Multi-agent Meeting Scheduling Using Mobile Context

Kathleen Yang, Neha Pattan, Alejandro Rivera, and Martin Griss

Carnegie Mellon University, Silicon Valley Campus
NASA Research Park, Bldg. 23, Moffet Field, CA 94035
{kathleen.yang,neha.patttan,alejandro.rivera,
martin.griss}@sv.cmu.edu

Abstract. Despite the use of newer and more powerful calendar and collaboration tools, the task of scheduling and rescheduling meetings is very time consuming for busy professionals, especially for highly mobile people. Research projects and commercial calendar products have worked since the early 1990s on implementing intelligent meeting organizers to automate meeting scheduling. However, automated schedulers have not been widely accepted or used by professionals around the world. Our research shows that the task of organizing meeting can be improved by reducing the scheduling workload and making meeting logistics more efficient. For example, new tools can decide meeting venues and dynamically handle exceptions, such as one participant not being able to arrive at the meeting location on time. In this paper, we discuss a solution to effectively employ the mobile user's context in making more intelligent decisions on behalf of the user. Business Meeting Organizer (BMO) is a multi-agent meeting scheduling system, designed to automate time and venue decisions and to handle exceptions. Rather than using a traditional multi-user calendar-based scheduler, BMO has representatives ("software secretaries" implemented as software agents) for each user to negotiate a best time for a particular meeting, taking into account availability, context and preferences. Several technologies are used by BMO to provide secure and intelligent meeting scheduling functionality. By using agent technology, BMO keeps private calendar information invisible to other meeting participants and allows diverse intelligent negotiation and scheduling policies to be employed. Through the use of a rules engine, BMO can consider meeting participants' personal preferences as to when to schedule the meeting. By means of mobile devices, BMO can get the user's context information, such as the user's physical location, which may be helpful in deciding the meeting venue and handling other meeting issues. In this paper, we discuss a solution to effectively employ the user's mobile context information in making more intelligent decisions on behalf of the user.

Keywords: context-aware; intelligent meeting scheduling; multi-agent system.

1 Introduction

In this age of constant change, one of the most effective methods to stay in touch, maintain business connections, share information, and make decisions is through meetings. It is, therefore, not surprising that many professionals spend significant

S. Ystad et al. (Eds.): MobiCASE 2009, LNICST 35, pp. 223–235, 2010.

amounts of time in attending meetings [1]. Much time is expended in negotiating different factors like time, venue, attendees and resources when scheduling and rescheduling meetings. The current methods of solving this problem, namely use of email, SMS or phone calls are time consuming and inefficient. It is believed that autonomous agents can significantly reduce this workload and the information overload by automating these tasks for users [3]. The problem is exacerbated for highly mobile users who may change locations frequently and are typically pressed for time in setting up meetings.

In this paper, we propose a distributed meeting scheduler for mobile devices – Business Meeting Organizer (BMO), which largely automates the process of meeting scheduling through communication between multiple software agents to negotiate and decide on the specifics of meetings. Each user has one software agent that acts as their personal secretary. This software secretary resides on their personal computer or on a secure company server and acts on the user's behalf even when the user is travelling or inaccessible. The user communicates with the software secretary using their mobile device. The secretary has access to all of the user's personal information such as calendar, contacts, current context, and preferences. The software secretaries obtain requests from their respective users and communicate with each other to decide when and where to arrange meetings. These secretaries also understand the users' preferences and priorities and make decisions accordingly.

In addition to accessing static information about the user through his/her contacts or calendar, the software secretaries also stay informed of the mobile user's context such as location, time of day or current activity. In addition to the current context, the software secretaries are also aware of how certain parameters of the context will change over time. For example, by using the user's calendar, the secretary can predict the future location of the user and may even give hints regarding activity like future travel, upcoming projects, and deadlines. The context of the user plays a very important role in managing the user's schedule for the day and in communicating important decisions back to the user. For example, if the user is going to be in San Francisco until 10 am in the morning on Friday, the secretary will not arrange any meetings for him/her at San Jose any time before 11 am (because it typically takes at least one hour to drive between these two locations). After scheduling the meeting, the secretary will send the user a confirmation with a link to the map of the exact location. If one of the users attending the meeting is not going to be able to make it to the meeting on time due to traffic congestion or if (s)he gets caught up in some other meeting, the software secretary will dynamically reschedule the meeting, notify the other attendees and inform of the revised meeting details. The software secretaries are modeled to enhance user experience by allowing soft and hard constraints on the meeting scheduling decisions. An example of a soft constraint would be: "I would rather not have meetings on Fridays, but if no other time is available, then it's ok". A hard constraint would be: "From 1 to 2 pm on weekdays, I'm not available since I am in a class".

