Adding Flexibility to a Room Booking System Using Argumentation-Inspired Negotiations as Mediated by Mobile Agents

Cheah Wai Shiang, Seng Wai Loke, Shonali Krishnaswamy, Sea Ling
School of Computer Science and Software Engineering, Monash University, Australia
wsche16@student.monash.edu, shonali.krishnaswamy@csse.monash.edu.au,
swloke@csse.monash.edu.au, chris.ling@csse.monash.edu.au

Abstract

Ideas from argumentation-based negotiation can be incorporated into human-agent interaction (HAI) to increase the flexibility of a system and allow greater user control. However, this can increase the complexity of system design and implementation. This paper discusses the use of mobile agents in mediating the interaction between people, and between people and a system, including resolving conflicts through negotiation in a flexible and user controllable manner. This paper also discusses the issues and design principles for argumentation-based negotiation in HAI in the context of a Flexible Smart Room Booking System.

1. Introduction

Agent technology has brought a new dimension to the global computing environment. It is a piece of software that is autonomous, proactive, responsive, adaptive and flexible [3]. This paper discusses the use of mobile agents in mediating interaction between people including resolving conflicts through negotiation in a flexible and user controllable manner. The mobile agent paradigm is used due to the ability to work with different protocols, execute asynchronously, adapt dynamically, operate in heterogeneous environments, and they are robust and fault tolerant [3]. Argumentation-based negotiation [1,6,8] and human agent interaction [2,4,5] have been applied to achieve our objective above. It has indicated that both techniques will enhance the flexibility and user controllability of systems. An example system, the Flexible Room Booking System, has been implemented.

The Flexible Room Booking System is a mobile agent based booking system that handles user tasks such as hall booking, service enquiring, booking confirmations, allocation and change of negotiation strategies and leaving messages for agents. In this case, the agent will respond to the instruction that has been assigned and pro-actively organize the strategy to handle conflicts that occur. Furthermore, the agent can go back to its owner for more information.

The rest of this paper is organized as follows. Section 2 describes the design of the flexible booking system. This includes a discussion on the interaction protocol towards argumentative conversation. Section 3 describes the implementation of the system, using Grasshopper.[1] The paper concludes in section 4.

2. Interaction Design

The main components of agent negotiation are interaction protocol and negotiation strategy. Agent negotiation consists of activities like finding the information, matching the preferences, reviewing the negotiation strategy, organizing the negotiation strategy and exchanging messages until reaching the final decision. The argumentation-based negotiation approach consists of issues like locutions design, interaction protocol design, intelligent mechanisms design, storage handling design and argument management. Agents in this case are aware of the information surrounding them and attempt to analyze, influence and understand their opponents.

Arguments can be categorized in terms of expression and exploration. Expression is the activities for agents to communicate their interests, needs and preferences. They would like to notify the other party about their needs, role(s), level of interest, make claims and make promises to prevent any influences from other agents. Meanwhile, exploration is the activities for agents to challenge other agents' interests, needs and preferences, which can be done by asking different types of questions.

By combining the work from Sierra *et al.* [7], Rahwan *et al.* [6] and Toda. *et al.* [8], we came up with a list of locutions towards argumentative conversations. They are propose, reward, refuse, accept, agree, inform, request, accept-proposal, reject-proposal, notify or notification (expression), promise (expression), clarify (expression) and ask (exploration). New keywords are needed to simulate the modes of expression and exploration. Notify, promise, clarify and ask are used for argumentation and

[1] http://www.grasshopper.de

Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04)
0-7695-2101-0/04 $ 20.00 IEEE

form the basis for different types of *communicative act specializations* such as *notify_claim, promise_reward, clarify_like, ask_more* and so on. Meanwhile, the interaction protocols for argumentative conversations are required to determine agents' conversations and ease of implementation. We utilize the FIPA communicative acts in the design of our interaction protocol for argumentative conversations.

The focus of human agent interaction (HAI) is to determine the agents' autonomy and task delegation. The design process involves several issues:

- Who will decide the agent's autonomy – agent or user?
- When should the agent be fully autonomous or semi autonomous?
- How to adjust the agent's autonomy?

