On the Design and Implementation of a Web-based Negotiation System

Fillia Makedon, Song Ye and Yan Zhao

The Dartmouth Experimental Visualization Laboratory (DEVLAB)
Department of Computer Science, Dartmouth College, 6211 Sudikoff Laboratory
Hanover, NH 03755, USA
Email: {makedon, yesong, yanzhao}@cs.dartmouth.edu
http://scens.cs.dartmouth.edu

Abstract: This paper describes SCENS [1], a Secure Content Exchange Negotiation System, which we are building to enable the exchange or sharing of private (sensitive) multimodal digital data that reside in distributed digital repositories. These data may include raw data, derived data, tools, methods or services. SCENS contains three interconnected layers, each layer implementing distinct functions. The paper focuses on the upper two layers, Layer one and Layer two. Layer one is a traditional web-based negotiation interface for human beings to interact with the system in order to register their data and their conditions. Conditions of negotiation can vary widely and Layer one enables the user to state these conditions. Layer two is an interface for the interaction between or among negotiation agents through web services. We show how these agents cooperate with each other to provide the service for different types of negotiating parties.

1 Introduction

Data sharing of sensitive or highly valuable informational resources requires new models of negotiation to promote communication with built-in incentives, secure authentication, new metadata standards and new metrics of evaluation. This paper describes SCENS [1], a Secure Content Exchange Negotiation System, which we are building to enable the exchange or sharing of private (sensitive) multimodal digital data that reside in distributed digital repositories. These data may include raw data, derived data, tools, methods or services. SCENS contains three interconnected layers, each layer implementing distinct functions. The paper focuses on the upper two layers, Layer one and Layer two. Layer three is described in a separate publication. Layer one is a traditional web-based negotiation interface for human beings to interact with the system while Layer two is an interface for the interaction between or among negotiation agents through web services. We show how these agents cooperate with each other to provide the service for different types of negotiating parties.

Negotiation has been used in electronic commerce transactions [16], where merchandise price/quality are negotiated, negotiation conditions are easy to be represented and values are to be agreed upon. As the conditions of negotiation vary widely, an automated and secure mechanism of reaching agreements on the conditions for sharing data (tools, datasets, methods, services) is needed. SCENS supports data sharing and enhances collaboration of users by supporting condition negotiation and by providing data usage tracking facilities. Our approach is based on the assumption that any type of asset negotiation requires that disparate informational assets be represented via a common, minimalist, one-stop interface that is easy to use and one that provides rewards and incentives for the data owner.

SCENS was originally designed and implemented to support data sharing in neuroscience research [5], where sharing is important in promoting discovery and collaboration. Neuroscience data are not only private but also of high value due to production costs. Traditionally, researchers have resisted sharing primary data in this field. SCENS provides a flexible, user-centered alternative to large, trans-national archiving of published-only results that provide limited access control to the data owner once he has given the data.

2 Related Works

Negotiations can be characterized by a set of common properties [13]: There are two or more parties and a conflict of interest among these parties; the conflict can be resolved by an agreement, which is accepted by all parties; there is a mutual dependency between the parties, one needs the other and vice versa; the parties communicate because they want a better agreement instead of simply accepting what the other side will voluntarily give them.

As a complex, ill-structured, and uncertainty-prone process, a negotiation must contend with half-truths, tricks, and other difficulties. Accordingly its multitude of aspects and dimensions is reflected in different research fields [14], such as game theory [15] and economic theory [16]. In many cases where negotiation might be used to reach agreement, negotiations will not be applied because of potentially high costs and uncertain benefits. Electronic negotiation is a better alternative than the traditional negotiation to lower the costs and increase the efficiency through the assistance of agents and automation of decision tasks.

Several electronic negotiation support systems are currently in use. SmartSettle [2] uses a central server to arrive at agreements without exposing confidential data. WebNS [3] is a prototype web-based Negotiation System, designed to facilitate remote negotiations on the Internet. INSPIRE [4] is a Web-based negotiation support system containing facilities for specification and assessment of preferences. However, most existing negotiation support systems do not have enough flexibility to support negotiation agents and automated negotiation. Furthermore, they are primarily designed to support negotiation in E-commerce, which makes them not suitable to be used in data sharing.

