

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI[®]

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

THE DEVELOPMENT OF AN AUTOMATED MEETING SCHEDULER

HOOMAN SALAMAT

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

MARCH 1999

© HOOMAN SALAMAT, 1999



National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-39117-5

Abstract

The Development of an Automated Meeting Scheduler

Hooman Salamat

The Virtual Secretary is a software agent that can act on behalf of its user's secretary. This dissertation presents an introduction to automating a secretarial task, that is, meeting scheduling. Meeting scheduling is a part of the Virtual Secretary's task that negotiates on its user's behalf to arrange a meeting at a suitable time for all attendees according to their priorities and preferences. In this sense, Virtual Secretary produces a compromise solution, which is a tradeoff between conflicting attendees' priorities and preferences. The attendees only have knowledge of their own calendar and preferences, which could be changed or fixed over time. The host of the meeting asks for a meeting, but Virtual Secretary is the central agent that monitors the negotiation. The assumption is that Virtual Secretary locates attendees by e-mail addresses to send them the meeting invitations. The attendees and the host of the meeting interact with the system through the Internet. The users receive their invitations about the pending meeting via e-mails and request their meeting, confirm their attendance in the meeting, cancel their meeting, get the status of their meeting, and prioritize their attendees on-line through the Internet. Virtual Secretary processes scheduling of the meetings off-line on the host machine.

Acknowledgements

This dissertation represents a great deal of time and effort not only my part, but also on part of my advisor Peter Grogono. He has helped me shape my work from day one, and encouraged me to achieve to the best of my ability. I would like to thank Peter for his support and suggestions that have had a significant impact on this work. He gave me the opportunity to develop the Virtual Secretary application. Without Peter, this dissertation would not have happened.

My parents, Reza and Agdas Salamat, have always been, and continue to be, there for me all times. They have been incredibly supportive, understanding, and encouraging as they have gone through the entire graduate experience with me. This dissertation is dedicated to my parents, who keep me connected to the world away from the keyboard, and remind me every day how good it feels to learn new things.

While I cheerfully share the credit for the accurate and educational aspects of this dissertation, the mistakes and omission I have to claim as mine alone. Please bring them to my attention.

1	INTRODUCTION	1
1.1	SOFTWARE AGENT	1
1.2	VIRTUAL SECRETARY	2
1.3	POSSIBLE TASKS FOR A VIRTUAL SECRETARY	2
1.3.1	<i>Directory Service</i>	3
1.3.2	<i>Notification Service</i>	3
1.3.3	<i>Bulletin Board Service</i>	3
1.3.4	<i>Meeting Scheduling Service</i>	3
1.3.5	<i>Text-Speech Conversion Service</i>	4
1.3.6	<i>Message Forwarding Service</i>	4
1.3.7	<i>Message Query Service</i>	4
1.3.8	<i>Remote File Management Service</i>	5
1.3.9	<i>Fax Service</i>	5
1.3.10	<i>Multilingual Service</i>	5
1.4	SECURITY	6
2	MEETING SCHEDULING & VS	7
2.1	INTRODUCTION	7
2.2	PRINCIPLE CONTRIBUTIONS	8
2.3	MULTI-AGENT SYSTEM	11
2.4	KNOWLEDGE-BASED SYSTEM	11
3	BACKGROUND	14
3.1	SURVEY	14
3.2	PROPAGATION OF VS	15
3.3	USER PREFERENCES	16
3.4	THE APPLICATIONS	17
3.4.1	<i>Wildfire</i>	17
3.4.2	<i>Portico</i>	18
3.4.3	<i>Microsoft Schedule +</i>	18
3.4.4	<i>Meeting Maker XP</i>	19
4	USER AND TASK MODEL	20
4.1	USER	20
4.2	TASK	21
4.2.1	<i>Users' Goals</i>	22
4.3	MODELING	22
4.3.1	<i>Metaphors for the interface design</i>	23
4.3.2	<i>Use sequences</i>	23
4.3.3	<i>User model</i>	24
4.3.4	<i>Proposed solution : Knowledge-based CGI systems</i>	25
5	DESIGN	27
5.1	MEETING SCHEDULING AUTOMATION	27
5.2	INFORMATION NEEDED FOR COOPERATION	28
5.3	SCHEDULING ALGORITHM	28
5.4	PREFERENCES	31
5.5	PRIORITIES	31
5.6	QUORUM REQUIREMENTS	32
6	IMPLEMENTATION	34
6.1	THE PROGRAMMING LANGUAGE	34
6.2	COMMON GATEWAY INTERFACE	35

