

## Constraint-based Negotiation in a Multi-Agent Information System with Multiple Platform Support<sup>\*</sup>

Dickson K.W. Chiu<sup>1</sup>, S.C. Cheung<sup>2</sup>, Patrick C.K. Hung<sup>3</sup>, Ho-fung Leung<sup>1</sup>

<sup>1</sup>*Department of Computer Science and Engineering, The Chinese University of Hong Kong*

<sup>2</sup>*Department of Computer Science, Hong Kong University of Science and Technology*

<sup>3</sup>*Commonwealth Scientific and Industrial Research Organization (CSIRO), Canberra, Australia*

*email: dicksonchiu@ieee.org, scc@cs.ust.hk, Patrick.Hung@csiro.au, lhf@cse.cuhk.edu.hk*

### Abstract

*Agent technologies have been deployed to model and implement E-commerce activities as multi-agent information systems (MAIS). Agents provide services to one another for mutual gain on behalf of their users. This paper presents an MAIS infrastructure based on belief-desire-intension (BDI) agent architecture, constraint technology, and contemporary Web Services to facilitate negotiation support. Further, the MAIS infrastructure also supports customizable degree of agent delegation for users on different platforms, with or without agent support. We present a constraint-based negotiation protocol extended for a MAIS environment and detail the required adaptations on different platforms from a three-tier implementation architecture aspect.*

### 1. Introduction

The Internet has recently become a global common platform where organizations and individuals communicate among each other to carry out various E-commerce activities, such as stock trading, auctions, and supply-chain management. In this paper, multi-agent information systems (MAIS) are used to conduct E-commerce activities by using agent technologies. Intelligent software agents are programs that act on behalf of their human users and exhibit some aspects of intelligent behavior. An end-user or another agent (known as the delegator) delegates an agent to process or search for information, or to provide a set of value-added services for some business purposes.

Recent advances in hardware and software technologies have created a plethora of mobile devices [22] with a wide range of communication, computing, and storage capabilities. The Internet is quickly evolving towards a

connected society. New mobile applications running on these devices provide users with easy access to remote services regardless of where they are, and start to take advantage of the ubiquity of wireless networking to create new virtual worlds. Moreover, as mobile devices become more powerful, mobile computing will become an increasingly important computation paradigm. New challenging problems arise from the handling of mobility, handsets with reduced screens and varying bandwidth. As such, there are increasing demands for the support of enable users' ubiquitous access. A major challenge of such systems is to match the corresponding requirements into a platform independent solution.

In this paper, we present our extension of a negotiation support system (NSS) into an MAIS infrastructure with multi-platform support, based on belief-desire-intension (BDI) agent architecture, constraint technology, and contemporary Web Services. With a flexible MAIS interface design and external users without agent support can also participate in the NSS. The rest of our paper is organized as follows. Section 2 introduces constraint based negotiation and BDI agent, with related work. Section 3 presents our three-tier implementation architecture of the NSS, with focus on MAIS adaptations required for mobile platforms. Section 4 discusses the applicability of our negotiation protocol. We conclude this paper with our plans for further research in Section 5.

### 2. Constraint Based Negotiation and Agents

Negotiation is a decision process in which two or more parties make individual decisions and interact with each other for mutual gain [29]. Proposals are sent to the other parties, and a new proposal may be generated after receiving a counter proposal. The process continues until an agreement is reached, or even one or more parties quit. As

---

<sup>\*</sup> This work was partially supported by the Hong Kong Research Grant Council with Earmarked Research Grant (HKUST 6170/03E).

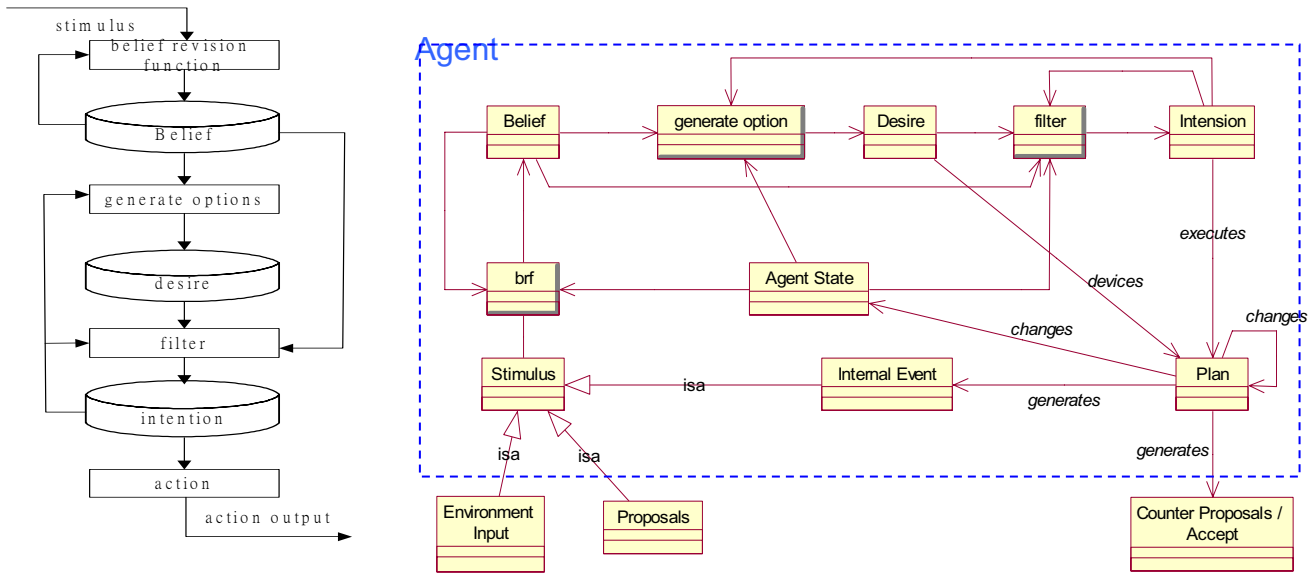


Figure 1. Architecture and class diagram of a Belief-Desire-Intention negotiating agent