3 Architecture

SCENS has a flexible three-layer structure to provide different types of services for different kinds of negotiation parties. Here we introduce the structure of Layer one and Layer two and show how they interact with each other. Layer three will be briefly introduced here but will not be covered in details. SCENS has the following three layers:

- Layer one is a Web-based Negotiation Support System for end user with browsers.
- Layer two provides negotiation web services for end user with customized agents
- Layer three provides an open and automated negotiation environment to support automated negotiation

Layer one is implementation as a web-based application using Apache web server and Tomcat, which provides JSP and Servlet support, Layer two and Layer three are built on AXIS, which is an implementation of SOAP [7] (Simple Object Access Protocol), the communication protocol for web services. All of these tools come from the Apache Software Foundation.

3.1 Layer One

Layer one is actually a web-based negotiation support system for human beings, which are similar to existing systems mentioned earlier. This layer provides a user friendly interface for two or more negotiating parties to complete their negotiation process and if possible, help them to achieve the agreement.

Layer one contains all the necessary components to support web based negotiation. The important components include:

1) **User Registration (UR)** To become a valid user of SCENS, a user has to get an account through UR.
2) **Dataset Registration (DR)** is used to register the datasets that the owner wants to share through SCENS; it is done through the MetaDL [5] [6] approach which holds the metadata information about the dataset.
3) **Negotiation Conditions Registration (NCR)** is used to register conditions (such as price, usage time, etc) that can be used in negotiation.
4) **Negotiation Strategy Support (NSS)** includes adaptive utility function and negotiation strategy definition to provide negotiation decision making support to users.
5) **Negotiation Activity Visualization (NAV)** allows the user to learn how to best define utility functions or adjust an existing one.

Layer one is the fundamental layer of the whole system; although Layer two and 3 provide additional negotiation services, some services can only be provided through Layer one, such as user registration. In the following subsections, NCR, NSS and NAV will be covered in details.

3.1.1 Negotiation Conditions Registration (NCR)

Negotiation parties involved in a negotiation activity have certain desired goals (e.g. to receive feedback/get payment for their work) and/or concerns (e.g., copyright protection for their work). These goals and/or concerns are expressed as *conditions*. Since conditions differ widely depending on the type of data or users, a flexible way to register conditions is devised.

In SCENS, negotiation conditions are basically divided into two groups, *negotiable* and *non-negotiable* conditions.

- Non-negotiable conditions have strict requirements for the requestor user; for example, the owner may impose a time-limit requirement that the user can not use the data for more than seven days when he registers his condition. If the requestor can not agree with any one of these conditions, the *negotiation will fail*, or enter a "fail" state. Thus, the requestor can actually enter negotiation mode if and only if he agrees with all the non-negotiable conditions at the outset.
- Negotiable conditions are conditions that are to be negotiated through SCENS; for example, the price of a dataset to be shared, the time of usage, etc. There are two types of negotiable conditions, *discrete conditions* and *continuous conditions*. Discrete conditions mean that a condition can have only a limited number of options; for example, the acceptable price is only $100, $200, or $500. Continuous conditions mean that a condition has an unlimited number of options which are variations of some basic negotiation scenarios. For example, if an owner agrees that the payment can range from $100.00 to $500.00, it is possible to define numerous options. In Section 3.1.2, we will see that different types of negotiable conditions will have different forms of *utility function definition*.

Negotiation conditions can be defined either at the owner level, i.e., the owner can apply the same conditions to all the datasets he owns, or at the data level, which means that conditions can vary depending on the data and how these data are to be used and by whom. In the first case, the negotiation is called user-dependent and in the second case is it called data-dependent.

A data owner can use a combination of user-dependent and data-dependent negotiation conditions. He can also modify pre-assigned conditions on a dataset by adding or removing some conditions. When a user first enters SCENS, he will gain access to a repository of commonly used negotiation conditions, even if he does not own any data yet. A user can also customize the condition repository by adding new conditions or modify available conditions. Since the user can define his own negotiation conditions, he has to assure that the defined conditions can be correctly understood by other users including both human negotiators and negotiation agents; otherwise, these conditions can not be successfully used in negotiation.