6.3	PLATFORM.....	36
6.4	ARCHITECTURE	37
6.5	THE MODULES.....	38
6.5.1	<i>Menu.html</i>	38
6.5.2	<i>Reqform.html</i>	39
6.5.3	<i>Submitreq.cgi</i>	42
6.5.4	<i>Confirm.html</i>	46
6.5.5	<i>Confirm.cgi</i>	47
6.5.6	<i>Submitconfirm.cgi</i>	49
6.5.7	<i>Status.html</i>	50
6.5.8	<i>Status.cgi</i>	51
6.5.9	<i>Priority.html</i>	53
6.5.10	<i>Priority.cgi</i>	54
6.5.11	<i>Submitpriority.cgi</i>	56
6.5.12	<i>Cancel.html</i>	57
6.5.13	<i>Cancel.cgi</i>	58
6.5.14	<i>Schedule</i>	59
6.5.15	<i>E-mail</i>	61
6.5.16	<i>Security</i>	62
7	SECURITY.....	64
7.1	INTRODUCTION	64
7.2	RISK	65
7.3	CGI SCRIPTS.....	66
7.4	FORMS.....	67
7.5	USER AUTHENTICATION.....	67
7.6	PLATFORM.....	69
7.7	OUR SUGGESTIONS	69
7.7.1	<i>Using SSL</i>	69
7.7.2	<i>Using SHTTP</i>	70
7.7.3	<i>Using Personal Certificates to control server access</i>	70
7.7.4	<i>Using Public-key encryption</i>	71
7.8	RESULT.....	72
8	CONCLUSION	74
8.1	IN GENERAL	74
8.2	FUTURE WORK	74
9	REFERENCES	76

1 Introduction

1.1 Software Agent

An agent acts on behalf of its user. A software agent is a particular type of agent that assists users with computer-based tasks. Software agents differ from each other based on the nature of the tasks performed, the nature and source of intelligence, and mobility. Software agents also can learn the user's preferences. In terms of intelligence, software agents rely on a programmer's skill. Figure 1 illustrates the relationship between software agent, applications, and users.

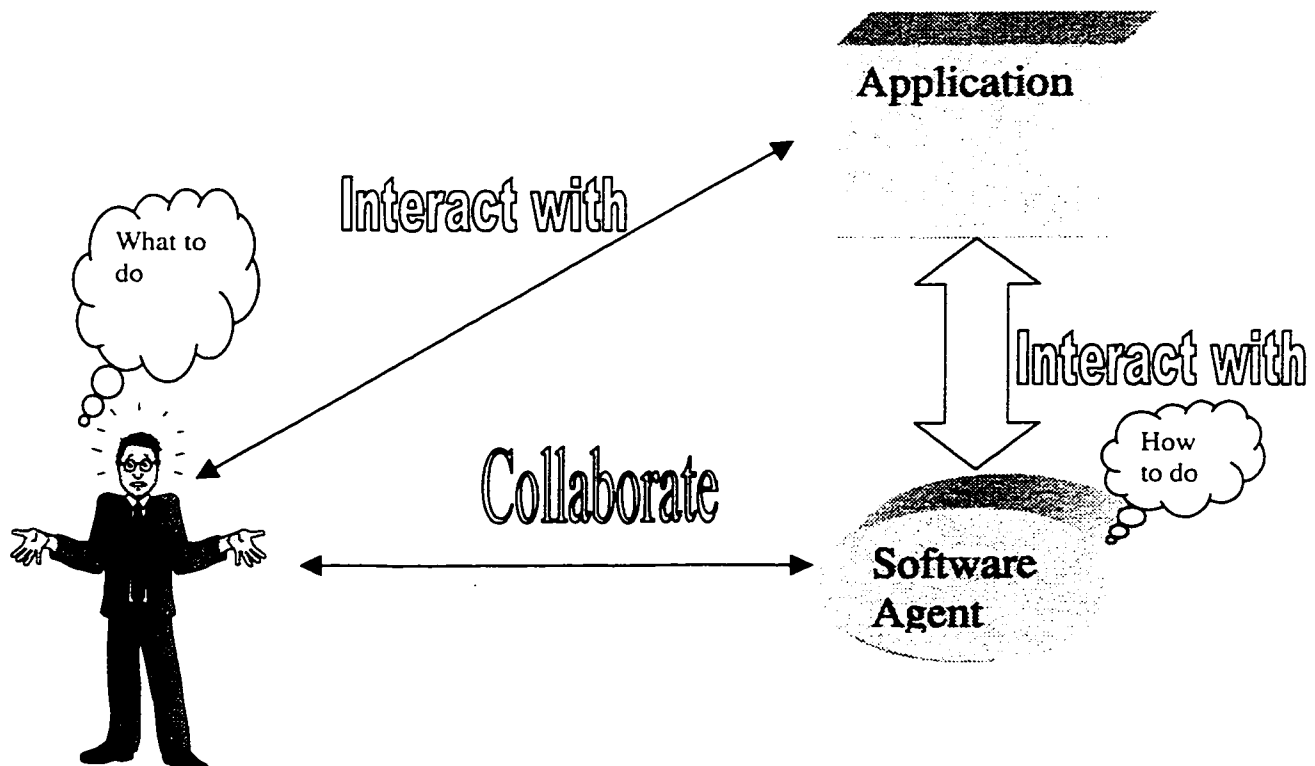


Figure 1

1.2 *Virtual Secretary*

The desirability of computerizing major secretarial tasks in order to achieve higher efficiency in terms of time and accuracy in the real world provided the motivation for building a Virtual Secretary (VS). The main goal was to construct a software agent that learns to perform important secretarial tasks based on requests from users. The next step could be to model other Virtual Secretaries to cooperate with each other in a multi-agent system.