In addition to location, we have considered other sources of context such as the time of day and related activity. Based on the user's context and preferences, the

software secretary will be able to make intelligent and informed decisions and communicate them back to the user.

In the rest of the paper we first discuss background and related work in Section 2, then describe the scheduling model and architecture of BMO in Sections 3 and 4, our prototype implementation in Section 5, our initial user tests in Section 6, and finally conclusions and future work in Section 7.

2 Background and Related Work

Since the 1990s, several groups have worked on intelligent meeting organizers, such as Ameetzer [3] and CoolAgent [4]. Generally these systems focus on negotiating and deciding a suitable timeslot for meetings based on the participants' availability and personal preferences. Many of the proposed meeting organizers also focus on the use of user preferences [2] and adaptation to the user environment [5]. Some of these proposals focus on learning opportunities [6] for the user's preferences and for preferences of other users within the system based on their agents' responses. They help to reduce people's workload of scheduling meetings. Our work builds on these ideas, but specifically considers the context of the mobile user in order to make intelligent and informed decisions on behalf of the user. For example, it would be helpful for meeting organizers to decide the venue for a meeting in a way that's more convenient for all participants, based on their current location or planned location at the proposed meeting time. It would also be of great help if the meeting organizer could find a facility for a participant to join a meeting remotely if he/she is not able to arrive to the meeting on time (for example a coffee shop with Wi-Fi). It is also helpful if a meeting organizer can help (or automatically) reschedule or cancel a meeting if some attendees become unavailable for a meeting because their previous meeting got delayed or other similar reasons. As such, there are several decisions that can be made on behalf of the meeting attendees that can greatly enhance the usability of a meeting organizer. Many of these decisions are tightly coupled with the context of the user and can be easily modeled using a mobile device to obtain contextual information.

The issue of user context has not been fully addressed in designing a multi-agent personal assistant based meeting scheduling system. While similar ideas have appeared in [7], his solution consists of a broker architecture in which the broker is responsible for providing knowledge sharing service between agents and maintaining a shared model of the context. Chen proposes that the system should support policy-driven privacy protection by allowing users to define how much information is shared and with whom it is shared. In our approach, we focus on user privacy and security needs by only allowing user information to be accessible to the user's personal software secretary. Since there is no central model of the context of all users, no single entity has access to all of the information. The preferences specified for the software secretary will define what information can and cannot be shared with other software secretaries or users.

3 BMO Meeting Scheduling Model

In order to make a meeting decision, several factors need to be taken into consideration. These factors can be summarized in a scheduling tree as depicted in Figure 1.

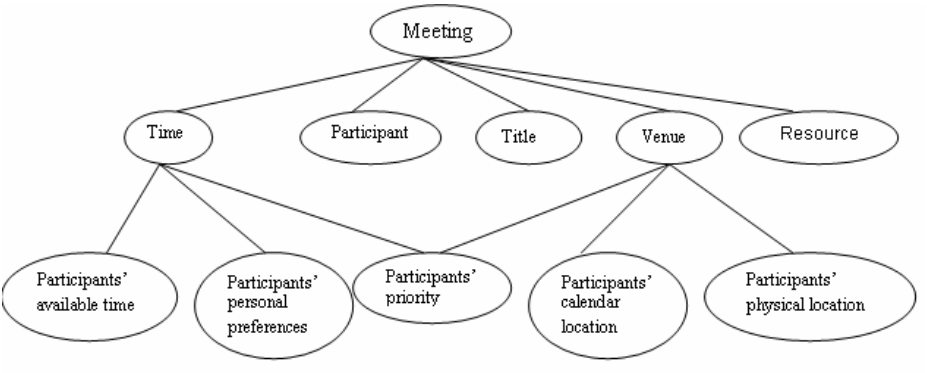


Fig. 1. Meeting Scheduling Tree

Some of the nodes in figure 1 are explained as below.