many business activities are being automated as electronic transactions, negotiation of service between human becomes the main bottleneck. A major problem of negotiation is its slowness, which is further complicated by issues of culture, ego, and pride [29]. By performing negotiation over the Internet, transaction costs can be greatly reduced. Further with ubiquitous access to NSSs, timely decision can be made for urgent tasks effectively and efficiently.

Intelligent agents are considered autonomous entities with abilities to execute tasks independently. An agent should be proactive and capable of personalization, with a certain level of autonomy. In particular, due to the different limitations on different platforms, users may need different degree of delegation to agents. The prior research usually focuses on the technical issues in a domain-specific application but there is no generic model for depicting the role of agents and other entities in negotiation activities. For example, MIT Media Lab's Kasbah [4] is an online, multi-agent consumer-to-consumer transaction system. Users create autonomous agents to buy and sell goods on their behalf, and also specify parameters to guide and constrain an agent's overall behavior. Similarly, Tete-a-Tete [23] is another application-dependent system, which provides an integrative negotiation approach to retail sales. Further, Teich et al. [28] presented heuristic algorithms for multiple issues electronic markets and auctions that were based on dovetailing buyers' and sellers' interests and preferences. In addition, Lo and Kersten [21] presented an integrated negotiation environment by using software agent technologies for supporting negotiators. However, all of them did not support their model in different platforms.

For logic based negotiation, Bui [1] described various protocols for multi-criteria group decision support in an organization. Bui et al. [2] further proposed a formal language based on first order-logic to support and document argumentation, claims, decision, negotiation, and coordination in network-based organizations. In this context, a constrained-based negotiation could be modeled as a specific case of the ARBAS language.

For the remainder of this paper, we shall use meeting schedule negotiation as an example to illustrate our negotiation protocol and multi-platform adaptation of the implementation. Meeting schedule is a very common collaboration task that is time-consuming and tedious. It involves negotiation between two persons or among several persons, taking into account many factors or constraints. In our daily life, meeting scheduling is often performed by ourselves or by our secretaries via telephone or e-mail. Most of the time, each attendee has some uncertain and incomplete knowledge about the preferences and calendar of the other attendees. Thus a meeting scheduler is a very useful tool for group collaborations. Meeting scheduling is one of the classic problems in artificial intelligence and multi-agent systems. There are some commercial products but they are just calendars with special features, such as availability checkers, meeting reminders [17]. Shitani [27] highlights a negotiation approach among agents for a distributed meeting scheduler based on the multi-attribute utility theory. Van Lamsweerde [20] discusses goal-directed elaboration of requirements for a meeting scheduler, but does not discuss any implementation frameworks. Sandip [26] summarizes an agent-based system for automated distribution meeting scheduler, but is not based on BDI agent architecture. However, these systems cannot support manual interactions in the decision process or any

mobile support issues. Therefore we use this application as an example for our methodology.

### 2.1. BDI Agent Model

As depicted in Figure 1, a BDI architecture is composed of three main data sets: belief, desire, and intention. Information or data are passed from one data set to another through the application of some functions. Once a *stimulus* is sensed as input, the *belief revision function (brf)* converts it to a belief. The *desire* set is updated by generating some options based on the data in belief set. Options in desire set are then filtered to new intentions of the agent, and a corresponding action is then output. The BDI negotiating agent simulates an assistant for decision on behalf of a human user.

Though the agent can receive signals from the environment (e.g., user location), the *stimulus* inputs are mainly incoming requests and responses from the other agents and users. These inputs are usually associated with a set of constraints and/or options (solution) to a proposal. As a result, the *belief* set contains several sets of constraints representing the requirements of a proposal. All solutions or even future options should satisfy these sets of constraints. The function *brf()* is represented by the following function signature:  $\wp(\text{Belief}) \times \text{Percept} \rightarrow \wp(\text{Belief})$ . We give an example to illustrate the task of *brf()* function shown in Figure 2.

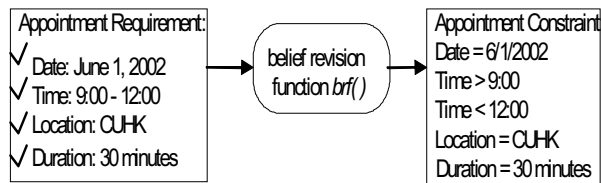


Figure 2. Belief Revision Function

The *brf()* function scans the received stimulus to examine the elements and values of the requirements. In Figure 2, there are 4 elements - *Date*, *Time*, *Location*, and *Duration*. The *brf()* function then converts them into a constraint format. In the above example, the belief generated is:  $(Date=6/1/2002 \text{ and } Time>9:00 \text{ and } Time<12:00 \text{ and } Location='CUHK' \text{ and } Duration=30minutes)$ . The belief implementing the proposals and constraints are later shown in Figure .