To build a Virtual Secretary, we imitate the activities of a secretary in real life. Then we think about how much intelligence would be enough for the Virtual Secretary. VS could accomplish its tasks using a knowledge-base system, and could update its knowledge by learning or by getting all the required information from users on-line.

In this work, we describe a VS that prompts its user in an interactive environment, in order to receive the essential information on-line, and performs the scheduling process off-line.

1.3 *Possible tasks for a Virtual Secretary*

The following services are some of the most important tasks that we expect that an ideal VS can take care of.

1.3.1 Directory Service

The Directory Service allows the user to maintain vital information about clients, friends and relatives - such as pager and telephone numbers, addresses, etc. Whenever the users wish to access such information, they just need to hook to their VS and it will give them the required information.

1.3.2 Notification Service

The alarm service helps users to organize their business. The users could inform VS in advance indicating the time they wish to get their wake-up call. Suppose, the users have a series of meetings lined up for the day, and they need to be reminded of the next meeting 15 minutes in advance. VS will call the users and remind them about important events.

1.3.3 Bulletin Board Service

VS can maintain a Bulletin Board where the users can store any information. VS will share the information that they wish to reveal. VS could keep the user's client updated round the clock on the latest news.

1.3.4 Meeting Scheduling Service

The automated meeting scheduler accepts a call-for-meeting request from the users to find a common free time slot. When such a free time slot is not found, VS should ask the

busy participants to change their schedules. The other type of meeting scheduling could be scheduling in a multi-agent-based environment where each agent knows its user's preferences and calendar availability in order to act on behalf of its user.

1.3.5 Text-Speech Conversion Service

If the users are leaving their office, VS will store all the messages for them. It notifies the users by a pager or the users can call VS from the outstation location everyday, if necessary, and VS will read out their electronic messages or fax for them. If the users want to reply their messages over the phone, VS will recognize their speech and convert the messages to text, and finally forward them as e-mails to the senders.

1.3.6 Message Forwarding Service

Whenever the users go to another location, all their messages from their office will automatically reach them there. Of course they have to inform their VS of their travel plans and duration of stay in advance. If the users are in a meeting and they don't want to be disturbed for a few hours, VS will re-direct all their messages to a friend, partner or colleague. VS will even inform all callers that the message is being diverted to an alternate address. The users can even retrieve all their re-directed messages directly from VS later.

1.3.7 Message Query Service

There may be occasions when the users might recall their messages. The reasons could be many. They may have received an incomplete message. Whatever the reason, they can connect to VS and request to send them all the messages received, for examples over the

past 3 days remotely. VS could supply the users by search engines and the users could sort results by date, time, word or any other priority.

1.3.8 Remote File Management Service

This service provides the users with access to archived documents through the Internet or the Intranet. The users can also manage the information from different sources (images, applications and files) remotely. A scenario would be when the users want some files from one of their workstations and they don't remember the file name and file server as well. VS carries with it a certain amount of knowledge of the users and it uses this knowledge (called "user model") to guide the system in searching of the correct file. This could be handled by kind of adaptive software, which contains a user model to improve the interaction. The user model could be copied to the remote machine as part of VS.

1.3.9 Fax Service

The users can receive their fax through VS. They can store a copy of their fax or have them automatically sent to their fax machine. VS can send fax to any machine the users choose from any location. VS can broadcast their fax to thousands of numbers all at once. If the line is busy, VS will keep calling till the fax gets through. If the users are out of the office, VS will convert their fax to voice and read it for them on the phone.

1.3.10 Multilingual Service

VS could express information and communicate with the users in any language.

1.4 Security

Global internetworking has enabled many new services and opportunities, including the use of different kinds of software agents. Software agents in the form of worm-like programs represent a great potential for location-independent problem solving. The main problem with such software agents is that unknown programs, which propagate to a computer, must be trusted, since the users have no control over these programs. It is, of course, possible to accept only active programs from users with whom we have an authenticated agreement. Given all the possibilities that the computer network prevents the computer fraud, a user or system manager still takes a risk in letting active programs penetrate the system.

One approach to deal with the security problems is to adopt a security policy that is designed to ensure appropriate levels of security. Our VS is supported by a method of security, known as user authentication. Chapter seven explains this issue in detail.