Participants' personal preferences
This node means people's time and venue preferences, for example, some people prefer to have meetings in the morning while some people prefer to have meetings in the afternoon.

Participants' priority
This node indicates how important participants are for the meeting. When a low priority person cannot attend a meeting, the meeting might not need to be re-scheduled.

Participants' physical location
This node indicates where the participants physically stay at one specific time point. For example, when a person is sitting in Mountain View library at 8:00pm, his physical location is Mountain View library. The basic location information is GPS coordinates. The outdoor GPS coordinates are retrieved from the smart phone GPS module and the indoor GPS coordinates are retrieved from GSM based locationing services. Based on the GPS coordinates, a location label based system is maintained. Each label represents a meaningful place for one particular user, such as home, office, or a geocode address. A group of GPS coordinates are associated with one label according to the user's explicit decision or some machine learning rules. For example, when the system detects a pair of GPS coordinates without associated any label, it can either

ask the user to manually input the label or assign an existing label to it if it's close enough, for example, within 10 meters.

Participants' calendar location
This node indicates where the participants stay at one specific time point according to their calendar information. For example, if the calendar shows that a person has a meeting at NASA Research Park at 10:00am; his calendar location at that time is NASA Research Park. A participant's calendar location is important because we cannot predict one's location based on his physical location but can predict based on his calendar location. Then, we can use the predicted location to decide meeting venue which can minimize the travel cost from previous meeting to next meeting. Different from the participant's physical location, his/her calendar location doesn't have any GPS coordinates. Only the label-based location system is used and the labels are subtracted from the calendar system.

Resource
This node indicates meeting resources such as teleconference system, projector and the size of the meeting room.

The other nodes are pretty self-explained, so no description is given here.

When a software secretary in the BMO system wants to schedule a meeting, it needs to establish the values of all child nodes of the *Meeting* node. If a child node has further children nodes, BMO needs to collect the value of those children nodes until it has reached the leaves. Thus, a meeting can only be decided when all the nodes in the scheduling tree have valid values. The decision on whether to schedule the meeting may be affirmative or negative depending on the exact values of the nodes.

4 BMO Architecture

The architecture of the system is shown in the Figure 2. The external services, the Map Service, Locationing Service, Calendar Service and Contacts Service, are used by the software secretary and the client application. The client application resides on the user's mobile device. Thus, the client application has easy access to all of the user's personal context information from the mobile device; in addition, some other public parts of the user's context are accessed directly by the server (e.g., calendar, history of meeting preferences, etc.) This information may vary from the user's location and activity to specific physical or medical conditions. As more hard and soft sensors get integrated with mobile devices, increased context information will become available to the client application through the mobile device.

The Software Secretary is running continuously on the user's personal computer or on a secure company server, either at home or in the office. Currently, the secretary processes context information of the user's location.