3.1.2 Negotiation Strategy Support (NSS)

The goal of negotiations is to help negotiators, i.e., owners and system subscribers or users, to achieve agreement. In general, the negotiating parties know their preferences (to be expressed as conditions) well before they begin to conduct negotiation on some dataset or tool. To enable SCENS to facilitate the decision-making regarding the defi-

nition of the conditions, or enable conducting negotiations automatically, it is important for the system to know the users' preferences as accurately as possible in the first place. A RDF-based language is under design to denote and represent preferences; although generally it is not visible to Layer one users, it will be used internally and by Layer two users.

A combination of different conditions associated with a particular offer value is called a *package*. For example, if we use price and usage time as two negotiation conditions on a dataset, then ($100, 3 days) and ($500, 10 days) are two different packages. A human negotiator is able to choose one from these packages, which means he assigns it a higher value than the other. (Different persons may choose different packages, however).

In SCENS, different *utility functions* are used to calculate the value of a package. Utility functions are an important component of forming a *negotiation strategy*. If negotiation is conducted in an interactive manner, the utility function can help negotiators to make decisions, such as rejecting a package with very low value or accepting one with very high value; if negotiation is conducted automatically, for example, when user A wants to negotiate with user B while B is not online, user B's utility function can be used to make decisions directly.

If negotiation conditions C_1, C_2, C_3, … C_n are used in a negotiation, the final goal of negotiation is to reach an agreement (O_1, O_2, O_3, .. O_n), O_i is the offer for conditions C_i. Suppose there are M negotiation parties, N1, N2, N3, N_m, for every Ni, there is a utility function f_i which is used to calculate the value of a possible package. The lowest accept value for N_i is L_i. Thus SCENS will help all the negotiation parties to reach an agreement (O_1, O_2, O_3, ... O_n), which satisfied that

$$f_i(O1, O2, O3, .. On) \geq Li$$

It is not necessary to achieve maximum $\sum_{i=1}^{m}(f_i(O_1, O_2, O_3, .. O_n) - L_i)$, which only maximizes the overall profit for all the negotiation parties. For a particular negotiation party, to achieve maximum profit, which means to maximize $f_i(O_1, O_2, O_3, .. O_n) - L_i$, both the utility function and negotiation strategy has to be carefully designed and used. One party's profit also depends on the other parties' utility functions and negotiation strategies.

Designing a flexible and accurate utility function is essential for Layer one. Depending on discrete or continuous conditions, the utility function definition is different. In what follows, we introduce the utility function definition for discrete conditions. Layer one provides a utility function definition interface, through which the negotiators can define utility functions by assigning a weight to each condition and its option according to his preference. Table 1 and Table 2 below are an example of a utility function.

Condition	Weight
C_1 (Price)	W_1
C_2(Usage time)	W_2

Table. 1. Conditions C1, C2 and their assigned weights

Option	Value
$O_{1,1}$ ($2)	$V_{1,1}$
$O_{1,2}$ ($5)	$V_{1,2}$
$O_{1,3}$($10)	$V_{1,3}$

Table. 2. Option list for C_1 and the assigned value for each option

Using these parameters, the system assigns priority to the conditions and derives an overall value for any package by using the following Utility Function formula:

$$f(O1, O2, O3, .. On,) = \sum_{i=1}^{n} (W_i \times V_{i,j})$$

where O_i is an option for condition C_i, W_i is weight for C_i, and $V_{i,j}$ is the value for O_i in the option list of C_i, (suppose O_i is option j in the list)

Once we have defined this function, given any package, the system can roughly get its value from the user's point of view by applying his function to this package. The users also need to provide the system his acceptable standard (he can accept packages with value above it) and rejection standard (he will reject packages with value below it). These standards are also import part of utility function

Once the system knows the way to estimate the value of a package and the standards, the negotiation begins. In every round, negotiation parties offer a package to each other and the system can automatically first filter this offer by rejection or acceptance, and when the package is fell between acceptable and rejection standard, they alert the user to bargain or bargain automatically.