2 Meeting scheduling & VS

2.1 *Introduction*

Meeting Scheduling is one of the everyday secretarial tasks that is iterative, time consuming, and tedious. The process of searching for a commonly available time through e-mail or phone can result in less satisfactory solutions by real secretaries due to the communication delays and other concurrent scheduled meetings. Meeting scheduling automation could save time and effort on the part of humans. The benefit of such software is two-fold: VS allows users to concentrate on more productive tasks and they improve the quality of information processing by preventing errors that might be introduced by human users due to the routine and tedious nature of the job in question.

To effectively act as a surrogate for the user, VS must have some model of the user. The model can either be input by the user or learned from interaction with the user. Along with the model there needs to be a decision-making system.

Our work has focused on the problem of how the user model can be analyzed and understood such that VS carries out tasks on behalf of human users. Our particular domain has been meeting scheduling and our user model is constructed from the user's input.

2.2 Principle Contributions

The following dimensions represent the capabilities of our automated meeting scheduling system:

- **Users:** The focus of this work is the automation of most of the tedious and repetitive activities that are performed by humans. Human input is critical for VS to represent the user model. For interaction to VS, the users only need to have access to the Internet. In contrast to most of the currently available software for meeting scheduling, the users can interact with the system at any time from any location. Our VS also minimizes the amount of supervision that the user must provide.
- **Process:** All of the scheduling process is done off-line. Thanks to multi-processing operating systems, VS handles as many processes as exist in the system simultaneously.
- **Platform:** Thanks to the Internet, the users don't need any specific platform to work with VS. The only thing the users need to interact with the system is access to the Internet and e-mail address for receiving invitation mails. In this sense, VS is completely platform-independent. On the other hand, the program itself can be installed on PC, workstations, etc., since this program has been written in Perl programming language. VS exists on the target host and users don't need to install any application or software on their site in order to use our VS.

- **Time:** VS allows users to process meeting scheduling asynchronously on-line through the Internet. The time required for all interactions depends on the user's Internet Service Provider, but is usually less than one minute.
- **Group context:** The convention adopted in our approach is that any user can initiate a meeting request. We also assume that the invitees are completely free to accept or reject. The users can give their proposals based on their own preferences and priorities and it doesn't depends on what position they have (hosts or invitees).
- **Preferences:** In our development, we concentrate on user preferences and priorities to categorize acceptable or unacceptable meeting proposals. Based on user preferences and priorities, we propose meeting times and locations.
- **Mobility:** Our VS propagates to the web browsers, bringing data and program to execute locally. The users run their own copy of VS, and VS is distributed across the entire Internet. In other words, VS takes advantage of both centralized and distributed environments.
- **Ease of use:** Working with our VS is simple and doesn't need any computer skill. VS provides the users with a user-friendly environment.

- **Privacy:** Privacy issues are key in our VS. Each meeting has its own password. The invitees have only access to their own meeting information. The passwords are distributed by the host of the meeting via e-mail. There is a risk, of course, that passwords will become known to unauthorized persons. This, and other aspects of security, are discussed in Chapter 7.

- **Quorum:** While scheduling a meeting, in general, it is not a necessary requirement that all potential participants participate. Such a constraint causes many scheduling failures and is avoidable. The goal of introducing the concept of quorum is to find out the best set of members for a meeting if the scheduler finds that it is impossible to schedule all the participants in a proposed time slots and locations.

- **Calendar:** In contrast to most of the meeting scheduler applications, there is no need for our VS and its users to have any calendar application. This extra software might restrict user's mobility for those applications, which need to install the calendars in their premises. The mobile calendars (such as the calendars over the Internet) also need high security and some times providing such a security is not easy if the calendar should be sharable by the users.

The sections 2.3 and 2.4 describe the other capabilities that the software agents might have. In particular, we focus on the structure of a multi-agent and knowledge-based systems. Our VS doesn't support these features.

2.3 *Multi-agent system*

A multi-agent system requires some kind of automatic process, which can communicate with other agents to perform some collective task on behalf of one or more humans. In multi-agent system, VS could learn from other VSs.

The following are major issues with multi-agent systems:

- ❑ The speed of communication in a distributed system.
- ❑ Security and privacy (necessity of having a reliable network to connect agents).
- ❑ Multiple agents might have different perspective and constraints.
- ❑ The communication approach (e.g., LAN, WAN, etc.)

In multi-agent systems, agents are basically organized into a cooperative group for the purpose of knowledge sharing. The following section will explain how the knowledge-based system can exploit the multi-agent system.

2.4 *Knowledge-based system*

Knowledge is the objects, concepts and relationships that are assumed to exist in some area of interest. A collection of knowledge, represented using some knowledge representation language, is known as a knowledge base, and a program for extending and/or querying a knowledge base is a knowledge-based system.

Knowledge differs from data or information in that new knowledge may be created from existing knowledge using logical inference. If information is data plus meaning, then knowledge is information plus processing.