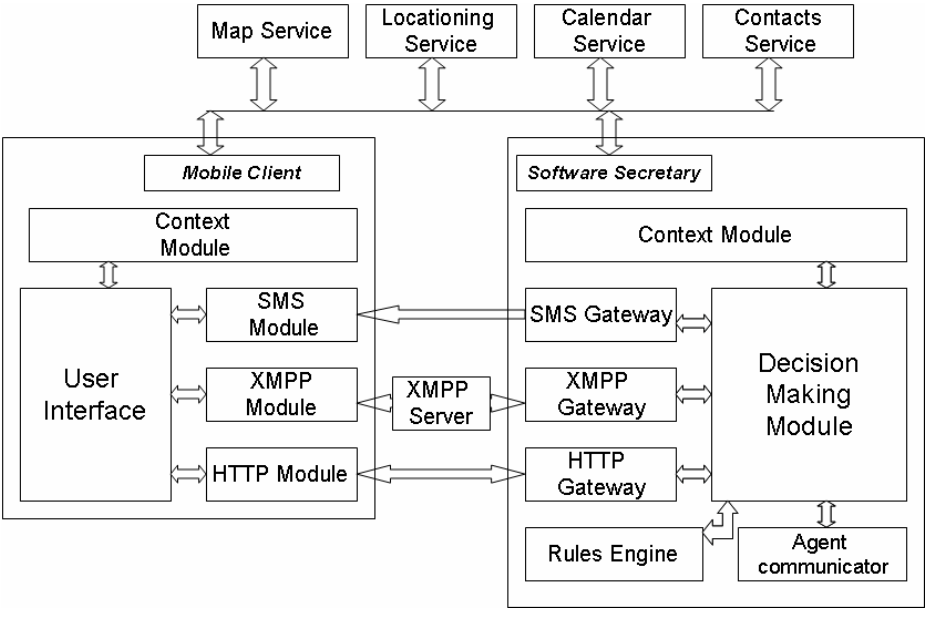


Fig. 2. Architecture of proposed system

4.1 Secretary Agent Side

The Software Secretary is a system consisting of the following components:

Context Module: This module contains the most recent context information obtained for the user. This module is triggered every time the client application sends a context update to the secretary. The context information is used by the decision-making module to make intelligent decisions based on the user's context. This module has a plug-in design, allowing us to add or remove several context sub-components that are relevant to the user.

SMS Gateway: This gateway is used to send updates to the user in case the message has a high priority and urgency. It may also be used if the user's mobile device does not support features like Wi-Fi access or data packets.

HTTP Gateway: This is used to communicate with the client application over HTTP. This may be used if the user prefers to communicate directly with the software secretary agent without relying on an intermediary service. (For example, as will be discussed next, XMPP can also be used for communication between the client and the agents. This would, however, require the messages to be routed through an external service). The HTTP gateway consists of a light-weight HTTP server running on the secretary listening for requests sent by the client application.

XMPP Gateway: This gateway is another means of transportation for messages between the client application and the secretary agent. If XMPP is used as the transport,

all messages will be relayed through a third party XMPP server (also known as Jabber server). Using XMPP has the main advantage of messages being stored by the server for delayed delivery in case the destination is not currently accessible or online.

Decision-Making Module: This is the most important component of the secretary agent side system. It is responsible for combining all information from the user's context, external services and user's preferences to make decisions about meeting scheduling.

Rules Engine: This module contains all user preferences in the working memory. The rules are triggered whenever a decision needs to be made by the *Decision Making Module.*

Agent Communicator: This is used by each secretary agent to communicate with other secretaries. JADE [9], being a distributed agent platform, takes care of the communication between secretaries located in the same computer or even in completely independent hosts. The communication is handled transparently in the background, no matter what transport protocol is in use (e.g. Java-RMI, HTTP, IIOP). JADE also relies heavily on the FIPA (Foundation for Intelligent Physical Agents) and its ACL (Agent Communication Language) standards to ensure cross-compatibility with other agent systems.

4.2 Client Side

The Mobile Client software consists of the following components:

Context Module: The context module on the client side is responsible for gathering all context information of the user and will always contain the most recent information. This information is updated by periodically querying different sensors on the device, such as sensors that support location detection or movement.

SMS Module: This module is useful in intercepting the incoming messages sent by the secretary agent. These messages will be processed by the application and deleted from the user's SMS inbox. So, the messages are invisible to the BMO users. This design can avoid large volume of annoying messages in the SMS box.

XMPP Module: This module is used in interacting with the XMPP server to send and receive messages to and from the secretary agent.

HTTP Module: This module is used if the transport used for communication between the client and secretary agent is HTTP. It is used to make requests to the HTTP server running on the secretary agent side.

User Interface: This module contains the interface displayed to the user and is responsible for input and output of information to the user.

