Neumann, D., M. Benyoucef, S. Bassil and J. Vachon (2003). "Applying the Montreal taxonomy to State of the Art E-Negotiation Systems." Group Decision and Negotiation 12(4): 287-310.

Stohr, E. and J. L. Zhao (2001). "Workflow Automation: Overview and Research Issues." Information Systems Frontiers 3(3).

Vaishnavi, V. and B. Kuechler (2005). Design Research in Information Systems. ISWorld, http://www.isworld.org/Researchdesign/drisISworld.htm.

Vetschera, R., G. E. Kersten and S. Koszegi (2003). User assessment of Interneg-based negotiation support systems: an exploratory study. Research Paper INR 04/03. I. R. Group, http://interneg.concordia.ca/interneg/research/papers/index.html.

WfMC (2005). XPDL, http://www.wfmc.org/standards/XPDL.htm.

Wikipedia Protocol (object-oriented programming). GNU, http://en.wikipedia.org/wiki/Protocol_%28object-oriented_programming%29. 2005.

Wohed, P., W. M. P. v. d. Aalst, M. Dumas and A. H. M. t. Hofstede (2002). Pattern-based analysis of BPEL4WS. QUT Technical Report FIT-TR-2002-04. Q. U. o. Technology, http://is.tm.tue.nl/research/patterns/download/qut_bpel_rep.pdf.

Appendix 1. SimpleNS protocol

SimpleNS protocol: initial sequences, states, and components

Sequence Name	Initial State	Mandatory State	Optional States (Associated Intervening Information)	Exit Links to (Associated Intervening Information)
Start Negotiation	Read Public Case	Read Public Case	Read Negotiation Details Read Public Case	Read Private Information
Read Private Information	Read Private Case	Read Private Case	Read Negotiation Details Read Public Case	Exchange Offer
Exchange Offer	Construct Offer and Message	Read Offer	Read Public Case Read Private Case History Construct Offer and Message (Offer Message) Read Offer and Message*	Agreement (Agreement)
Agreement	Read Agreement	Read Agreement	History Read Agreement	Terminate Negotiation (Termination)
Terminate Negotiation	Terminate Negotiation	Terminate Negotiation	History	

* Hidden optional state

SimpleNS protocol: intervening rules

Activating hidden optional states

Condition (received information type)	Sequence to apply	Hidden optional state to activate
Offer Message	Exchange offer	Read offer and message

Updating initial states

Condition (received information type)	Sequence to apply	Component to set as the initial state
Offer Message	Exchange offer	Read offer and message

Forwarding to a sequence

Condition (received information type)	Allowed origination (from sequence)	Forwarding destination (to sequence)
Offer Message	Exchange offer	Exchange offer
Agreement	Exchange offer	Agreement
Termination	Any	Termination

Appendix 2. Inspire protocol

Inspire protocol: initial sequences, states, and components

Sequence name	Initial state	Mandatory state	Optional states (associated intervening information)	Exit links to (associated intervening information)
Start Negotiation	Read Public Case	Read Public Case	Read Negotiation Details Read Public Case	Read Private Case
Read Private Information	Read Private Case	Read Private Case	Read Negotiation Details Read Public Case	Rate Issues
Rate Issues	Rate Issue	Rate Issue	Read Public Case Read Private Case Rate Issue	Rate Options
Rate Options	Rate Option	Rate Option	Read Public Case Read Private Case Rate Option	Rate Packages
Rate Packages	Rate Package	Rate Package	Read Public Case Read Private Case Rate Package	Exchange Offer
Exchange Offer	Construct Offer	Read Offer	Read Public Case Read Private Case Write Message (*Message*) History Construct Offer (*Osffer*) Read Offer* Read Message*	Agreement (*Agreement*) Rate Issues Rate Packages
Agreement	Read Agreement	Read Agreement	History Read Agreement	Post-Settlement Terminate Negotiation (*Termination*)
Post-Settlement	[Construct Post-Settlement Offer]	Read Post-Settlement Offer	Read Public Case Read Private Case History View Nego-Dance graph [Construct Post-Settlement Message (***PS Message***)] [Construct Post-Settlement Offer (***PS Offer***)] Read Post-Settlement Offer* Read Post-Settlement Message*	[Post-Settlement Agreement ***(PS Agreement)***]
Post Settlement Agreement	[Read Post-Settlement Agreement]	[Read Post-Settlement Agreement]	Read Post-Settlement Agreement View Nego-Dance graph History	Terminate Negotiation (*Termination*)
Terminate Negotiation	Terminate Negotiation	Terminate Negotiation	History	

* Hidden optional state

Inspire protocol: intervening rules

Activating hidden optional states

Condition (received information type)	Sequence to apply	Hidden optional state to activate
Offer	Exchange offer	Read offer
Message	Exchange offer	Read message
PSOffer	Post settlement	Read PS offer
PSMessage	Post settlement	Read PS message

Updating initial states

Condition (received information type)	Sequence to apply	Component to set as the initial state
Offer	Exchange offer	Read offer
Message	Exchange offer	Read message
PSOffer	Post settlement	Read PS offer
PSMessage	Post settlement	Read PS message

Forwarding to a sequence

Condition (received information type)	Allowed origination (from sequence)	Forwarding destination (to sequence)
Offer	Exchange offer	Exchange offer
Message	Exchange offer	Exchange offer
Agreement	Exchange offer	Agreement
Termination	Any	Termination
PSOffer	Post settlement	Post settlement
PSMessage	Post settlement	Post settlement
PSAgreement	Post settlement	Post settlement agreement