When the belief is updated, the next process is to generate some options for the counter proposals. The agent looks up the user's schedule (in a PDA, the schedules are stored with the *calendar* program) to see which time slots are free for a new appointment. The time periods available are converted to constraints again and

then stored in a desire set. In Figure 3, this *option generation function* has the following function signature:  $\wp(\text{Belief}) \times \wp(\text{Intention}) \rightarrow \wp(\text{Desire})$ . We give an example to illustrate the task of the *options generation function* in Figure 3.

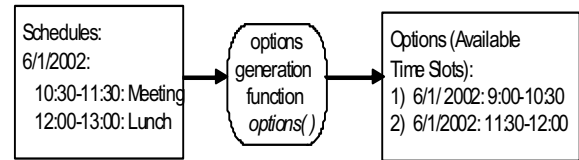


Figure 3: An Options Generation Function Example

The *options generation function* accesses the schedule database to get all appointments within the specified date, say '6/1/2002'. In the above example, two appointments are retrieved. By retrieving the existing appointments, the agent knows when the user is free or not. The *option generation function* then generates a list of valid options (in constraints format). In the above example, two constraints are generated:  $(Date=6/1/2002 \text{ and } Time>9:00 \text{ and } Time<10:30)$  and  $(Date=6/1/2002 \text{ and } Time >11:30 \text{ and } Time<12:00)$ .

Although both the *belief* and *desire* sets contain constraints, their nature is different. Constraints in the *belief* set represent the requirements and preferences of users and agents. These constraints do not reflect any information about their availability. On the other hand, constraints in the *desire* set are created by looking up the users' schedules and reflect their actual availability. The agent can determine solutions from these options in later step.

After generating some options, the agent finds solutions by filtering. The filter function takes the *belief* and *desire* sets as input and returns solutions as output. This approach is called *tree searching* [19], which systematically tests different combinations of values of date (or weekday), time, location, duration, and attendees against the constraints. If a combination of values for these variables satisfies all the constraints, then it is one of the solutions and is output returned as output of the *filter* function, e.g.,  $(Date=6/1/2002, Time=9:00, Location='CUHK')$ . This triggers an update to the intention. The filter function has the following function signature:  $\wp(\text{Belief}) \times \wp(\text{Intention}) \times \wp(\text{Desire}) \rightarrow \wp(\text{Intention})$ . We give an example to illustrate the task of *filter* function shown in Figure 4.

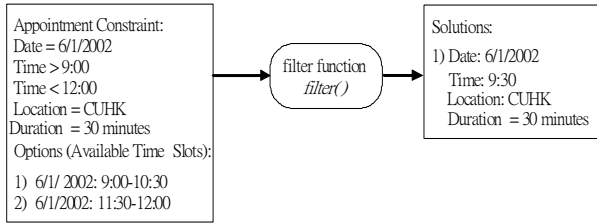


Figure 4: A Filter Function Example



Figure 5. Some sample constraints for meeting scheduling

Constraint satisfaction [19, 30] is one of the major research areas in Artificial Intelligence. Many real life problems, which cannot be easily modeled in specific mathematical forms and solved by conventional Operations Research methods, may be expressed as constraint satisfaction problems. In its basic form, a constraint satisfaction problem consists of a finite number of variables, each ranging over its own finite domain of discrete values, and a finite number of constraints. Each constraint is a relation over a subset of the variables, restricting the combinations of values these variables that they can take. Formally, a *Constraint Satisfaction Problem* is a tuple  $\langle X, D, C \rangle$ , where  $X$  is a finite set of variables,  $D$  a function mapping a variable to its domain, and  $C$  a finite set of constraints. For any variable  $x$  in  $X$ , we require that  $D(x)$  is a finite set of discrete constants. A constraint  $c(y_1, y_2, \dots, y_n)$

in  $C$  is a relation over a finite subset  $\{y_1, y_2, \dots, y_n\}$  of  $n$  variables in  $X$ . A solution to a constraint satisfaction problem is an assignment of values to variables so that all constraints are satisfied.

To illustrate this negotiation problem, suppose that the two main issues to be determined are the time and place for a meeting. Usually, a user has a number of possible options for the time and place of an appointment. Each of these options is usually associated with a degree of preference so that the feasibility of these options can be interrelated. For example, a user Franklin may wish to make an 30-minute appointment at CUHK, the Hong Kong University of Science and Technology (HKUST), or the University of Hong Kong (HKU). He can be there any time from 9:00 to 11:00 if the appointment is either CUHK or HKUST, but only after 14:00 if the appointment is HKU. In addition, the possibility of appointment at HKUST is excluded if it is on a Wednesday. On Sundays and Saturdays, Franklin is not available. Figure 5 shows the constraint satisfaction problem that models Franklin's requirements. As such, a user's requirements can naturally be formulated as a constraint satisfaction problem for negotiation activities.

