Utility Revision in A Java-based Group Decision Support System

Takayuki Ito and Toramatsu Shintani

Department of Intelligence and Computer Science,
Nagoya Institute of Technology,
Gokiso, Showa-ku, Nagoya, 466-8555 Japan.
E-Mail:{itota tora}@ics.nitech.ac.jp

Abstract. We propose a utility revision mechanism for persuasion among Java-based agents in the Group Choice Design Support System (GCDSS). The GCDSS helps a group decision to make a reasonable choice from alternatives. In the GCDSS, agents negotiate with each other on behalf of their users. The GCDSS is written in the Java language with IBM's Aglets Software Development Kit (ASDK). By using the Java language, the GCDSS becomes platform-independent. The ASDK enables us to build agents that are robust with respect to security. In the GCDSS, each user manages a system for an Analytic Hierarchy Process (AHP) and a Java-based agent. Each user subjectively constructs a decision hierarchy and determines the various weights of alternatives by using AHP. Based on the hierarchy and the weights, agents negotiate with each other on behalf of their users. In the negotiation, agents persuade each other in order to reach an agreement. If an agent is persuaded, the agent tries to revise the utility for the state of her decision hierarchy by adjusting the value of pairwise comparisons in the AHP. In this paper, we propose a utility-revision strategy that uses criteria in the user's subjective decision hierarchy for the AHP. The advantage of the utility revision mechanism is that persuasion between agents is facilitated and agents can more effectively come to consensus.

1 Introduction

In recent years, there has been much interest in *intelligent agents* in the field of Artificial Intelligence. In order to act autonomously on behalf of humans in a network environment, intelligent agents should have their own knowledge and utilities. Therefore, in multiagent systems[17], agents' goals can be conflicting. Ideally, agents should be working toward an agreement in order to resolve conflict or encourage cooperation. Reaching an agreement is one of the most important task of agents.

There are several Java-based agent-building packages that allow users to build their own Java-based agents. Because these Java-based agent-building packages are written in Java[1], they can run on any computer with a Java runtime. IBM's Aglets Software Development Kit (ASDK) was one of the first complete packages for building agents, and it has received much attention[10].

The Java-based agents in the ASDK are called Aglets. Aglets are Java objects that can move from one host on the Internet to another. The ASDK provides a security model that enables us to use Aglets in the network. The ASDK's security model enables us to build agents that are robust with respect to security.

There is growing interest in group decision support systems (GDSSs) based on Intelligent agents[18] in the field of Groupware[3] and Computer Supported Cooperative Works(CSCWs)[19]. A GDSS[2] is a computer-based system that facilitates the solution of unstructured problems by a group of decision makers. We have proposed a Group Choice Design Support System (GCDSS) that is a GDSS based on multi-agent negotiation[9]. GCDSS helps a group decision to make a reasonable choice from alternatives. In the GCDSS, we employ the Analytic Hierarchy Process (AHP)[15] in order to quantify the user's subjective judgements. The AHP is a method for making decisions in situations that are hard to analyze quantitatively. The AHP aims to maximize the user's intuition and experience. The AHP involes three stages: (1) The user develops a hierarchy by organizing the problem into its basic components. We call this hierarchy the decision hierarchy. (2) The next step is to establish relative importance weights for each set of elements at each level of hierarchy by using pairwise comparisons based on the decision hierarhcy. (3)Finally, the resulting relative importance weights are composed into composite value that relfect the overall importance of alternative.

In the GCDSS, agents negotiate with each other. In order to reach an effective agreement among agents, we have proposed persuasion mechanisms[8][16]. In a persuasion between agents, an agent who persuades another agent is called a **persuader**, and an agent who is persuaded by a persuader is called a **compromiser**. The persuasion mechanism can be outlined as follows:

1. **Request** The persuader sends a proposal to the compromiser in order to reach an agreement.
2. **Utility revision** The compromiser receives the proposal. If the compromiser is able to accept the proposal, she needs not revise her utility (needs, decisions, or preference). If she is unable to accept the proposal, the compromiser tries to revise her utility in order to accept the proposal.
3. **Reply** If the compromiser is able to accept the proposal as a result of the utility revision, she replies with an agreement message. If not, the compromiser replies with a reject message.