One of the requirements to work with our VS is to retrieve the wanted data from the users directly, which means, the users should go to the host page and enter their preferences and requests in a user-agent interactive environment. Our VS doesn't have any knowledge about the users by default. The user's profile is given individually for each meeting.

In a knowledge-based system, VS has basic knowledge of the users and we name it as the first level of an intelligence agent. VS on this level could fulfill tasks with the help of the knowledge-based system. The key issue for VS on the second level of intelligence is to keep its knowledge fresh and updated by learning. If VS stays only on the first level, it will be useless since the users' preferences and priorities change over time. VS on the third level of intelligence should know how to cooperate with each other to solve problems efficiently. An important characteristic for VS with such cooperative behavior is how to find out *who can help me*, to predict the other VSs activities, and contact them for assistance.

VS could know the others' activities in two ways:

- The most straightforward way is through communication. But, it is time-consuming, and may decrease system performance.

- Agent modeling: In this sense, we have to find out how to model an agent's knowledge for the purpose of intelligent cooperation, and how to update and maintain this information consistently for later prediction.

In general, what kind of information that is appropriate to represent an agent depends on the agent's application domain (e.g., the host of the meeting versus the invitees).

The following information is required to model VS for cooperation purposes:

- The tasks of agents.
- When do agents need cooperation to perform a task?
- What kind of information do agents need for the purpose of cooperation?
 - ❑ VS has no idea how to handle a task.
 - ❑ VS lacks some information to complete a task.
 - ❑ VS only has the ability to perform one fraction of the task.
- Which agent(s) should store this information?
- How to store this information?
- How agents have access to this information (where they exchange the information)?

To satisfy the cooperation process, VS needs a database to store the most updated activities of the other VSs. Whenever VS needs cooperation, VS should extract the above information from the database.

3 Background

3.1 Survey

Past efforts in developing an automated meeting scheduler have met with limited success. For example, [Maes, 1994] focused on learning personal assistants that learn user preferences by watching the user scheduling meetings. The goal was to assist users in making decisions by suggesting alternatives and not to automate the process.

Most of the commercially available products for scheduling over computer networks have been based on PC systems. Microsoft Outlook Meeting Scheduler, CyberMatrix Meeting Manager, Meeting Maker and Microsoft Schedule+ are examples of these products. These products provide the users with only a nice interface to view their own calendars and that of other users, and in some cases find time intervals to propose by searching these calendars. When using these systems, users have to allow complete access to their calendars by all other users. The only restriction is that the users have the sole authority to modify their associated calendar. Calendaring and scheduling products are well established for personal or organizational use [Sen et al., 1997], [Maes, 1994], but they usually are limited to exchange of information among users of the same system, usually within the boundaries of a single organization. None of these systems satisfy the real need for intelligent scheduling that honor user preferences and priorities.

3.2 *Propagation of VS*

Gunnar Hartvigsen, Arne Helme and Stig Johansen (1995) focused on the propagation of the Virtual Secretary's user model. Propagation of the user's environment includes export of the user environment to a remote host. The virtual secretary body process is referred to the part of the application, which performs all the processing. To reduce the security problem, VS body process exists on the target host, and its process is well known. A user model contains a description of the mission, the user (including user's history, current task, preferences, etc.), administrative and control data, etc. The user model, or if appropriate, part of the user model is initialized by the body process on the remote host to continue its mission. When a user needs the virtual secretary on a mission, the propagation will take place through the copying of the user model or part of the user model. This approach implies that no executable code is transferred.

Wen Cao, Cheng-Gang Bian, Gunnar Hartvigsen (1997) presented VS in two aspects: (1) to construct individual agents on three intelligence levels: knowledge-base level, learning level and cooperation level; and (2) to model other agents' activities by task-based for the purpose of efficient cooperation. The cooperation process finds out who can help VS as quickly as possible. The task-based provides the direct mapping between tasks and the related agent sets that could perform such tasks.

3.3 *User preferences*

[Sen 1997] proposed a tradeoff between conflicting user preferences to produce a compromise solution using voting theory. His meeting scheduling uses the calendar manager software to manipulate the user's calendar, and uses the e-mail system to communicate messages with other meeting-scheduling agents.

In contrast to most of the currently available software for centralized calendar management and meeting scheduling, Sen's approach is distributed, where each employee in the organization is provided with an automated meeting scheduler agent. When a user wants to schedule a meeting with other users, the user requests to the agents of the meeting.

Sen proposed a voting mechanism to schedule meetings in order to satisfy user's preferences. Users also can provide preferences for meeting topics, lengths, time, date, etc. Finally the user should rate each preference dimension. By default, VS will assume a preferred profile. The user may, however, specify any alternate profile or VS may learn from observed behavior of the user.

The user could assign a value between 0 and 1 for all options of each dimension. The total of all options under a given dimension need not sum to 1. For example, if the user specifies the preference for meeting on days of the week, Monday could be .56 and Tuesday could be .87 and so forth. The important point is that every participant should be able to specify the weight for the options in each preference. The user also specifies a

minimum threshold for each dimension. If the value of the option falls below the minimum threshold, then the user would prefer not to attend that meeting.