In our design, the user will decide the level of agents' autonomy. The user will decide under what conditions or in which situations the agents can become fully autonomous or semi autonomous. Furthermore, the user is fully responsible for assigning responsibilities to the agents. The user can allocate the level of user disturbance or interruption to the agents. The level of user interruption will determine the degree of the agent's autonomy –e.g., agents will always refer to the user for additional information and important decisions or not disturb the user at all. The level of interruption also influenced by information from the agents. During the negotiation process, the user can adjust the level of interruption by leaving notes to agents. The agents will direct the incoming message to the user or refer to the notes given by the user for further processing.

The important component of HAI is providing user input to agents. A graphical user interface (GUI) can be designed for such a purpose. The details of the interaction protocols that have been used for argumentative conversation are described below.

Interaction Protocol. Interaction protocols (IPs) have been developed to handle the argumentative conversation between agents. An interaction protocol is a set of rules that governs the communication or conversation between software entities (e.g. agents). It determines the communication pattern and message sequence during agent conversation. Also, it will reduce the complexity of software implementation (FIPA[2], 2000). Different IPs can be utilized for various conversations with different agents.

Figure 1 shows the interaction between the roles of initiator and participant, which can be represented by different agents. The initiator starts the whole protocol. The protocol starts when there is an enquiry for service by an initiator. In this situation, the participant will either reply with an agreement to process the enquiry or a refusal. An unsuccessful activity will lead to the following activities such as creating a new request, terminating the service, or arguing against the refusal. This will lead to a looping process of argument, counter argument, proposal and counter proposal from both agents until satisfaction with the negotiation results or there is failure of the negotiation process. An accept message will result in the success of the entire request with a physical or mental outcome. The physical outcome is categorized as a result that has physical world practical implications for the user such as a successful hall booking, a successful reservation, a successful ticket purchase and so on. The mental outcome is categorized as users' experience such as accepting the fact from a resulting argument.

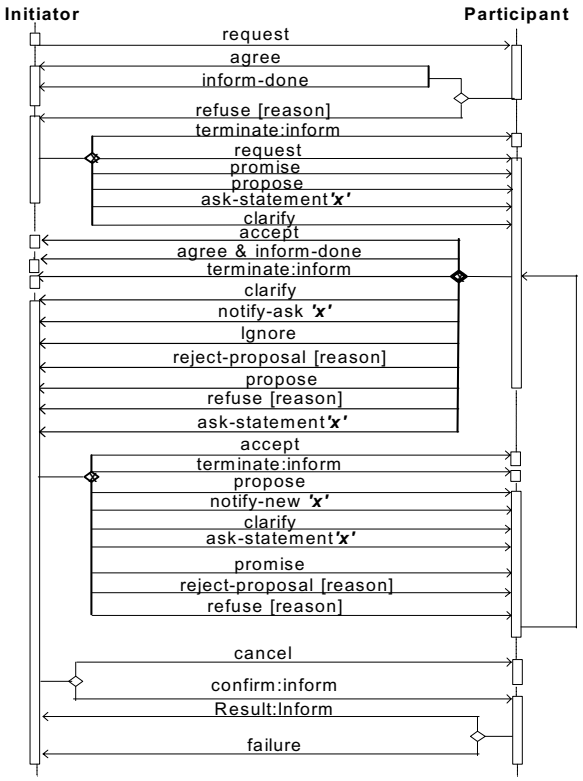

Figure 1: The General Interaction Protocol for Argumentative Conversation between agents

There are different elements for argumentation - both parties can argue by clarifying the importance values of the item, clarifying their preferences or limitations and notifying their internal behavior such as expressing likes, dislikes, usage history, purpose of event, owner's role, and changes of decision. Also, they can argue by promising rewards, informing consequences, increasing the rewards or reducing the negotiation component, and exploring the opponent's behavior like asking for

[2] http://www.fipa.org/specs/fipa00037/SC00037J.html

purpose, claiming a reward, reward offering, increasing the variety of reward offers, increasing the value of a reward offer, reducing the extent of negotiation and declaring roles. During an argument, both parties can decide to accept the argument, provide counter-argument, ignore the argument, or terminate the entire conversation. This termination process will result in an unsolved conflict or negotiation failure. Meanwhile, the successful request or argument enables the initiator to perform cancellation or confirmation for the future.