### 3. System Architecture and Implementation

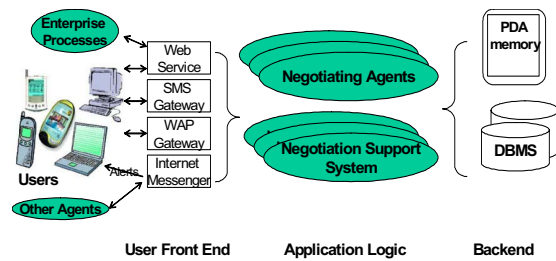


Figure 6: A three-tier MAIS architecture with multi-platform support

Platforms	Enterprise	PDA	WAP	SMS
<b>Views</b>				
<b>User Front End</b>	Web Service interface for programmed interactions	Simplified screen layout Low resolution graphics Panning and zooming	WML translation Highly reduced screen	SMS message presentation
<b>Application Logic</b>	Cross-organizational process interoperation	Simplified process steps and procedures		SMS dialogue presentation
<b>Backend Database</b>	Mutually agreed schema and semantics for interoperation	Omit some fields Summarized information May need to access PDA memory	Mandatory fields only Highly summarized information	Highly summarized and mandatory information as message content

Figure 7: Adaptation for users on various platforms

Internet applications are generally developed with a three-tier architecture comprising user front ends, application logic and backend databases. Figure 6 depicts our MAIS architecture that supports agents, human users, and enterprises. In our architecture, users may either interact manually with other negotiators or delegate an agent to make decision on behalf himself. Thus, users without agent support can still negotiate through an NSS with flexible user interface for multiple platforms, while enterprise business processes can interact with the NSS via Web services.

Figure 7 summarizes the adaptation on each tier required to address the technological limitations of different mobile platforms, such as, PDA, WAP and SMS. Manual negotiators should be supported on different platforms, i.e., web browsers on PDAs, and SMS/WAP on mobile phones. For example, a traveling negotiator may be alerted for immediate decision through mobile devices, such as his/her mobile phone via SMS. Then the negotiator may connect with the mobile phone via or simply send back a reply with SMS. Alternatively, if more information is required, the negotiator may access the NSS or other applications for more information before making the decision with a PDA on a wireless network or a PC in a net cafe. Thus, such extensions to our previous approach [14] by alerting users just with ICQ or emails, can further reduce decision delay when negotiators are away from their home or office. This approach also separates alerts from sessions to improve the flexibility. If a negotiator is not online or does not reply within a pre-defined period, the application server will send the alert again by email. Whatever the alert channel has been, the negotiator need not connect to application server on the same device, or even on the same platform.

### 3.1. Negotiation Protocol Design and Adaptations

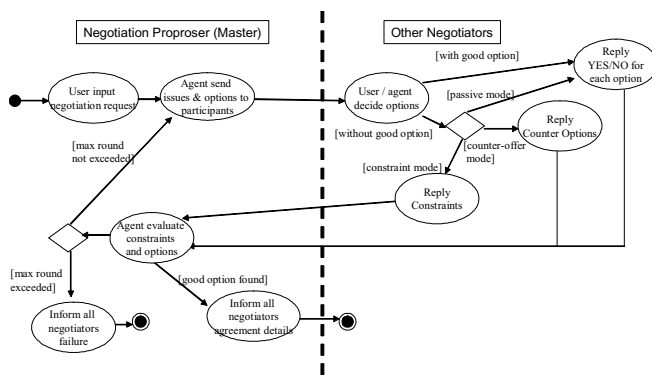


Figure 8: Activity Diagram of agent-based negotiation protocol

Figure 8 depicts the activity diagram of an agent-based negotiation protocol. The user who proposes a negotiation

(or proposer) enters his/her requirements and suggestions (or options) and delegates an agent to initiate the negotiation process. The delegated agent then contacts the other negotiators or their delegated agent. Each of the negotiators or agents then determines if any options are good. If so, the decision can simply be passed back to the proposer. Otherwise, depending on the platform such as *Personal Digital Assistant (PDA)*, *Wireless Application Protocol (WAP)* and *Short Message Services (SMS)* (cf. Figure 9), a negotiator might reply in the following three respond modes: (i) *Passive mode* – the negotiator (or his agent) just reply that all the proposed schedules are bad, without any counter proposals; (ii) *Counter-offer mode* – the negotiator (or his agent) suggests other counter proposals; and (iii) *Constraint mode* – the negotiator (or his agent) gives some of the constraints representing the negotiator’s preference and availability. After gathering the feedback, the proposer’s agent evaluates it for a common feasible solution for all negotiators. If this is successful, all negotiators will be informed of the result. Otherwise, the proposer’s agent will attempt another round of proposals, with the consideration of the options already rejected by the negotiators and their constraints (if any). However, if the maximum round is exceeded, the proposer’s agent will consider the scheduling failed and inform all negotiators.