3.4 *The Applications*

3.4.1 Wildfire

Wildfire is a personal assistant developed by Wildfire Communications (www.wildfire.com), which uses speech recognition to manage the users' phone, fax and email communications. Wildfire offers the following services:

- Answer phone and even recognize frequent callers.
- Let users voice-dial their calls.
- When a person leaves a voice mail message, it captures their name and number, so the user can return calls by simply saying "Give me a call!"
- Announce every caller's name so the users always know who is on the line, even if the user is listening to a message.
- Remember names and phone numbers for 150 contacts, and the user can call them by simply saying their name.
- Route the users' incoming calls to any phone that the users designate.

Although Wildfire can make a call to all invitees, it doesn't support meeting scheduling which honors the users' preferences and priorities.

3.4.2 Portico

Portico is a new kind of virtual assistant developed by General Magic (www.generalmagic.com). It can take messages, screen calls, track the users down when they are out of the office, check emails, even get the user the latest business and stock quotes. With Portico the users can prioritize email messages, access their email, voice mail, address book, calendar, news, and stock quotes over the phone or the web and reply to messages right then and there.

MagicTalk™ is the foundation technology for Portico. Portico will understand more than one million different phrases, and can reply to the user with approximately 5,000 responses and helpful hints.

Although Portico is a strong virtual assistant, it doesn't provide a service to users by finding common time slots and locations for scheduling purpose.

3.4.3 Microsoft Schedule +

Microsoft Schedule + is an organizer developed by Microsoft (<http://channels.microsoft.com/scheduleplus>) that helps users keep track of their schedule and contacts. With Schedule+, The users can:

1. Track appointments.
2. Schedule meetings with other Microsoft Exchange Client users.
3. Prioritize tasks.

4. Organize business and personal contacts.
5. Set reminders to help you remember your appointments, meetings, and tasks.

Although Microsoft has a nice interface as compared to our VS and provides its users with recurring meetings, the users are not able to prioritize their free time slots, location and attendees.

3.4.4 Meeting Maker XP

Meeting Maker is an automated meeting scheduler developed by ON Technology (<http://www.inform.umd.edu/ARHU/mm/mmtoc.html>) that helps users schedule a meeting. In contrast to our VS, the users can propose one time slot and location like Microsoft Schedule +. Meeting Maker and Microsoft Schedule + enable users to propose recurring meeting. The users can select the frequency option to propose weekly, monthly or daily recurring meetings. Both software also provide users with calendar where the users organize their tasks and calendar availability.

The major difference between our implementation and other work on meeting scheduling is that we are interested more in efficiently automating the scheduling process than in learning about user preferences over time and minimizing the supervision of the users. Also, anyone can use VS.

4 User and Task Model

4.1 User

User modeling has been found to enhance the effectiveness and/or usability of software systems in a wide variety of situations. A user model is an explicit representation of properties of a particular user. A system that constructs and consults user models can adapt diverse aspects of its performance to individual users. Techniques for user modeling have been developed and evaluated by researchers in a number of fields, including artificial intelligence, education, psychology, linguistics, human-computer interaction, and information science.

User modeling may be used for several purposes: to improve the interpretation of user actions at the interface level, to improve the interface presented to the user or to improve the actions of a system that operates on behalf of the user.

The major emphasis in our work is on user and task analysis. User and task analysis is the process of constructing the user and task model and using these models when decision-making. The analysis includes:

- What users' goals are; what they are trying to achieve?
- What do users actually do to achieve those goals?
- What personal, social, and cultural characteristics the users bring to the tasks?
- How are the users influenced by their physical environment?

- How users' previous knowledge and experience influence how they think about their task

- What do users value most ?

Once we have developed a preliminary picture of the users, we begin to plan how to obtain information about users. We studied the users of VS in three categories:

- How do they define themselves (Personal characteristic)?

- How do they differ (Hosts or invitees)?

- How do they interact with VS?

- How the information VS gains about the users.

The fact is that no matter where the complexity of users' personal characteristics may lead the design decisions, we recognized that the more data we have to assist in decision making, the more successful the decisions are likely to be. Our user and task analysis relies on data obtained from user or task model rather than assumptions. The data is constructed by the users when:

- Request a meeting

- Attend a meeting

- Prioritize the invitees

4.2 Task

We need not only know about the users, but also about users' task. Initially, we tried to find out the ways to simplify what users do so that they can accomplish their goals easily and how the tasks that one person (like host) does relate to tasks that others (like

attendees) do to accomplish a given piece of work (such as meeting scheduling). To answer these questions we need task analysis and building a task model.