In the persuasion mechanism proposed in the GCDSS[9] agents try to adjust the value of pairwise comparisons in the AHP. In this paper, we propose a utility-revision strategy that uses criteria in the user's subjective decision hierarchy for the AHP.

We present in this paper a utility-revision strategy for persuasion among agents in intelligent group decision support systems. The paper consists of five sections. In section 2, we give an outline for the GCDSS. In section 3, we present a new persuasion strategy based on a subjective decision hierarchy. In section 4, we show a user interface for the persuasion mechanism of the GCDSS and

discuss the results of our current experiments. Related works are discussed in section 5, and in section 6, we make some concluding remarks.

2 The outline of the GCDSS

2.1 Agent-based Group Decision Support in the GCDSS

The GCDSS[9] is a group decision support system based on multi-agent negotiation. The GCDSS is written in the Java language[1] with the Aglets Software Development Kit (ASDK)[11]. The ASDK provides API libraries for building Java-based agents. The Java-based agents are called Aglets. By using the Java language, the GCDSS becomes platform-independent. Each user's decision-making is supported by their own hierarchical decision support module in the GCDSS. The hierarchical decision support modules are written in Java. Agents manage their particular user's hierarchical decision support module and negotiate based on the information which is supplied by the hierarchical decision support module. We build agents in the GCDSS as Aglets. Aglets can interact with the decision support modules by method calls in Java and can interact with other Aglets by message objects provided by the ASDK. The hierarchical decision support module has functions to help generate alternatives, to make judgements for pairwise comparisons, and to construct a hierarchy. The process for supporting group decision-making is described as follows: (1)A host user proposes a topic to be decided. (2)Users make and choose alternatives from a shared alternative database. The alternative database has alternatives for the topic. For example, if *Choosing a Destination for travel* is the proposed topic, then the alternative database has *Los Angeles*, *Las Vegas*, *San Francisco*, *Tokyo*, and so on. (3) Each user constructs a decision hierarchy for AHP using the module. The hierarchy clarifies elements which should be considered in the decision making process. The module is used to quantify the subjective judgements of decision makers by using AHP-based pairwise comparisons. (4) Agents negotiate with each other based on their users' subjective weights and the decision hierarchy. Negotiation among agents is based on the persuasion mechanism. (5) The results of the negotiation are reported to all users.

In step (4) of the process for supporting group decision-making, agents negotiate with each other. A negotiation among agents consists of persuasion between two agents. We discuss persuasion among agents in section 3. A negotiation among agents can be described as follows[9]. First, agents pair off into groups. Next, within each group, one agent who selected randomly persuades the other. If these individual persuasions succeed, the persuading agents assume representation of their respective groups. If the persuasions fail, the agents change places. Next, the representatives begin negotiating with each other. Finally, the agents reach a consensus.

2.2 Constructing a Decision Hierarchy with AHP

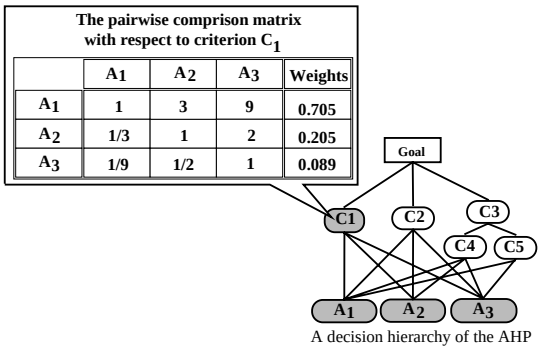

Fig. 1. Constructing a Decision Hierarchy

In the GCDSS, in order to measure the subjective judgements of users, we employ the AHP. Fig. 1 shows a typical hierarchy and a pairwise comparison matrix. In the AHP, users divide the problem into a hierarchy that consists of a goal, criteria (and possibly sub-criteria), and alternatives. For example, in Fig. 1 the goal, the overall objective, is decomposed into criteria C_1,C_2, and C_3. The subcriteria C_4 and C_5 are added under the C_3 criterion. The features of the decision hierarchy are as follows: (1)Each criteria exists at the same level is independent. (2)Each level of the hierarchy is independent. The judgement of the pairwise comparison between factors (in Fig. 1, alternatives A_1,A_2, and A_3) on a certain level is made with respect to the criterion that is a factor (in Fig. 1, criterion C_1) on the upper level. By interpreting a set of judgement values as a matrix (top left of Fig. 1), the weights (i.e., measurement of criteria) of factors are derived from the matrix by using an eigenvector approach. To put it more concretely, the weights of each factor are derived as the eigen-vector for the max eigen-value of the pairwise comparison matrix. As a whole hierarchy, the weights of the alternatives can be recalculated by composing the weights of the criteria.