5 Implementation

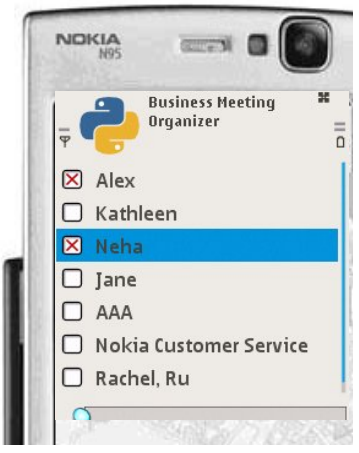
We use Mobile Python [8] (referred as Python hence forth) running on Symbian S60 to implement the mobile client on Nokia N95 smart phones. Users use the mobile client to input the meeting title, preferred meeting time range and expected meeting participants. The advantages of using Python are its high coding productivity and strong support by the Symbian S60 developers and community. Also Python has access to low-level information on the system (the mobile device) that's not typically accessible by some other high-level programming languages (such as J2ME).

The Java Agent Development Framework, or JADE [9], is used to implement the software secretary agents. One software secretary is implemented as one agent in JADE. By using agent technology, the secretary of the meeting initiator doesn't need to access the other meeting participants' calendars. What the secretary needs to do is to ask the other secretaries questions such as "When will you be available today or tomorrow?" This solution helps to protect the privacy of the meeting participants, who may not want to expose their calendar information to others.

The software secretary in BMO makes use of several Google Data APIs (GData for short), such as the Calendar, Contacts and Maps API. As an instance of the Internet service API, Google APIs have several advantages compared to other traditional library APIs. First, the APIs provide functionality as well as abundant data, such as users' contact data and calendar data. Second, the APIs provide high coding productivity. Third, the library deployment is significantly minimized.

Extensible Messaging and Presence Protocol (XMPP) is used for the communication between the mobile client and the secretary. The mobile client and the secretary work as the XMPP client, sending messages to and receiving messages from the XMPP server. In this way, the mobile client can send meeting request to the secretary and the secretary can send meeting confirmation to the mobile client. The secretaries of all the participants can talk with each other to negotiate the meeting scheduling. The advantage of using XMPP is its offline capacity. If an XMPP client is offline, XMPP messages sent to it can be queued in the XMPP server. When the XMPP client connects to the Internet, it can retrieve all the queued messages from the XMPP server. For users who don't have Internet access, SMS can be used instead. For those users who are not using BMO, email can be used as the default method of communication. For instance, the email could contain a simple HTML form the user would fill in and submit it back. Making use of this technique would simplify the data exchange while maintaining a standard communication format.

As illustrated in the following screen shots (Figure 3), when a user wants to schedule a meeting using BMO s/he needs to follow four steps. After that, the meeting request is transmitted to the virtual secretary using XMPP (if connected to the Internet) or is stored in the device's memory while waiting for the Internet connection. Using these five steps, users need less than one minute to schedule a meeting on average.



Step 1. Select the expected participants

Step 2. Select the preferred time range

Step 3. Input the meeting subject and minimum required duration of the meeting

Step 4. Get SMS reminder about the meeting

Fig. 3. Mobile App Screenshots

6 Technology Trade-Offs

During the implementation, we made decisions on several technological trade-off points.

The communication protocol between the mobile client and the secretary agent is one of the trade-off points. Although we decided to use SMS and an Internet protocol to assure the connectivity between the two parts, there are more than one alternative for the Internet protocol, such as JASON, HTTP and TCP. The most important factor we considered is the accessibility of the mobile client to the Internet. Because the data plans provided by most of the carriers are still expensive, it's reasonable to assume that many users cannot access the Internet when they are away from any WIFI access point. So, we decided to use XMPP as the Internet protocol because it offers off-line capacity. For example, the meeting confirmation can be queued on the XMPP server until the participant's mobile client connects to the Internet and thus triggers the XMPP server to send the confirmation to the client.

The programming language on the mobile phone is another trade-off point. Java, C++ and Python are three primary used programming languages on the smart phones. We decided to use Python because it's a powerful, fast and secure prototyping language on Symbian platform. We can use two to three lines of codes to access many capacities, such as GPS capacity, SMS capacity, and Calendar capacity. It doesn't provide any mechanism to explore the layers under the mobile device middle ware or other applications.

7 Experiments and Data Analysis

We observed and measured people scheduling meetings in five ways: a) using BMO, b) using "Friend's calendar" function of Google calendar, c) exchanging emails, d) using the telephone and e) face-to-face negotiation.