Negotiators	Enterprise	PDA / Desktop User	WAP User	SMS User
Service Request	Web Service	Browser interface or programmed action	WAP interface	SMS
Automation	Business Process on Server	Agent run on PDA	Agent run on server (and on some new handsets)	
Alerts	SOAP Message	ICQ, email (or SMS if user also accessible)	SMS	
User response mode	Passive, Counter-offer, Constraint	Passive, Counter-offer, Constraint	Passive, Counter-offer	Passive

Figure 9: Different features for negotiators on different platforms

In particular for manual negotiators, we have to impose process restrictions on different platforms. After receiving an alert via ICQ or emails, a PDA negotiator (or other users on a web browser) or WAP negotiator logs on the server to review the details of the service proposal. The negotiator then determines if any options are good. If so, the decision can be passed back to the service provider through the Web or WAP interface respectively. Otherwise, a PDA negotiator might reply in any of these three respond mode described in the previous section. However, the *constraint mode* is too complicated to be supported through a WAP interface and is therefore not provided to WAP users. As for a SMS negotiator, the only practical

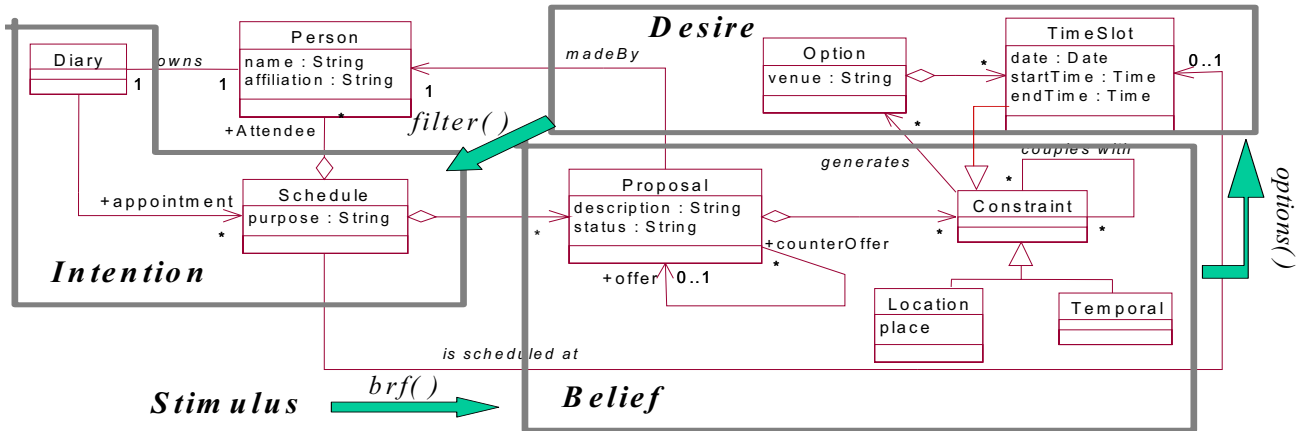


Figure 10. Mapping between the data schema and the BDI architecture in Figure 1

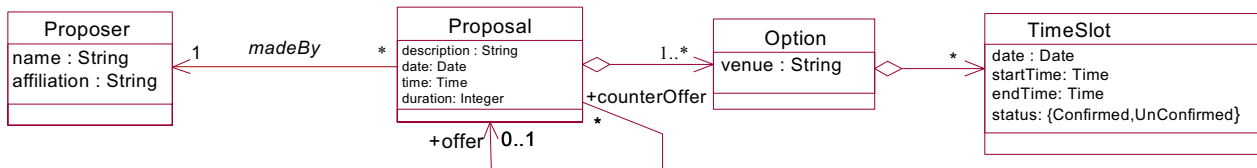


Figure 11. Database view of Display Options Task in UML class diagram

object-oriented design is to carry out requirement analysis. In designing an NSS, we should concentrate on the differences in the decision and operation requirements between various mobile devices and standard desktop PCs. These differences are compared with the standard process to formulate the restrictions. We should identify similar or identical tasks to maximize reuse, and in particular, consider the possibility of customizing them instead of rewriting them.

Usually, a complete detailed decision process is too complicated for a mobile environment. Therefore, typical solutions are simplification of the process, reordering of work steps, delegation of tasks (work steps) to other personal, etc. For example, a user with only SMS support can only make very simple decisions or feedback. As this kind of operating environment is often error prone and may have security problems, we suggest not to allow critical options in these process views. On the other hand, it is often difficult to tell if a negotiator can tolerate a complex process because of the operating environment or even due to the negotiator's mood, which cannot be determined merely with observation facts, such as the mobile platform and physical location. Therefore, it is always a good idea to allow negotiators to choose their desired view options whenever feasible.

### 3.2. Data Schema and View Design

Figure 10 presents visually the schema of a database view in UML Class Diagram for the *Display options* task, based on a meeting schedule negotiation. Note that a class

proposal can be a counter-offer of a previous proposal. Each proposal contains a description and the time when the proposal was made. A proposal consists of constraints concerning the service options. Multiple alternatives can be suggested in an option.

The data view for the *Display options* task is a projection and selection of the entire data set for the NSS. Figure 10 summarizes the implementation of the complete data schema using BDI agent architecture. As shown in the figure, the data view in Figure 11 is a selected subset of data objects as well as a projection of some data object fields. Thick arrows indicate update operations. Here, we have not explicitly elaborated the *plan* in the mapping because it is outside the scope. For instance, the plan to attend a meeting at 9:30am in the Chinese University of Hong Kong (CUHK) may consist of taking a bus for Kowloon at 8:00am and then change for a train to CUHK. In general, a plan of an agent involves proper resource allocation so as to realize its intentions.

The data requirement of a negotiation process is mainly based on the issues and their relation to the data from the database in some form. In particular, we should identify mandatory fields, optional fields and fields that are to be skipped in the view, in order to cope with the simplification required for mobile users. However, additional fields those have to be computed for summarizing information and knowledge may be required. In addition, security requirements should also be considered, e.g., sensitive information may have to be restricted to negotiators within the office or to those of pre-approved locations.