Automatic bargaining has been widely studied [8, 9, 10] and it remains an open question as no optimal solution has been found. Some researchers argue that it is a *strategy paradox* [8] because if the bargaining algorithm is simple, it is easier deciphered by the other parties, if it is very complex, it will be difficult to formalize it in real life. Therefore we don't provide any automatic bargaining support yet, however, users can use their own preferred algorithm to implement automatic bargaining through Layer two, which will be introduced in section 3.1.4.

Our system employs an adaptive utility function to help the user update his utility function as the time going. Intuitively, when a user first joins the negotiation, somehow he can not correctly evaluate his owning and estimations, and set his initial standards not suitable for achieving his maximum interest. For example, a user is willing to advance his satisfaction standards or update some of his package value assignment if he finds a lot of people feel interested with his data or his offer is accepted quickly by most of opponents. The system can help him to do this adjustment by using adaptive utility function. There are a lot ways to do this. One case is the system tracks the history of the user's negotiations and found there are some users have tried to negotiate for his data in a certain time period, and then the system will advance his acceptable standard and rejection standard by 10%.

3.1.3 Negotiation Activity Visualization (NAV)

Visualization is an important component of Layer one because it allows the user to learn how to best define a new negotiation strategy or change an existing one. The users can learn better negotiation strategy and gain better knowledge about his data-set's value from the negotiation history, the negotiations he has involved up to now. And visualization is the best way to display this record from different point of view. To do this, for each valid user, the system holds detailed information about each negotiation activity he involved in history.

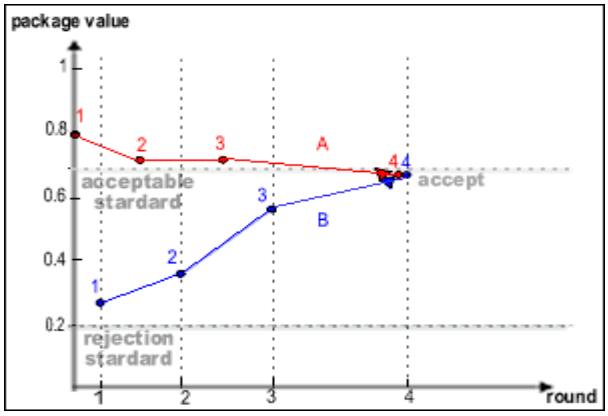


Fig. 1 Single negotiation history between A and B from A's point

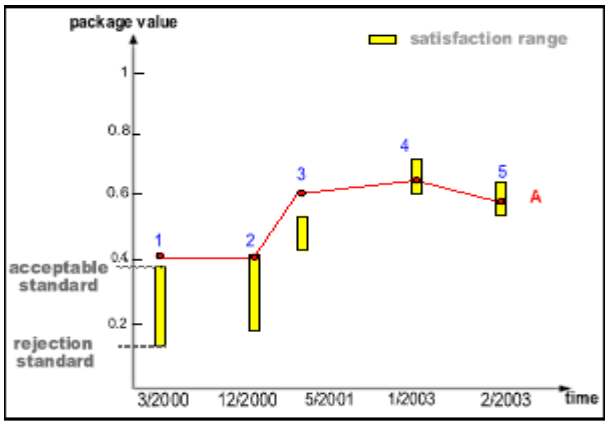
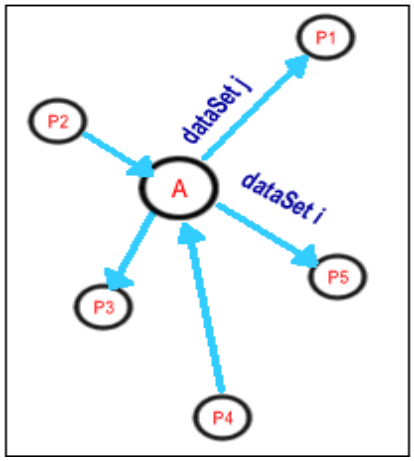


Fig. 2 Multiple negotiation history for a user A

We propose several types of visualization to try to catch every characteristic of negations. First, keep track of single negotiation time for participant separately. We choose the package value in both offers of each round computed by this participant's utility function and both his standards as crucial parameters here, as shown in Figure 1. User A is the data holder and User B wants to negotiate with A to buy his data.