3 A Utility Revision Mechanism

3.1 Utility Revision based on the Decision Hierarchy

When an agent is persuaded, the agent tries to revise her utility in order to compromise. Agents have the following two strategies for revising their utility: (1)Utility revision based on user's pairwise comparison proposed in[9]. (2)Utility revision based on user's subjective decision hierarchy proposed here. If an agent can not revise her utility for the state of the decision hierarchy by the strategy (1), she can try using strategy (2). Strategy (1) is based on adjusting weights in the pairwise comparison matrix for the AHP. In contrast, strategy (2), proposed here, is based on user's subjective decision hierarchy for the AHP.

In the AHP, criteria will be used to evaluate how well each alternative satisfies the decision objectives. It is hard for the users to create criteria because

AHP's decision hierarchy must satisfy the following conditions: (1)Each criterion existing at the same level must be independent. (2)Each level of the hierarchy must be independent. In order to reduce the work load for creating criteria and to enable more flexible decision-making, we add certainty to the criteria. In order to describe the user's certainty regarding a particular criterion, we propose two types of criteria: (1) a *certain* criterion and (2) an *uncertain* criterion. For example, if a criterion is created as a certain criterion by a user, this means that the criterion is reliable. On the other hand, if the user creates an uncertain criterion, this means that the criterion is unreliable.

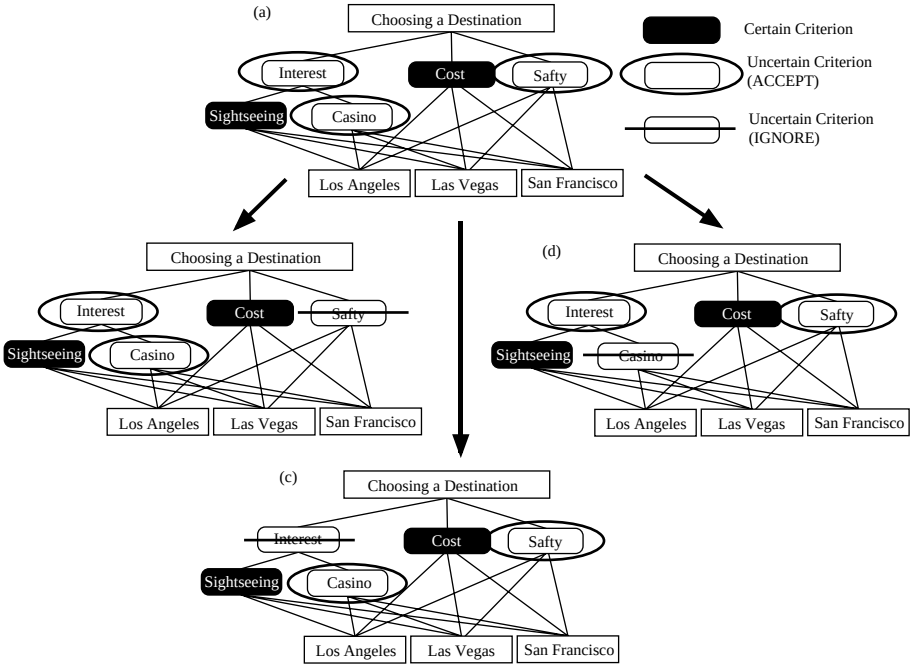

Fig. 2. Creating Candidates

In Strategy (2), a compromiser, an agent who is persuaded, tries to relabel uncertain criteria and to recalculate the weights of alternatives in order to revise her utility. We propose the labels $ACCEPT$ and $IGNORE$ to describe the states of an uncertain criterion. If the label of an uncertain criterion is $ACCEPT$, this means that the agent calculates the entire hierarchy that includes this criterion. If the label of an uncertain criterion is $IGNORE$, this means that the agent calculates the entire hierarchy that does not include this criterion. For example, we show a decision hierarchy at the top of Fig.2. "Choosing a Destination," enclosed

by a square, shows the goal of this decision hierarchy. "Los Angeles," "Las Vegas," and "San Francisco," also enclosed by a squares, show the alternatives for the goal. A black round box indicates a certain criterion, and a round box enclosed by an oval indicates a temporary (or uncertain) criterion labeled $ACCEPT$. A crossed-out, rounded box indicates a temporary (or uncertain) criterion labeled $IGNORE$. Here, the alternatives are omitted in Fig. 2. The initial label of each uncertain criterion is $ACCEPT$.