In our work, we predicted that the list of tasks may change. The procedures they take to do those tasks may change. This prediction enables VS to add more tasks so that our implementations are much less likely to change. VS already supports the following tasks:

- Request a meeting
- Cancel a meeting
- Get the meeting status
- Attend a meeting

4.2.1 Users' Goals

To do a task analysis, we should understand users' goals and how users move from goals to tasks to actions. A task is what someone does to achieve a goal.

Our users' goal is to schedule the meeting according to the users' calendar availability. Since we have different users with different preferences, there is no guarantee that all the users' goal match.

4.3 Modeling

To organize and analyze the user data and build the user model, we need an effective interface. The model that we build will depend on the type of information we have collected from users through interface, the tasks and the users' goals. We take this information into account as we develop our user or task model.

Modeling is a complex process that requires both creativity and a firmly grounded understanding of users, tasks, and environments. In modeling, we have to figure out:

- How to make sense of all data retrieved from the users.
- How to turn the data into useful information such that VS communicates effectively to its other components.
- How to turn the communications into decision making

4.3.1 Metaphors for the interface design

Metaphors provide analogs from the users' real world to the virtual world you have constructed in the interface. In our VS, we designed a straightforward visual representation of the users' current process. For example, if users are requesting a meeting, we constructed a form in the interface that helps users express their request as they communicate by phone.

4.3.2 Use sequences

A use scenario tells a story about the users and their proposed interactions with VS. With a use sequence, you take part of a scenario and turn it into a sequence of steps. The steps should clearly indicate what actions the user will perform, what decisions the user must make, and what actions the system will perform for the user.

After designing the interface, we have to make sure that the interface:

- conveys the user model.
- provides messages where and when the user needs them.
- streamlines tasks for the user.
- works for all the scenarios.
- covers the tasks that users expect to be able to do with VS.

4.3.3 User model

A user model is a knowledge source, which contains explicit assumptions on all aspects of the user. A user modeling component is that part of a dialog system whose function is to incrementally construct a user model; to store, update and delete entries; to maintain the consistency of the model; and to supply other components of the system with assumptions about the user.

Every user is assigned a user model representing the user's knowledge. The user model can be initialized by either the user's input or stereotypes. the user model can be manipulated by the user or the system while interacting with the user. VS could handle stereotypical knowledge about users. It could make use of two stereotypes: Host and Invitee.

The user's input is the most reliable in problem-solving domains. The Human-Computer interface is one of the technique for receiving the user's input in order to construct the initial user model. The initial user model can be improved based on inferring the initial

assumptions about the user and updating the knowledge while the user interacts with the system.

4.3.4 Proposed solution : Knowledge-based CGI systems

Integrating knowledge-based system and CGI could provide more flexible and intelligent navigation. A database can be used to keep and help access the information.

First generation CGI offers only links between the interface and the server side program. Second generation systems can rely on knowledge representation approach to describe relationships between information. The question is how to provide VS with user model which is conveyed and queried by CGI.

The proposed model of the system relies on a four-level organization:

The information level contains data, which are considered as raw information mainly dedicated to VS user. Only the user gives this information to VS. This level can be implemented by human-computer interface.

The knowledge level helps reasoning. It contains general knowledge about the task domain (useful concepts, their relationship and task model). It is implemented as knowledge representation system.

The mapping level defines a mapping between data of the information level and concepts of the knowledge level.

The communication level defines the pipelining of an intelligent preprocessing and a database. This can be implemented using CGI. Explicit query can be sent to the mapping level.

In this chapter, we discussed what factors should be taken into consideration in order to make the user and task model by designing the interface. The next chapter will explain in detail how we used them in order to make our initial user or task model.

5 Design

5.1 *Meeting Scheduling Automation*

When a user requests a meeting to be scheduled, participants have preferences about when they like to meet such as time of day, day of week, position of invitees, topic of the meeting, etc. The Virtual Secretary should balance such concerns, proposing and accepting meeting times that satisfy as many of these criteria as possible. For example, a user might prefer not to meet at supertime unless the president of the company is hosting the meeting.

Scheduling a meeting by real secretaries takes a lot of time and effort, and there is no guarantee that they find a free common time slot and location that satisfy the invitees' preferences. In the worst case, there is no time slot that satisfies the preferences of all the invitees, and the secretary should ignore that time slot or location, and the host of the meeting should try to schedule the meeting in another time slot or location. This process will continue until the scheduling succeeds.

The other important issue in scheduling is the prioritization. In each meeting, there are some participants whose presence is mandatory for the meeting. The meeting can proceed in the absence of some of the non-mandatory or low priority participants. This fact would relax the constraint of finding common time slot or location to finding a “fairly” common time slot or location.