Second, keep track of multiple negotiations that a dataset owner went through on some specific dataset. The movement of his standards and eventually accepted package value in each negotiation is visualized. The user can see whether he has made any improvement and gained more benefit recently. Figure 2 shows such an example.

Third, keep track of communication flow between different users in the history. We want to build a social network for a certain user and for a certain kind of subject (dataset) during negotiation communications. Figure 3.a and 3.b show the communication flow.



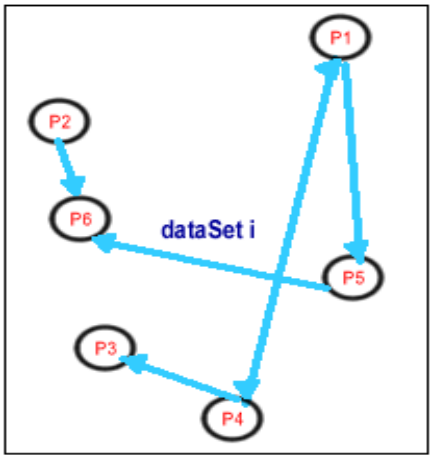


Fig. 3.a Communication flow for user A **Fig. 3.b** Communication flow for dataset i

The figure above shows several visualization approaches that we are implementing. We believe that more and more visualization approaches will be developed as more and more users conduct negotiation through the system.

3.2 Layer two

Although Layer one provides support for negotiation activities under most situations, if one party wants to use a very complex negotiation strategy, it cannot be expressed by adaptive utility functions mentioned in section 3.1.2. For example, a negotiation strategy may be related to current time: the owner may want to accept a proposal from 6pm to 6am, but she may choose to reject the same proposal from 6am to 6pm. To support full customization of negotiation strategies, we design Layer two, which provides negotiation support in the form of web services, and communicates with negotiation parties through SOAP, which is a lightweight protocol for exchange of information in a decentralized, distributed environment.

Layer two supports complete negotiation strategy customization by users. In Layer two, users are allowed to have their own negotiation agents running on client side to implement virtually any possible negotiation strategies. The negotiation agents, which

are treated as web service consumers, conduct negotiation with other negotiation agents or human beings through web services.

The implementation of negotiation agents is slightly different between the owner and requestor. For the requestor, who will initiate a negotiation, the negotiation agent will call the corresponding web service named *InitiateNegotiation* in Layer two, and then send the offers and negotiation decisions to Layer two to conduct the negotiation; for the owner, who will always respond to negotiation initiation request, the negotiation agent will call the web service named *PollNegotiation* in Layer two, and if there are pending negotiation requests, it will send the offers and negotiation decisions to Layer two to continue the negotiation. For both requestors and owners, multiple negotiation activities can be conducted simultaneously.

Figure 4 shows a possible negotiation activity between negotiation agents for the owner and requester. In the figure, square stands for a web service calling, and oval means an operation inside the negotiation agents. For the requestor, after *InitNegotiation*, it will get the default negotiation conditions from the owner; then it will send its offer to the owner and wait for the feedback, which can be a revised offer or a critique. After it gets the reply, it will evaluate the offer or critique and may choose to accept the offer, reject or simply suspend the current negotiation activity (it may need input from human); it can also compose yet another offer and send it to the own and wait for the reply again. For the owner, the negotiation steps are similar, except it will have to check (*PollNegotiation*) regularly if someone has initiated a negotiation request on its datasets. If it finds a pending negotiation request, it will send the default negotiation conditions on the requested dataset and wait for the reply from the requestor.