INPUT : decision hierarchy (DT) and proposal alternative (PA).
OUTPUT: new decision hierarchy (NDT) or empty.
Function $adjusting(DT, PA)$
  Begin
    $NDT := \phi$
    $CANDIDATES := \phi$
    For $i := 1$ **to** Number of Uncertain Criteria in DT
      If the label of $criterion_i$ in DT is labeled $ACCEPT$
      then
        change the label of $criterion_i$ in DT to $IGNORE$
        If DT has enough criteria labeled $ACCEPT$
        then
          $CANDIDATES := CANDIDATES \cup DT$
        end If
      end If
    end for
    $ADT := \phi$
    For $i := 1$ **to** Number of DTs in $CANDIDATES$
      calculate the weights of alternatives in DT_i
      If $max\_weight\_alternative(DT_i) = PA$
      then
        $ADT := ADT \cup DT_i$
      end If
    end for
    $NDT := random\_select\_from(ADT)$
    return NDT
  end.

Fig. 3. An Algorithm for the Utility Revision

In a persuasion, a persuader sends a proposal to a compromiser. The proposal is the most preferable alternative for the persuader in the GCDSS. If the

compromiser's most preferable alternative is not the same as the persuader's, the compromiser tries to adjust her decision hierarchy in order to revise the utility for the state of her decision hierarchy. Fig.3 shows an algorithm for utility revision. The input of the function is a current decision hierarchy and the proposal alternative. The output of the function is a new decision hierarchy if the compromiser can accept the proposal alternative. If the compromiser cannot accept, the output is empty. Empty means that the utility revision has failed.

First, the states of one uncertain criterion labeled $ACCEPT$ changes to $IGNORE$. An agent builds adjusted decision hierarchies as candidates as shown in Fig.2. In Fig.3,the set of candidates is expressed by $CANDIDATES$. If a decision hierarchy does not have enough uncertain criteria labeled $ACCEPT$ for calculating the weights of alternatives, an agent removes the decision hierarchy from the candidates. In calculating the weights of alternatives, there are enough uncertain criteria labeled $ACCEPT$ if there is one or more uncertain criterion labeled $ACCEPT$ at each level of the decision hierarchy. If there are no candidates, the utility revision is a failure. In Fig. 2, the candidates are decision hierarchies (b),(c), and (d).

Second, for each of the candidates, the weights for alternatives are calculated. The criteria labeled $IGNORE$ and the criteria connected to the criterion labeled $IGNORE$ are not used in calculations for the entire hierarchy. For example, in Fig. 2, the criterion labeled $IGNORE$ and the criteria for the level under the level of the criterion labeled $IGNORE$ in decision hierarchy (c) are not used for calculating the weights of the alternatives.

Finally, an agent selects an acceptable decision hierarchy from the candidates as her new decision hierarchy. In the acceptable decision hierarchy, the most preferable alternative is the same as the proposal which is the persuader's most preferable alternative. In Fig.3, the function $max\_weight\_alternative$ returns the most important alternative. If there are several acceptable decision hierarchies, the agent randomly selects one decision hierarchy from the acceptable decision hierarchies. In Fig.3, the set of acceptable decision hierarchies is represented by ADT, and the function $random\_select\_from$ selects a decision hierarchy from this set.