We also decided to observe meetings with four participants, because in our survey people stated that when meeting has four participants or more, scheduling the meeting becomes painful.

Table 1. Time for meeting scheduling

	Tools use for scheduling	Number of participants	Negotiation time (minutes)	Request pending time (minutes)
Meeting #1	BMO	4	1	0
Meeting #2	Google calendar	4	2	0
Meeting #3	Email	4	7	50
Meeting #4	Telephone	4	11	0
Meeting #5	Face-to-face	4	5	0

In table 1, the time used for scheduling meetings is recorded for the five different approaches. The negotiation time is the time that people spend to talk with each other, discussion via email and check calendar. The request pending time is the period of time during which people wait for the email response. This request pending time is specific to scheduling meeting via emails.

From the data, we found that using email is the least efficient approach because the negotiating uses seven minutes and the time waiting for email response is 50 minutes.

Telephone and face-to-face are better than email because there is no pending time between the time when the meeting is requested and when the confirmation is responded. However, they still cost five and eleven minuets respectively. It's still painful for meeting organizers.

The Google Calendar can help people organize such a meeting similarly efficient to BMO. However, it has several disadvantages compared to BMO.

First, all meeting participants must expose their calendar to others. Many people don't feel comfortable about this privacy exposure. Second, Google Calendar doesn't consider context information, such as people's personal preferences, location and priorities. Third, people need to check others' availability manually. It's unpleasant work when a meeting has four or more participants.

In contrast, BMO can protect people's privacy, consider context information and automatically negotiate and decide a meeting time slot and venue.

Figure 4 displays the communications between three different secretaries (Kathleen, Neha and Alex's) while arranging one meeting.

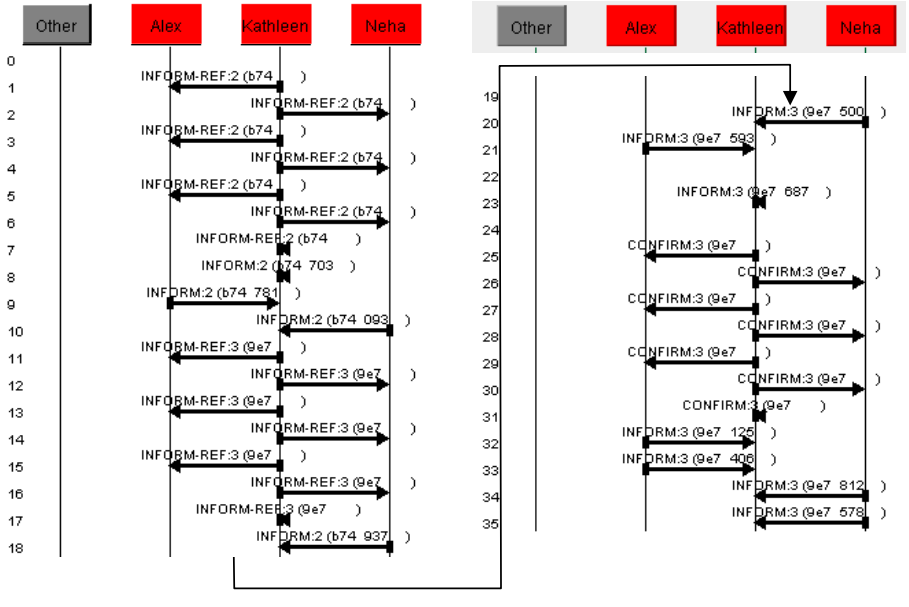


Fig. 4. Agent communication

In Figure 4, it is easy to see that there are many messages being exchanged, yet the time it takes these software secretaries to organize the meeting is in order of a few seconds. If the meeting participants were to employ traditional means of coordinating a meeting, such as discussing the options by telephone, the number of distinct communications among the participants would potentially be much higher and the results might not be as optimal.