Database views may also be employed to hide less important data field or to show alternate summary columns.

### 3.3. Multi-platform User Interface Design

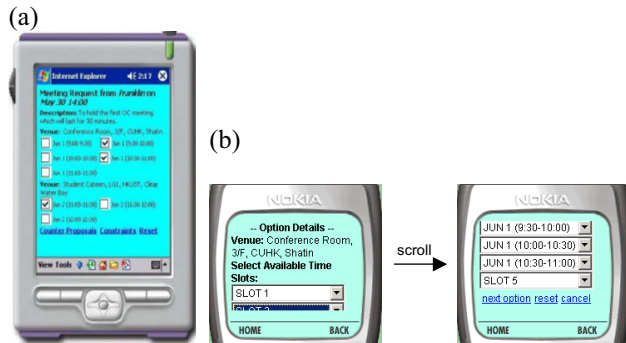


Figure 12. Two types of display option screens for (a) PDA users and (b) WAP users

We have to provide negotiators with appropriate interfaces to interact with other negotiators and agents within the capabilities of front-end devices. This means the user interface for a web user is different from that for a WAP user. Figure 12 presents two possible user interfaces that support the *Display options* task in the negotiation process depicted in for web users and WAP users, respectively. Both Figure 12(a) and (b) correspond to the same XML document object in Figure 14, but are rendered by two different XSL style sheets.

Different user interface can be facilitated by contemporary XML technologies augmented with XSL in elegantly. In most situations, negotiators on PDAs and PCs may also prefer different user interfaces to cater for the difference in screen size. Figure 13 presents a possible implementation framework for multiple user interface using XSL technology [22].

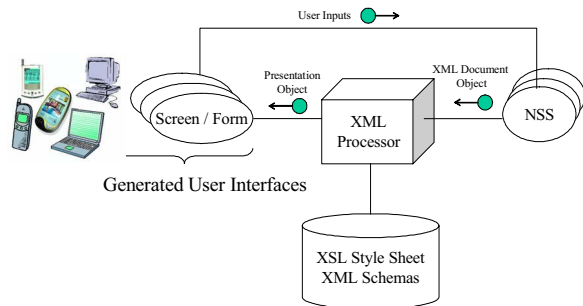


Figure 13. Implementation framework for generating different user interfaces

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Proposal id="mtg-fkn-01">
  <description>Service Options.</description>
  <date>May 30</date>
  <time>14:00</time>
  <Proposer>
    <name>Franklin</name>
    <affiliation>CUHK</affiliation>
  </Proposer>
  <Option>
    <venue>Conference Room, 3/F, CUHK, Shatin</venue>
    <TimeSlot>
      <date>Jun 1</date>
      <startTime>9:00</startTime>
      <endTime>9:30</endTime>
      <status>Unconfirmed</status>
    </TimeSlot>
    <TimeSlot>
      <date>Jun 1</date>
      <startTime>9:30</startTime>
      <endTime>10:00</endTime>
      <status>Unconfirmed</status>
    </TimeSlot>
  </Option>
  ...
</Proposal>
```

Figure 14: An XML Document Object of a Service Options

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="start" title="Option Details">
    <p><b>Venue:</b> Conference Room, 3/F, CUHK, Shatin</p>
    <p><b>Select Available Time Slots:</b></p>
    <p>
      <select name="slot1">
        <option value="NULL">SLOT 1</option>
        <option value="0900">JUN 1 (9:00-9:30)</option>
        <option value="0930">JUN 1 (9:30-10:00)</option>
        <option value="0930">JUN 1 (10:00-10:30)</option>
        <option value="0930">JUN 1 (10:30-11:00)</option>
        <option value="0930">JUN 1 (11:30-12:00)</option>
      </select>
    </p>
    ...
  </card>
</wml>
```

Figure 15: Service Options for Service Appointment in WML

Figure 14 gives an XML document object of a set of alternatives generated by the *Display Option* process. XML document objects are then transformed by an XML processor into presentation objects based on specified XML schemas and XSL style sheets. For example, the presentation objects for a WAP user interface are decks and cards in the Wireless Markup Language (WML). A sample presentation object for the supporting the WAP based process is shown in Figure 15. On the other hand, examples for interfaces definitions for business-to-business (B2B) interactions in Web Services Definition

Language (WSDL) can be found in our earlier work [5, 7, 10, 11].

To accommodate user interfaces in mobile devices, we usually need to remove graphics or reduce the resolution, provide panning and zooming, shorten fields or provide summarized ones instead, break one web page into several screens, etc. For user input, we should consider the difficulties in entering data (especially typing) on mobile devices, and provide menu selections as far as possible. For PDA interface, the main problem is just a smaller screen, some of which may be black-and-white. If the original full-function user interface is too complicated (e.g., too many unnecessary features or high resolution screen layout), another simplified user interface is probably required. Pictures and documents may require to be shown in lower resolution and documents may be outlined and level-structured. Panning and zooming (supported by most browsers) should also help. For negotiators on a WAP interface connecting to the application server via a WAP gateway, the screen is extremely small and it is mandatory to translate the content into WML for display.

## 4. Discussions

In this section, we discuss the applicability of our negotiation protocol and its flexibility in supporting multiple platforms.