3.2 The Persuasion Process based on Utility Revision

In this section, we show the process of persuasion between two agents. An agent who persuades another agent is called a *persuader*, and an agent who is persuaded by a persuader is called a *compromiser*. The process of persuasion can be described as follows: (Step 1) The persuader sends a proposal to the compromiser. The proposal is the most preferable alternative for the persuader(*request*). (Step 2) The compromiser receives the proposal. If the compromiser is able to accept the proposal, this persuasion succeeds. If she is unable to accept the proposal, the compromiser tries to revise the utility for the state of her decision hierarchy in order to accept the proposal(*utility revision*). If the most preferable alternative is the same as the proposal, the agent is able to accept the proposal,

and the utility revision succeeds. If not, the utility revision has failed. The utility revision process can be described as follows: (Step 2.1) By using strategy (1) based on pairwise comparisons described in section 3.1, the agent tries to revise the utility for the state of her decision hierarchy. If the utility revision is a failure, the agent proceeds to step 2.2. (Step 2.2) The agent tries to revise her utility for the state of the decision hierarchy by using the strategy (2) based on the decision hierarchy described in section 3.1. If the utility revision succeeds, the agent is able to accept the proposal and proceeds to step 3. If the utility revision is a failure, the agent is unable to accept the proposal and proceeds to step 4. (Step 3) The agent asks her user whether the user can accept the new decision hierarchy. This step ensures that the user feels in control of her agent. The feeling of control is an important factor in designing an agent[14]. (Step 4) If the compromiser is able to accept the proposal as a result of the utility revision, she replies with an agreement message. If not, the compromiser replies with a reject message (*reply*). If the persuader accepts an agreement message, this persuasion has succeeded. If the persuader accepts a reject message, the compromiser tries to persuade the persuader (i.e. the agents change position), and the process begins again with steps 1,2, and 3. If the new persuader accepts a reject message from the new compromiser, this persuasion is a failure.

4 Discussions

4.1 An Example

Fig. 4. A Window for Presenting the Decision Hierarchy

Fig.4 shows an example of the GCDSS's window for presenting the labeled decision hierarchy. In Fig.4, the white rectangles for "Interest" and "Safety" represent uncertain criteria and the black rectangles for "Academic," "Sightseeing,"

and "Cost" express the certain criteria. An *ACCEPT* label is shown above each uncertain criterion.

In Fig.5, the agent asks her user whether the user can accept the new decision hierarchy at step (3) of the persuasion process. The two windows at the bottom of Fig.5 are candidates for the new decision hierarchy. The window whose frame is represented by a thick line is the agent's proposal. In Fig.5, the proposal is the bottom right window.

The top window of Fig.5 shows the agent server for managing the agents. The agent server is provided by the Aglets Software Development Kit. The middle window of Fig.5 shows the current status of the agent. If the user can accept the proposal, the user can push the "OK" button in the center window of Fig.5. If not, the user can push the "No" button.

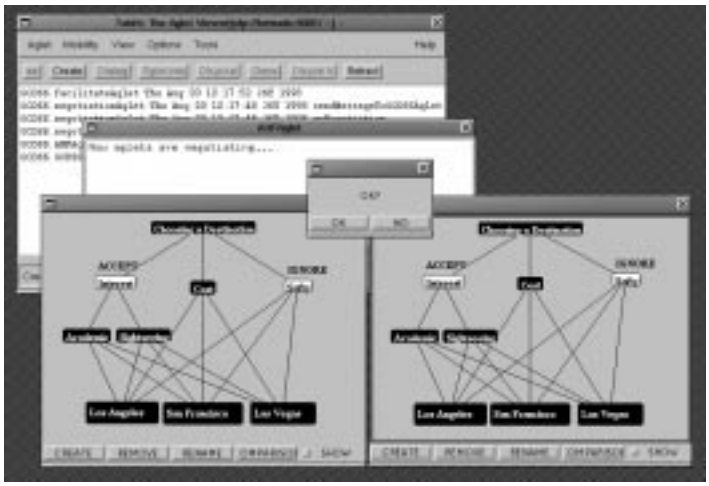

Fig. 5. An Example based on Aglets

4.2 The Advantage of The Utility Revision Mechanism

The advantage of the utility revision mechanism proposed here is that it facilitates successful persuasion among agents by providing an additional persuasion strategy, as described in Section 3.1. Therefore, the utility revision mechanism allows agents to reach a consensus more effectively. However, the easier agents reach a consensus, the greater the number of agents who must compromise. This means that the number of users who must compromise is increased, making the explanation mechanism even more important. In the GCDSS, agents explain to users how to the decision hierarchy is being revised (Fig.5).