8 Conclusion and Next Steps

As discussed in Section 6, we observed through several tests that the time taken to schedule and reschedule meetings with the intelligent meeting organizer is a fraction of the time spent in performing the same tasks manually by using email or calling all attendees. In addition, our solution also scales well when meetings require many more attendees for whom location and time constraints need to be considered. The next step for us is to take the system out to test in the field with additional users and to analyze data on usability, reliability and effectiveness of using the tool.

Currently, the system requires all users to have an installed and running software secretary. BMO is built to require communication between each of the secretary agents representing the various participants to negotiate the meeting time and venue details. We would like to make a provision for secretaries to be able to communicate with other users who may not have software secretaries. In this case, if a secretary agent is requesting a meeting to a user who does not have a secretary agent, the requester agent sends an email or text message to the user directly using standard meeting requests (such as iCal attachments) or by providing a web form the user could fill in that the software secretaries would be able to understand to continue the negotiation process.

These secretary agents can be implemented as part of the client application on the mobile device rather than on the user's personal workstation. We plan to prototype such a standalone intelligent meeting organizer on the Android platform using lightweight agents provided by the Light Extensible Agent Platform (LEAP) for mobile platforms.

We would also like to improve the preference module to allow more interesting scheduling patterns. For instance, allowing the user to decide whether or not to schedule adjacent meetings (with no breaks). Another example is to inform the secretary that instead of selecting the next available timeslot for the meeting, it should try to balance all meetings throughout the week. Perhaps users would also like their secretaries to know that it's best for them to have meetings early in the day or to respect lunch hours at all costs, etc. Another kind of preference can be related to the progress of particular projects. For example, if the project is running late and a bottleneck has been detected, it's of extreme importance to schedule a meeting to address the issue and all participants should be aware that this meeting has priority over all others.

Another area we will investigate is to associate some form of weighting or economic value to decide which meetings are more important or urgent than others; for example, a meeting with the CEO is sufficiently important that even canceling or rescheduling other meetings or a planned trip is appropriate. Similarly, user preferences and soft constraints can associate weights with alternative meeting slots

In addition to understanding user preferences and context, the tool can be made more powerful by learning the behavior, preferences and context of the user. This can be achieved by using appropriate machine learning algorithms within the secretary agent to achieve a high level of intelligence within the agent.

We believe that, in the future, automated software agents will play a very important role in simplifying the lives of business professionals and organizing meetings will be one of the several functions carried out by them.

Acknowledgements

This research was supported by grants from Nokia Research Center and the CyLab at Carnegie Mellon under grant DAAD19-02-1-0389 from the Army Research Office. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ARO, CMU or the U.S. Government or any of its agencies.

We also acknowledge the efforts of Patricia Collins and the anonymous referees for their valuable feedback and suggestions on earlier drafts of this paper.

References

1. Maes, P.: Agents that reduce work and information overload. Communications of the ACM 37(7), 30–40 (1994)
2. Haynes, T., Sen, S., Arora, N., Nadela, R.: An automated meeting scheduling system that utilizes user preferences. In: Proceedings of the First International Conference on Autonomous Agents (1997)
3. Lin, J.H., Wu, J., Lai, S.L.: A Multi-agent meeting scheduler that satisfies soft constraints. In: Proceedings of the Second International Conference on Multiagent Systems (1997)
4. Griss, M., Letsinger, R., Cowan, D., Vanhilst, M., Kessler, R.: CoolAgent: Intelligent Digital Assistants for Mobile, Professionals – Phase 1 Retrospective, HPL Technical Report (November 2001)
5. Sen, S., Durfee, E.H.: On the design of an adaptive meeting scheduler. In: Proceedings of the Tenth Conference on Artificial Intelligence for Applications (March 1994)
6. Crawford, E., Veloso, M.: Opportunities for learning in multi-agent meeting scheduling. In: Proceedings of the AAAI 2004 Symposium on Artificial Multiagent Learning, Washington, DC (2004)
7. Chen, H.: An intelligent broker architecture for context aware systems (2002)
8. Scheible, J., Tuulos, V.: Mobile Python: Rapid Prototyping of Applications on the Mobile Platform. Wiley, Chichester (2007), ISBN: 978-0-470-51505-1
9. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley, Chichester (2009), ISBN: 978-0-470-05840-4