### 4.1. Applicability and Validation of Negotiation Protocol

Once all the negotiators' issues are represented as constraints, conventional constraint solving method, such as systematic search, possibly enhanced by constraint propagation [19], can be applied to find the feasible day, time, duration, and place for a service appointment. However, since all these conventional solvers are centralized, it is inappropriate to employ any of them in a mobile setting. This is because employing a centralized solver implies having to require the negotiators to send all or part of their private information to a designated agent, which is supposed to find out a feasible solution to the appointment. Obviously, this is inappropriate as all negotiators are supposed to enjoy privacy protection and autonomy.

We observe that any negotiation protocol involves a trade-off between privacy protection, message exchange costs, and computational efficiency. The protocol described in the previous paragraph, which commonly known as *open-calendar protocol* is computational efficient yet provides no privacy protection. Another problem is that too much unnecessary data is sent. Therefore this approach wastes bandwidth and is not suitable for mobile users or agents. On the other extreme, the most privacy-protected protocol is to require, say, to make a sequence of specific suggestions to the proposer's agent. Each of these suggestions consists of a specific day, a specific time on

that day, and a specific place. The proposer's agent then considers each of them and decides whether or not to accept the suggestion. This protocol, which we call the *passive mode*, is a simple, inefficient protocol (may be causing too many exchanges of short messages) that provides high degree of privacy protection. We can see that there is a spectrum of protocols in between these two extremes, which require the agents to exchange their private information to a certain degree. For example, in the *counter-offer* mode implemented, the proposer's agent can send several options to other negotiators, who are expected to indicate the feasibility of each of these individual options. The negotiators may also counter-propose by replying with a set of his/her own constraints. As such, a balance among privacy protection, message exchange costs, and computational efficiency can be achieved.

Another problem concerning negotiation is the recognition of negotiator's preferences. In our implementation, user preferences are recognized by associating a solution evaluation function defined according to the user's preferences, and enhance the tree search to a branch-and-bound search strategy [19]. In adaptation towards mobility, the agents might need to be aware of the current locations of all the negotiators, such as in this example of meeting negotiation. The information is used to ensure that the negotiators can arrive at the venue on time.

As constraints can be used to express very general problems, including even those involving higher logic [30], our protocol though looks simple is very powerful. It can be applied in different domains for solving different negotiation processes. For formal validation of the protocol for any particular negotiation process, we have also developed a methodology for consistency check based on process algebra and automata theory [10].

### 4.2. Overcoming Platform Limitations

Our origin project aimed at exploring the feasibility of implementing agents on PDA platforms. We implemented them in Microsoft embedded Visual C++. We found that the mobile agents can run a good speed on a PDA over a wireless local area network (LAN), i.e., they can solve the constraints in a few seconds. As such, this distributed architecture does not need a centralized server and therefore can be highly scalable, especially noting that there may be many concurrent negotiations but the participants involving in a single negotiation are usually few.

Next, we discuss the issue of customizing display on different platforms. Although automatic conversion of XML into different output markup languages (such as HTML or WAP) is possible, further summarization and customizing of information to be displayed on different platforms needs knowledge on the information beforehand. Thus, we discover that our previous work on contract-template based negotiation [5, 7, 9, 13] is highly applicable to facilitate customization and reuse. Based on

policies and requirements of negotiation that are to be often repeated, contract templates with parameters are formulated and designed while common issues and options for negotiation are identified based on typical requirements of the target users. These templates and sample tradeoff views are then stored in a repository and available for reuse and user adaptation. In particular, customization for the issues and options to be displayed can be stored with the templates.

To overcome the limitation of the small screen size of mobile devices, decomposing a complex negotiation problem into smaller sets of related issues with our developed methodology [6, 7] further helps. For example, a negotiation process for organization a function can be decomposed into three sets of issues like ({meeting place, time}, {activities, cost}, {cost-sharing}) and then negotiated one set after the other. As such, this negotiation process can be broken down into three rounds of negotiation tasks and executed with the protocol in this paper. As each round has a smaller number of tradeoff issues, the screens are thus simpler and more convenient for mobile users, overcoming both the device and recognition limitations.

## 5. Conclusions

In conclusion, this paper has presented a pragmatic design of adapting a NSS into an MAIS environment, to support various types of users, in particular wireless mobile ones, together with accommodating for enterprise Web Service participation into the negotiation. We have discussed practical implementation details and adaptation required for different platforms based on a three-tier architecture and constraint technologies. Our system supports customizable degree of agent delegation and users without agent support. We have demonstrated the feasibility of our system with a commonly used example of meeting scheduling negotiation. Compared with other researches on this topic, our approach employs an improved environment through standard state-of-the-art technologies, which can adapt to changing requirements of mobile users and devices, with extensive support for reuse. We perceive that our approach is applicable to other group collaboration systems and decision support systems.

We are working on a more detailed methodology of adapting the display of negotiation information and process customization for mobile platform. In particular, we are working on e-Marketplaces negotiation support, ranking of preferences, and alert management for mobile platforms. We are also looking into location depend application, such as mobile workforce management and mobile customer relationship management (CRM) applications [8, 12]. We are also interested in adaptation techniques for mobile computing at program code level.