The utility revision algorithm (Fig. 3) proposed here tries to change the label of one uncertain criterion in a decision hierarchy. The reason why the algorithm change only one uncertain criterion is described as follows: If the algorithm change the labels of several uncertain criteria in a decision hierarchy, weights of alternatives change dramatically. The users cannot accept these dramatic changes that have been made by their agents in their decision hierarchy. Therefore, our algorithm tries to change only one uncertain criterion at a time in a decision hierarchy.

There has been much discussion of the design of agents [14] [13]. Based on our own research, we believe that the following three points are the most important considerations in the design of agents. First, the users need to feel in control of their agents. Our agents ask the user when the agent adjusts the decision hierarchy. If the user is unable to accept the agent's proposal, the user can reject that proposal. Second, agents should hide complexity while simultaneously revealing the underlying operations. In the AHP,the calculation of the decision hierarchy is complex. Therefore, in the GCDSS the agents hide the calculation of the weights of alternatives while revealing the results of the adjusting. Third, agents should maintain the privacy of the users. In the GCDSS, agents negotiate based on the persuasion mechanism. In the persuasion mechanism, an agent's information regarding the user's preference cannot be accessed by the other agents. The agent who is persuaded tries to revise her utility on her own.

5 Related Works

Multi-agent systems[17] have been investigated very actively. Some interesting recent work regarding systems that support group activities includes Haynes, Sen, Arora, and Nadella's automated meeting scheduling system[6], Garrido and Sycara's distributed meeting scheduling system[5], a multiagent-based office work support system[7], Mediator[4], and so on. In the Hayes, Sen, Arora, and Nadella[6]'s system, agents can schedule meetings within user's preference. In this system, agents cannot adjust user's preference. Garrido and Sycara[5] have modeled negotiation as a constraint-relaxation process for distributed meeting scheduling. In constraint-relaxation process, agents can adjust user's preferences. Since the constraint-relaxation proceeds automatically, the agents cannot explain the reason why the user's preference is adjusted. In our system, agents can adjust user's preference in a persuasion process. Based on certainty for criterion, agents can explain the reason for adjusting the user's preference. There exist agents for personal work and group work, and they are mutually connected in a two-layered agent network. Since the goal of the system[7] is to facilitate office work, agents perform tasks by communication based on requests and answers. The Mediator[4] is a groupware coordination tool that is designed to share knowledge structures across local and wide area networks by using concept maps. These works[7][4] provide functions for sharing information and knowledge among the users. In the GCDSS, agents negotiate on behalf of users. In the negotiation, each agent tries to revise her utility in order to compromise. Specifically, our system provides sup-

port for negotiations among users. Agents that support personal activity have been developed. Maes[12] has focused on learning the single user's preference by agent, implementing an agent for electronic mail handling, and so on. She does not, however, focus on agent negotiation. In the GCDSS, the user's preference is represented by a decision hierarchy for the AHP. Agents can use the decision hierarchy as a representation of the user's preference.

6 Conclusions

We propose a utility revision mechanism for persuasion among Java-based agents in the Group Choice Design Support System (GCDSS). The GCDSS is an intelligent group decision support system based on negotiation among Java-based agents. The GCDSS is written in Java. Therefore, the GCDSS can run on any computer with a Java runtime. We employ IBM's ASDK in order to build Java-based agents. In the ASDK, Java-based agents are called Aglets. In the GCDSS, Aglets can interact with other Aglets with message objects provided by the ASDK and can interact with the hierarchical decision support module via method call in Java. The ASDK provides a security model for building Java-based agents in the network. The security model enables us to build agents that are robust with respect to security. If an agent is persuaded during a negotiation, the agent tries to recalculate the decision hierarchy in order to revise the utility for the state of her decision hierarchy. Agents have two strategies for utility revision. Strategy (1) is based on adjusting weights in the pairwise comparison matrix for the AHP. In contrast, strategy (2), proposed in this paper, is based on a user's subjective decision hierarchy in the AHP. In order to reduce the user's work load for creating criteria, we introduce *certain* and *uncertain* criteria. Users can create *uncertain* criteria if the criteria are not reliable. Furthermore, we introduce the labels *ACCEPT* and *IGNORE* to describe states of a *uncertain* criterion. By using these labels, agents adjust the user's decision hierarchy in order to revise their utility. In this paper, we discuss the advantage of the proposed utility revision mechanism and the design of agents. The advantage of the utility revision mechanism is that persuasion between agents is facilitated and agents can more effectively come to consensus. Our design of the agents satisfies the following three points for the design of agents which are also included in the discussion[14][13]: First, the users can have the feeling that they are in control of their agents. Second, our agents can hide complex work such as decision-hierarchy calculations but reveal underlying operations such as the adjusting process by showing the results of the adjusted decision hierarchy. Third, our agent can maintain the user's privacy regarding individual preferences.

