

# The IDEAL Approach to Internet-Based Negotiation for E-Business

J. Hammer, C.B. Huang, Y.H. Huang, C. Pluempitiwiriwaj, M. Lee, H. Li, L. Wang, Y. Liu, and S.Y.W. Su

University of Florida, Gainesville, FL 32611-6125

*jhammer@cise.ufl.edu*

## Abstract

*With the emergence of e-business as the next killer application for the Web, automating bargaining-type negotiations between clients (i.e., buyers and sellers) has become increasingly important. With IDEAL (Internet-based Dealmaker for e-business), we have developed an architecture and framework, including a negotiation protocol, for automated negotiations among multiple IDEAL servers. The main components of IDEAL are a constraint satisfaction processor (CSP) to evaluate a proposal, an Event-Trigger-Rule (ETR) server for managing and triggering the execution of rules which make up the negotiation strategy (rules can be updated at run-time to deal with the dynamic nature of negotiations), and a cost-benefit analysis to help in the selection of alternative strategies. We have implemented a fully functional prototype system of IDEAL to demonstrate automated negotiations among buyers and suppliers participating in a supply chain.*

## 1. Internet-Based Negotiation

Our prototype system, called IDEal (Internet-based Dealmaker for e-commerce), automates bargaining-type negotiations between clients in e-business. The goal is to replace or assist humans with software that can automatically execute the negotiation process to increase the speed and consistency of the negotiation while at the same time reduce human errors. We have developed an Internet-based negotiation infrastructure consisting of interoperating negotiation servers. Furthermore, we have developed a high-level protocol for communication between our servers and for conducting the negotiation process. Negotiation related information is encoded as XML business object documents (BOD) and transmitted via a messaging infrastructure based on TCP/IP. With IDEal, clients can select their own trusted negotiation server to conduct negotiations on their behalf much like lawyers represent their clients in legal matters.

Before the actual negotiation takes place, we assume that potential sellers publish information about the goods and services they provide, which can be browsed by potential buyers using Web browsers and search engines. Through searching and browsing, a buyer can identify

potential sellers with whom the buyer wants to conduct negotiations. In order for buyers and sellers to make use of our automated negotiation service, each must register with an IDEal negotiation server of their choice and provide the necessary negotiation information. The following information is needed to configure each server at build-time: (1) The *negotiation specification*, which describes the requirements and constraints of the item(s) or service(s) to be negotiated. (2) *Negotiation rules*, which describe the negotiation strategies to be followed when constraints are violated; typically, the strategies include rules for relaxing the constraints to reach a mutual agreement. (3) *Preference scores* to evaluate each negotiation position and to select the optimal one under the given circumstances during cost-benefit analysis. This so-called *registration information* is entered into the system through a Web-based registration interface.

During the negotiation at run-time, the negotiation servers exchange negotiation information in the form of proposals and counter-proposals. A proposal contains the constraints from the registration information, specifying acceptable values or ranges of values for the attributes, which describe the item under negotiation (e.g., unit price or desired quantity for a sale). Constraints may involve more than one attribute, which allows the modeling of inter-dependencies. For example, a seller can specify in his registration information that, if during the negotiation, the quantity reaches  $x$  number of units, then the delivery period must be more than  $y$  days after the closing date of the deal. This type of constraint is called an inter-attribute constraint.

Each negotiation engine matches and evaluates the registered constraints of the party it is representing against those specified on the incoming proposal. If no overlap can be found by the constraint evaluation module, the negotiation server applies one or more negotiation rules to determine if its own constraints can be relaxed without compromising its overall goal. If it is possible, a counter-proposal, which contains the relaxed versions of one or more of its own constraints, will be returned to the other party for further negotiation (so far, in our demo we encode the commonly accepted strategy of relaxing only a fraction of the violated constraints to demonstrate cooperation and expect the other party to do the same). Please note, since the negotiation rules encode the

negotiation strategy of the registered clients, they are not revealed to the other side. This exchange of proposal and counter-proposal continues until no constraint violations are detected by either side. At this point, an acceptance message is exchanged and an agreement is reached. Alternatively, if despite existing conflict violations no further modifications can be made to the proposal, the negotiation process terminates unsuccessfully.