## 6. References

- [1] T.X. Bui, *Co-oP: a Group Decision Support System for Cooperative Multiple Criteria Group Decision Making*, Springer, LNCS 290, 1987.
- [2] T.X. Bui, F. Bodart, and P.-C. Ma, "ARBAS: A Formal Language to Support Argumentation in Network-Based Organization," *JMIS* 14(3), Winter 1998, pp. 223-240.
- [3] D. Carlson, *Modeling XML Applications with UML*, Addison-Wesley, 2001.
- [4] A. Chavez and P. Maes, "Kasbah: An Agent Marketplace for Buying and Selling Goods," In *Proc 1st Intl. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996, pp. 75-90.
- [5] S.C. Cheung, D.K.W. Chiu, and S. Till, "A Data-driven Methodology to Extending Workflows across Organizations over the Internet," In *Proc HICSS36*, Jan 2003, CDROM, 10 pages.
- [6] S.C. Cheung, P.C.K. Hung, and D.K.W. Chiu, "A Meta-model for e-Contract Template Variable Dependencies Facilitating e-Negotiation," In *Proc. 21st Intl. Conf. on Conceptual Modeling (ER2002)*, Tampere, Finland, Oct 2002, Springer LNCS 2503, pp. 55-64.
- [7] S.C. Cheung, P.C.K. Hung, and D.K.W. Chiu, "On e-Negotiation of Unmatched Logrolling Views," In *Proc. HICSS36*, Jan 2003, CDROM, 10 pages.
- [8] D.K.W. Chiu, W. C. W. Chan, G. K. W. Lam, S. C. Cheung, and F. T. Luk, "An Event Driven Approach to Customer Relationship Management in an e-Brokerage Environment," In *Proc. HICSS36*, Jan 2003, CDROM, 10 pages.
- [9] D.K.W. Chiu, S.C. Cheung, and P.C.K. Hung, "A Contract Template Driven Approach to e-Negotiation Processes," In *Proc. 6th Pacific Asia Conf. on Information Systems*, Tokyo, Japan, Sept 2002, CDROM.
- [10] D.K.W. Chiu, S.C. Cheung, E. Kafeza, and H.F. Leung, "A Three-Tier View Methodology for adapting M-services," *IEEE TSMC*, Part A, 2004, (to appear).
- [11] D.K.W. Chiu, S.C. Cheung, K. Karlapalem, Q. Li, Sven Till, and E. Kafeza, "Workflow View Driven Cross-Organizational Interoperability in a Web Service Environment," *Information Technology and Management*, 2004 (to appear).
- [12] D.K.W. Chiu, S.C. Cheung, and Sven Till, "An Architecture for E-Contract Enforcement in an E-service Environment," In *Proc. HICSS36*, Jan 2003, CDROM, 10 pages.
- [13] D.K.W. Chiu, K. Karlapalem, Q. Li, and E. Kafeza, "Process Views Based E-Contracts in a Cross-Organization E-Service Environment," *Distributed and Parallel Databases*, 12(2-3):193-216, 2002.
- [14] D.K.W. Chiu, Q. Li, and K. Karlapalem, "Web Interface-Driven Cooperative Exception Handling in ADOME Process Management System," *Information Systems*, 26(2):93-120, 2001.
- [15] V. Chopra. *Professional XML Web Services*, Wrox Press, 2001.
- [16] R. A. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, Addison-Wesley, 3rd edition, 2000.
- [17] L. Garrido, R. Brena, and K. Sycara, "Cognitive Modeling and Group Adaptation in Intelligent Multi-Agent Meeting Scheduling," In *Proc. of 1st Iberoamerican Workshop on Distributed Artificial Intelligence and Multi-Agent Systems*, 1996, pp55-72.

- [18] ICQ. <http://www.icq.com>
- [19] V. Kumar, "Algorithms for Constraint-Satisfaction Problems: A Survey," *AI Magazine*, 13(1)32-44, 1998.
- [20] A. van Lamsweerde, R. Darimont, and P. Massonet, "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt," In *Proc. 2nd IEEE Intl. Symposium on Requirements Engineering (RE '95)*, 1995, pp194-203.
- [21] G. Lo and G. K. Kersten, "Negotiation in Electronic Commerce: Integrating Negotiation Support and Software Agent Technologies," In *Proc. 29th Atlantic Schools of Business Conference*, 1999.
- [22] Y.-B. Lin and I. Chlamtac, *Wireless and Mobile Network Architectures*, John Wiley & Sons, 2000.
- [23] MIT, Media Lab. *Tete-a-Tete*. Available at: [ecommerce.media.mit.edu/tete-a-tete](http://commerce.media.mit.edu/tete-a-tete), 2000.
- [24] E. J. Naiburg and R. A. Maksimchuk, *UML for Database Design*. Addison-Wesley, 2001.
- [25] Object Management Group. *Foreword UML specification 1.4*, Sept. 2001.
- [26] S. Sandip, "Developing an Automated Distributed Meeting Scheduler.," *IEEE Expert*, pp. 41-45, 7/8-1997.
- [27] T. Shitani, T. Ito, and K. Sycara, "Multiple Negotiations among Agents for a Distributed Meeting Scheduler," In *Proc. 4th Intl Conf. on MultiAgent Systems*, July, 2000, pp. 435-436.
- [28] J. Teich, H. Wallenius, and J. Wallenius, "Multiple-Issue Auction and Market Algorithms for the World Wide Web," *Decision Support Systems*, vol. 26, no. 1, pp. 49-66, 1999.
- [29] L. Thompson, *The Mind and Heart of the Negotiator*, Prentice-Hall Inc., 1998.
- [30] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.