The communicative acts used in our system are divided into two categories. The first category is the FIPA communicative acts that have been used throughout the design. Meanwhile, the second category is the new primitives that have been created for the argumentation-based system. Argumentation is identified by keywords like "notification", "notify-statement", "promise" and "ask-statement". "Notification" and "promise" are used for expression and "ask-statement" is used for exploration.

3. Implementation

As mentioned earlier, the Flexible Room Booking system is an agent based room-booking system. The architecture of the system consists of different types of agents with different roles and functions, which is depicted in Figure 2. It can be described as a client-server system with login agent, booking agent, query agent, negotiation agent at the client side and lecture theatre agent and confirm agent at the server side. The execution of the Flexible Room Booking system involves the interaction among agents and database processing. Some of the interactions among the agents are described as below:

- Instructing AgentGenerator for agent's generation from GUIAgent.
- The interaction among the LoginAgent and lecture theatre agent (LTAAgent) for validation process.
- The interaction between the QueryAgent and LTAAgent for query process.
- The interaction between the BookingAgent and LTAAgent for booking process. In this case, both agents will utilized the IP derived from Figure 1.
- The interaction among the BookingAgent and negotiation agent (NAAgent) for negotiation process.
- The interaction among the GUIAgent with the NAAgent, BookingAgent, QueryAgent and LoginAgent for displaying purpose.
- The interaction between the ConfirmAagent and NAAgent for cancellation process.
- The interaction between the ConfirmAgent and LTAAgent for informing purpose.
- The interaction among the LTAAgent with the database for data management. The interaction among the LTAAgent with NAAgent for negotiation process. In this case, both agents will utilized the IP derived from Figure 1.

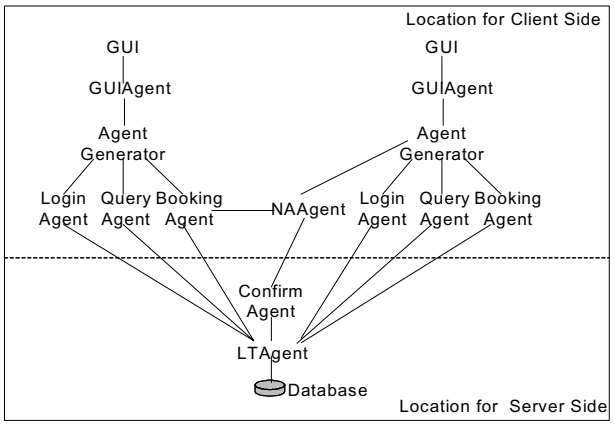

Figure 2: The Architecture of the Flexible Room Booking System

The functionalities of the agents are identified via their names, which described below.

The **AgentGenerator** is the stationary agent that will generate different types of task-oriented agents (login agent, query agent, booking agent) for handling the instruction from the user.

The **LoginAgent** is the mobile agent that will migrate to the server side for the validation process.

The **QueryAgent** is a mobile agent that will perform the query request after receiving the input from the user.

The **BookingAgent** is a mobile intelligent agent with the capability to perform the activities listed below: (1) Make booking or handle the booking process. (2) Negotiate for the unavailable timeslot using the argumentation-based negotiation process. (3) Refer to the user for additional information and important decision. (4) Argue about the proposal that has been given by the LTAAgent.

The **negotiation agent (NAAgent)** acts as a contact agent for the user (interfaces the user to the negotiation process). It is an intelligent agent with the capability to perform the activities listed below: (1) Handle the negotiation process by using argumentation-based negotiation. (2) Pass the argument from the BookingAgent to the GUIAgent for displaying purposes. Some of the arguments are express like, dislike, reward and so on. (3) Disturb the user when necessary for decision-making. (4) Refer to the user for incomplete information and important decisions. (5) Handle the cancellation process upon the agreement to give up the room for the other user.

The **LTAAgent** acts as the server for the System. It handles the incoming requests from the agents and provides the services to agents.

The **ConfirmAgent** is generated by LTAAgent to prevent any illegal activities from the BookingAgent.