A cost-benefit analysis is used in the generation of the counterproposal or the acceptance message by selecting, from possibly many choices of attribute values, the particular ones which are optimal with respect to the given constraints. The preference scores are used as indications of a negotiator's degrees of satisfaction for different values of an attribute. An effective human-computer interaction mechanism is also provided for monitoring the negotiation process and for letting the user influence the negotiation process by directly modifying the counterproposal before it is submitted to the other party.

The structure of a negotiation message is as follows: each proposal contains a negotiation primitive which specifies the intent of the message (accept, reject, withdraw, etc.) followed by a set of constraints which specify the set of values for each attribute of the item under negotiation. We have defined a set of negotiation primitives, including `call-for-proposal`, `propose`, `accept`, `reject`, `terminate`, etc. which is based on the existing negotiation primitives in the Agent Communication Language (ACL) plus new ones to support bargaining and other types of negotiations.

A detailed description of IDEal including our negotiation protocol, the constraint evaluator, and cost-benefit analysis can be found in [2] on our ftp server <ftp.dbcenter.cise.ufl.edu/Pub/publications>.

## 2. Status of Implementation

We have implemented a fully functional prototype system, which is based on the object-oriented, active database technology that has been developed in the database center. Specifically, we have implemented the following essential components:

- A *Web-based graphical interface* which is used to enter (register) the negotiation knowledge including negotiation specifications, strategies, and preference scores to be used by our negotiation engine. The interface also includes a client monitoring system for supervising the negotiation process.
- A *negotiation transaction manager* for managing and coordinating the multi-threaded negotiation sessions, which are modeled as long-running transactions.

- A *repository* for persistent storage of the negotiation information gathered at build-time and the negotiation process information at run time.
- A *cost benefit module* [3] for conducting the cost-benefit analysis and for evaluating negotiation alternatives.
- A *constraint evaluator* for evaluating the consistency of the constraints in the received proposal against the constraints in the registration specification.
- An *event-trigger-rule-based rule engine* [1] for executing strategic rules. At build-time, the rule engine generates the run-time Java executable encoding of the negotiation strategies. At run-time, in the case of constraint violations, the rule engine invokes relevant relaxation rules.

Negotiation information is exchanged among the negotiation servers via XML-formatted proposals. Proposals are sent using a messaging infrastructure. All of the components are implemented in Java using JDK 1.2. To support multi-threaded processing in a distributed environment we use Java RMI. Web-based client and user interfaces and the corresponding server processes are implemented as Java Applets and Servlets respectively.

Although there is considerable substructure (e.g., the registration of negotiation requirements, strategies, and preference scores, the update of rules at run-time), the ICDE demo primarily illustrates the automatic negotiation process among participants in a supply chain. Specifically we demonstrate how IDEAL servers evaluate incoming proposals and generate counter-proposals based on strategic negotiation rules and cost-based analysis of alternatives.

## References

- [1] H. Lam and S. Y. W. Su, "Component Interoperability in a Virtual Enterprise Using Events/Triggers/Rules," in *Proceedings of the OOPSLA '98 Workshop on Objects, Components, and Virtual Enterprise*, Vancouver, BC, Canada, 1998.
- [2] S. Y. W. Su, C. Huang, and J. Hammer, "A Replicable Web-Based Negotiation Server for E-Commerce," in *Proceedings of the Thirty-Third Hawaii International Conference on System Sciences (HICSS-33)*, Hawaii, Hawaii, January 2000.
- [3] S. Y. W. Su, J. Dujmovic, D. S. Batory, S. B. Navathe, and R. Elnicki, "A Cost-Benefit Decision Model: Analysis, Comparison, and Selection of Data Management Systems," *ACM Transactions on Database Systems*, 12:3, pp. 472--520, 1987.