References

1. Arnold, K. and Gosling, J.: The Java Programming language. Addison-Wesley, 1996.

2. Desanctis, G. and Gallupe, R.B.: A foundation for the study of group decision support systems. *Management Science*, Vol.33, No.5, pp.589–609, 1987.
3. Ellis, C.A., Gibbs, S.J., and Rein, G.L. : Groupware : Some Issues and Experiences. *Communications of the ACM*, Vol.34, No.1, pp. 38–58, 1991.
4. Gaines, B.R., and Shaw, M.L.G.: Using Knowledge Acquisition and Representation Tools to Support Scientific Communities. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pp. 707–714, 1994.
5. Garrido, L., and Sycara, K.: Multi-agent meeting scheduling: Preliminary experiment results. In *Proceedings of Second International Conference on Multi-Agent Systems(ICMAS-96)*, pp.95–102., 1996.
6. Haynes, T., Sen, S., Arora, N., and Nadella, R.:An Automated Meeting Scheduling System that Utilizes User Preferences, in *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pp. 308–315, 1997.
7. Ishiguro, Y., Tarumi, H., Asakura, T., Kida, K., Kusui, D. and Yoshifu, K.: An Agent Architecture for Personal and Group Work Support, *Proceedings of Second International Conference on Multi-Agent Systems(ICMAS-96)*, pp. 134–135, 1996.
8. Ito, T., and Shintani, T. : An Agenda-scheduling System Based on Persuasion Among Agents, In *Proceedings of IPSJ International Symposium on Information Systems and Technologies for Network Society*, pp.287-294, World Scientific, 1997.
9. Ito, T., and Shintani, T. :Persuasion among Agents : An Approach to Implementing a Group Decision Support System Based on Multi-Agent Negotiation, In *Proceedings of the Fifteenth International Joint Conf. on Artificial Intelligence (IJCAI-97)*, Morgan Kaufmann, pp. 592–597, 1997.
10. Kiniry, J. and Zimmerman, D. :A Hands-on Look at Java Mobile agents, IEEE Internet Computing, pp.21-30, Vol.1, No.4, July/August, 1997.
11. Lange, D. and Chang, D., :IBM Aglets Workbench, Programming Mobile Agents in Java, http: //www.trl.ibm.co.jp /aglets /whitepaper.htm , 1996.
12. Maes, P.: Agents that Reduce Work and Information Overload., *Communications of the ACM*, Vol. 37, No. 7, pp. 31–40 ,1994.
13. Malone, T. W., Lai, K., and Grant K. R.,: Agents for Information Sharing and Coordination: A History and Some Reflections, *Software Agents* (Bradshaw, J. M.(ed.)), AAAI Press/The MIT Press, chapter 7, pp. 109–143, 1997.
14. Norman, D.A.: How Might People Interact with Agents, *Software Agents* (Bradshaw, J. M.(ed.)), AAAI Press/The MIT Press, chapter 2, pp. 49–55, 1997.
15. Saaty, T. L. : The Analytic Hierarchy Process, McGraw Hill, 1980.
16. Shintani, T., and Ito, T., : An Architecture for Multi-Agent Negotiation Using Private Preferences in a Meeting Scheduler, In *Proceedings of the 5th Pacific Rim International Conferences on Artificial Intelligence (PRICAI'98)*, 1998 (to appear).
17. Sycara, Katia P. : Multiagent Systems, In *AI magazine*, AAAI, Vol.19, No.2, 1998.
18. Turban, E., and Aronson, J.E., : Decision Support Systems and Intelligent Systems, fifth edition, Prentice-Hall International, Inc., 1998.
19. Wilson, P.: Introducing cscw - what it is and why we need it. In Stephen A.R. Schrivener, editor, *Computer-Supported Cooperative Work*, Applied Information Technology series, chapter 1, pp. 1–18. UNICOM, 1994.

This article was processed using the LaTeX macro package with LLNCS style