A workthrough example of the system. The booking process enables the BookingAgent to make a booking for a particular timeslot. Failure to do so, the BookingAgent will provide a list of potential actions to its owner (the user) for further activity. After selecting to negotiate by the user, the BookingAgent will migrate to deliver the negotiate request to LTAAgent. Meanwhile, the LTAAgent will check the cancellation date and confirmation record for this particular request and propose an available timeslot to the BookingAgent. The BookingAgent then asks the user if it should negotiate further about the proposal, ignore the proposal or accept the proposal. The acceptance of the proposal will lead to a successful outcome.

Negotiation between user (represented by BookingAgent) and the system (represented by LTAAgent): Negotiating about the proposal means that the BookingAgent argues autonomously with the LTAAgent (provided the user has assigned full responsibility to the BookingAgent).

Negotiation between two users (one represented by BookingAgent and the other by an NAAgent): Ignoring the proposal will cause the BookingAgent to resend the negotiate request to the LTAAgent. The LTAAgent will then check the previous reply and respond with the opponent's (opponent refers to the person who has booked the room for the slot being requested) details. The BookingAgent will display the response to its owner together with the collection of negotiation strategies before proceeding to negotiate with the NAAgent (representing the opponent). For the negotiation process, the NAAgent will be assigned the level of interruption as specified by its owner. This level of interruption will determine the degree of autonomy for the NAAgent. Furthermore, the opponent can adjust his/her NAAgent's autonomy by leaving messages for it. The NAAgent will use the contents of the messages to modify its negotiation strategy and adjust the level of interruption/interaction with its owner. The BookingAgent might go back to its owner for additional information during the negotiation process.

4. Conclusion

Negotiation between users and between users and the system as mediated by the mobile agents increases the flexibility of the system. Typically, if a room is booked, no one else can book it, but in this system, the user who wants the room can initiate negotiation (with the help of agents) with the user who holds the booking. We have developed a system that permits flexibility in booking rooms (in that negotiation in booking rooms is allowed) and that users can configure their agents who are negotiating on their behalf. Our argumentation-based negotiation approach fits the problem since flexible negotiation is facilitated and the richness of argumentation-based negotiation enables modelling of the complex interactions between users (and their agents) such as in the booking system. By allowing users to configure the behavior of their agents, the agents become more controllable and increase the satisfaction value for the user. We contend that our approach to adding flexibility to resource management systems (e.g., room bookings), where users competing for or sharing resources can negotiate (with the help of agents) with the system or with other users when resources they want are not immediately available, is general, and can be applied to a variety of such systems. Also, the use of mobile agents means different user interfaces (encapsulated in agents) can move into users' devices (thereby without requiring their a priori installation except for a general agent hosting server), and when an agent moves to where the other agent is, intensive interactions between agents can take place locally without heavy network communication, which is particularly useful for users with mobile devices.

For future enhancement, we will look into the multi issues negotiation using the argumentation-based negotiation approach.

5. References

[1] Ashri, R., Rahwan, I., and Luck, M. (2003) Architectures for negotiating agents, In Multi-Agent Systems and Applications III, Proceedings of the 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS), Prague, Czech Republic. Lecture Notes in Artificial Intelligence 2691, Springer-Verlag, pp. 136-146.

[2] Dickinson, I.(1998), Human-Agent Communication, HP Labs Technical Report.

[3] Jennings, N. R. and Wooldridge, M. (1998) Application of Intelligent Agents, Springer-Verlag.

[4] Lewis, M. (1998) Designing for Human-Agent Interaction, AI Magazine, pp. 67-78.

[5] Myers, K. L., and Morley, D. N. (2001) Human Directability of Agents In proceeding on K-CAP '01, Victoria, British Columbia, Canada.

[6] Rahwan, I., Sanerberg, L., and Dignum, F. (2003) Toward Interest-Based Negotiation, Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent System (AAMAS), Melbourne, Australia, ACM Press, pp. 773-780.

[7] Sierra, C., Jennings, N. R., Noriega, P., and Parsons S. (1997) A Framework for Argumentation-Based Negotiation, Proceedings of Fourth International workshop on Agent Theories, Architectures and Languages, pp 167-182.

[8] Toda, Y., Yamashita, M., and Sawamura, H. (2001) An Argument-based Agent System with KQML as an Agent Communication Language, 4th Pacific Rim International Workshop on Multi-Agents, Taiwan.