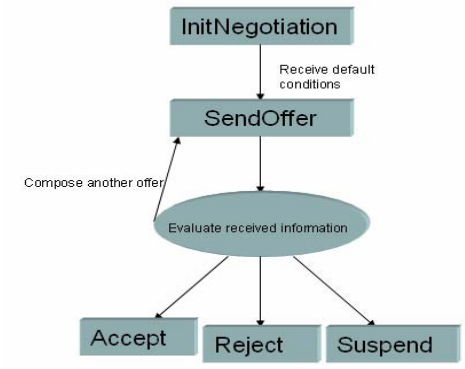


Fig. 4.a Requestor

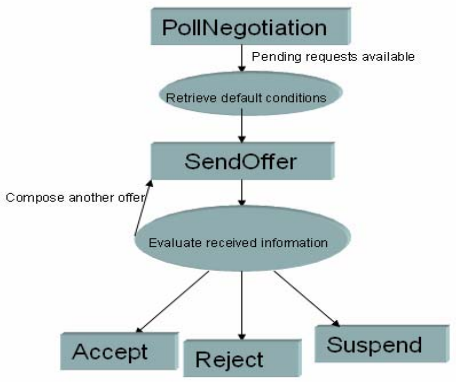
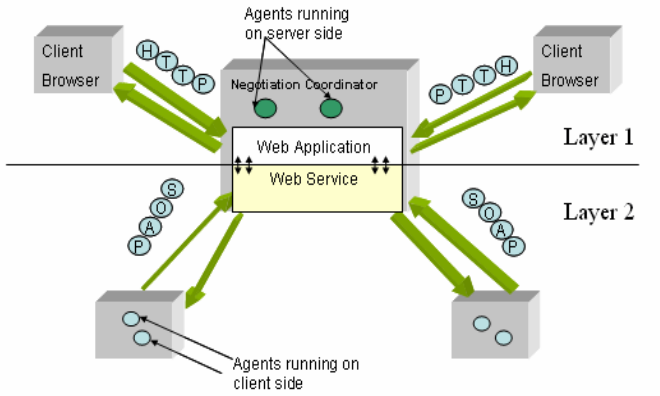


Fig. 4.b Owner

The negotiation parties, which communicates with Layer one and Layer two respectively, can conduct negotiation seamlessly through SCENS. A person conducting negotiation through Layer one will not know whether the other party is another person, or a negotiation agent. Figure 5 shows the interactivity between Layer one and Layer two. A person will use Layer one through a web browser (Internet Explore, Netscape or Opera, etc). He is able to initiate negotiation on the dataset he requests, or

response to dataset requests if he is an owner of some datasets. Layer two is always access by negotiation agents rather than human being directly. For Layer two, negotiation agents are always programs running on the client sides and implementing some negotiation strategies, and they communicate with Layer two through SOAP.



Fig, 5 Layer one provides web-based negotiation services for human beings; Layer two interacts with negotiation agents through web services

It should be noticed that currently Layer two only support actual negotiation process, namely initiating negotiation, bargaining, accepting and rejecting. If the user wants to registration negotiation conditions or visualizes the negotiation history, he has to go to Layer one. We will support registration through Layer two in the future; for visualization, which is implemented primarily for human beings, will be accessed only through Layer one. Since through Layer two, utility functions will be implemented in the negotiation agents on client side, it is not necessary to support utility function definition in Layer two.

3.3 Layer Three

Layer three is designed to provide an open and automated negotiation environment. DAML+OIL [11, 12], a language for creating ontologies and marking up information, is used in Layer three to define a negotiation ontology, which allows agents to acquire knowledge about how to conduct negotiations. This knowledge includes negotiation protocols, negotiation proposals and conditions, etc. Agents communicating with Layer three can be used in any negotiation activities given the proper negotiation ontology. In Layer two, in contrast, the knowledge about negotiation rules is actually hard-coded into the agents.

4 A SCENS Usage Scenario

As an example, consider a negotiation between two users, Hansel and Gretel, researching brain imaging. Assume that Hansel needs a particular type of data that he has learned could be provided by Gretel. Hansel and Gretel use SCENS to negotiate on Hansel's use of a dataset from Gretel.

Suppose they will begin negotiation using price and usage period as conditions. Various other conditions could be involved; for example, if Hansel and Gretel are employees in two different lbs in one department, Gretel might require that Hansel must have his director's approval to conduct this negotiation activity because of the confidentiality of the dataset.

Hansel and Gretel then enter the actual negotiation stage to negotiate on the two conditions they just agreed to work with. Gretel's initial conditions serve as a starting point, and might be, e.g., $1000 in payment and 7 days of use. Gretel can disagree with this and present a counter-offer of conditions, e.g. $700 and 10 days, which can lead to a back-and-forth series of new offers and counter-offers. Eventually, Hansel might accept Gretel's latest offer of a $500 payment and 4 days of use. At this stage, it is also possible that new conditions might be introduced; for example, Gretel might offer free and unlimited usage of the dataset in return for reciprocal use of an equivalent dataset under the same conditions. If the negotiation fails, nothing further will happen between the two parties. If the negotiation succeeds, an electronic contract will be signed by Hansel, Gretel and SCENS. Some privacy issues may have arisen as a result of the negotiation (e.g., Hansel may not want others to know what he is interested in, but revealed this to Gretel).

Negotiation Agents can always be involved in such an example. Suppose Gretel is responsible for all dataset negotiations for her lab, but is not able to monitor negotiations as continuously as she would like. A negotiation agent can act for Gretel in negotiations within certain limits. The agent may be empowered to accept, reject or bargain with requesters. Gretel can define the utility function through the interface provided by Layer one, however, when negotiation strategies are complex and can not be expressed using the utility function, Gretel will choose to negotiate through Layer two. What she needs to do is using a programming language she prefers to implement negotiation strategies through a negotiation agent, which will act as a consumer of web services provided by Layer two.

5 Conclusion and Future Work

In this paper, we introduced SCENS, a negotiation system for mediated data sharing, which provides a three-layer structure to support different types of negotiators and automated negotiation. SCENS not only can be used to support negotiation activities on data sharing, but also serves as a testbed for testing different negotiation strategies (Layer two) and automated negotiation (Layer three). Our next work is primarily in Layer two and three. In Layer two, we are working on improving the performance,

security and reliability of negotiation web services. In Layer three, we are currently building applicable negotiation ontologies for information sharing and will extend these to a general-purpose approach.

References

1. S. Ye, F. Makedon, et al. "SCENS: a System for the Mediated Sharing of Sensitive Data", *Proceedings 3rd Joint Conference on Digital Libraries (JCDL),* Houston, TX (2003)
2. http://www.oneaccordinc.com/ Visited 8/3/2003
3. http://webns.mcmaster.ca/ Visited 8/3/2003
4. G. Kersten, S. Noronha, Supporting International Negotiation with a WWW-based System. Internet Research Report INR05/97, (1997)
5. F. Makedon, J. Ford, L. Shen, T. Steinberg, A. Saykin, H. Wishart, S. Kapadakis: "MetaDL: a Digital Library of Metadata for Sensitive or Complex Research Data", Proceedings European Conference on Digital Libraries (ECDL), Rome, Italy, (2002)
6. F. Makedon, Y. Wang, et al. "A System Framework for the Integration and Analysis of Multi-modal Spatio-temporal Data Streams: a Case Study in MS Lesion Analysis", *Proceedings IEEE 29th Annual Northeast Bioengineering Conference*, Capri, Italy, (2003)
7. http://www.w3c.org/2000/xp/Group/ Visited 8/3/2003
8. C. Beam, A. Segev, M, Bichler, R., Krishnan: "On Negotiations and Deal Making in Electronic Markets", *Information Systems Frontier*, 1(3):241-258 (1999)
9. R. Kowalczyk, A. Bui: "FeNAs: a Fuzzy e-Negotiation Agents System", *Proceedings IEEE/IAFE Conference on Computational Intelligence for Financial Engineering,* (2000) 26-29
10. Y. Wong, D. Zhang, M, Kara-Ali: "Negotiating with Experience", Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, AAAI Press, Technical Report WS-00-04, (2000)
11. Hendler J., McGuinness D.L., The DARPA Agent Markup Language, *IEEE Intelligent Systems*, 16(6):67-73 (2000)
12. http://www.daml.org/2001/03/daml+oil-index Visited 8/3/2003
13. R. Lewicki, D. Saunders, J. Minton. *Essentials of Negotiation*. Irwin McGraw-Hill, Boston MA, (1997)
14. C. Holsapple, H. Lai, A. Whinston. Analysis of Negotiation Support System Research. *Journal of Computer Information Systems*, (1995)
15. P. McAfee, J. McMillan. Game Theory and Competition. *Journal of Marketing Research* 33, (1996)
16. Y. Vakrat, A. Seidmann. Implications of the Bidders' Arrival Process on the Design of Online Auctions. *Proceedings 33rd Hawaii International Conference on System Sciences, (HICCS)* (2000)