5.2 Information needed for cooperation

Generally, a meeting is specified by:

1. The set of participants, their e-mail addresses, their positions and places of work (attendee's profile).
2. The host of the meeting and the host's e-mail address (host's profile).
3. The meeting identification (which is used as a password).
4. The length of the meeting.
5. The possible starting times (date and time) on the calendar for a meeting. The current version limits the number of starting times to six.
6. The attendee's preferences for proposed time slots when confirming.
7. The attendee's preferences for proposed locations when confirming.
8. The priority of time slots (date and time) and locations for the host of the meeting.
9. The priority of invitees (non-mandatory, low, high, mandatory).
10. The scheduling deadline (date and time): This is the exact time that VS start off-line processing in order to schedule a meeting.
11. The subject of a meeting.
12. The time of request by the host of a meeting (Dynamically generated by VS).
13. The time of confirmation by each attendee (Dynamically generated by VS).

5.3 Scheduling Algorithm

The following algorithm explains how VS carries out meeting scheduling:

1. Initially, the host of the meeting starts requesting a meeting by pressing the Request-meeting button on the main menu of VS which is currently located at: (<http://www.cs.concordia/~grad/salamat/cgi-bin/menu.html>). This page is password protected by the system administrator. Therefore, all potential attendees must be given this URL and its corresponding password.
2. An HTML form will be opened which will enable the host to request for a meeting by entering:
 - The host' name.
 - The host's e-mail address.
 - The meeting ID which is used as a password and meeting identification for each individual meeting.
 - The date and time of request. The application provides users with the current day and time dynamically.
 - The time slots proposal: An array of possible date and times with top-down priority, meaning that if the application find more than one time slot, the host preference is the one which is situated higher in the list box.
 - Deadline (date and time).
 - Location(s): The possible locations.
 - Meeting subject.
 - The e-mail addresses of attendees (mandatory and non-mandatory).

3. The host requests a meeting by pressing the OK button and the program automatically will send an invitation e-mail to the attendees. "You are requested to attend the meeting ID4024419. Please confirm your attendance at <http://www....>". Simultaneously, the application creates a process for scheduling purpose. This process will remain in sleeping mode till the deadline is arrived.
4. It is worthwhile to mention that at this time VS also creates a profile for the meeting by saving the meeting information in various files. Chapter 5 discusses meeting profile in detail.
5. When invitees receive their invitation mail, they go to VS main menu on the Internet, and click on the "Attending Meeting" button. A pop-up window asks for the meeting ID. Then the invitee will be taken to another URL, which contains all information about the pending meeting. The invitees should click on all of the intervals (time slots) they can attend. Then, the invitee notifies VS of the request by pressing the OK button. It also triggers VS to update the meeting profiles by saving the information to the files.
6. The meeting should be scheduled by the time the deadline is expired. The process will be woken up and VS tries to find a common time slot. If it cannot find any interval, it fails and the meeting is abandoned by informing the host of the scheduling failure. Otherwise it schedules the meeting for the earliest interval (which has highest priority).

5.4 *Preferences*

Users have preferences on when they like to meet, e.g. time of day, day of week, status of other invitees, location, etc. An automated meeting scheduling system should be able to balance such concerns. In our work, the preferences target the invitees' preferences over the time slots and locations offered by the host of meeting. Since the hosts of meeting can offer more than one time slot and one location, the invitees can choose as many locations and time slots as they wish. Our VS allows only the host to offer the time slots and locations.

The user's preference ranking is an integral part of the decision making of VS. VS arrives at consensus for meeting time and locations while balancing different preferences.

5.5 *Priorities*

To be useful, an automated meeting scheduling system has to be adaptive to user priorities. The fact is that VS is to be used by humans, therefore, it is essential to encode and follow the priorities and preferences of associated users. This section gives an overview of our prioritization mechanism that reasons with user priorities in order to schedule a meeting in a manner that will satisfy the user. This section will also illustrate how the prioritization scheme produces a compromise solution involving tradeoff between conflicting user preferences.

When it comes to priority, we target the invitee's priorities from the point of view of the host of the meeting. The host can assign points to each invitee. For example, Invitee_1 has 10 points (low priority), Invitee_2 has 20 points (high priority), Invitee_3 has 100 points (Mandatory), and Invitee_4 has 0 points (Non-Mandatory). Hosts are not obligated to choose priority and in this case, VS assumes that all members are mandatory, therefore it finds a mutual time slot and location that is common between all the invitees.

Hosts have the choice to prioritize their invitees. They can do this task at any time between requesting a meeting and the deadline for the scheduling.

VS weighs the invitee's points given by the host of the meeting. Using these priority values, the scheduler generates a ranking for each time slot and location. VS provides the host with default priority values.

5.6 Quorum requirements

The host can also specify a minimum threshold. If the result of ranking falls below the minimum criteria, then VS won't announce that meeting for that specific time slot and location. Since VS is responsible for rating each time slot and location, the scheduler will choose the time slot and location corresponding to the maximum value above the minimum threshold.

To calculate the priority values relating to a specific time slot or location, the scheduler retrieves the files where the information about the user preferences and priorities has been saved. Then, for each time slot, the scheduler checks if the invitee has chosen that interval. If so, the scheduler accumulates the invitee's score. The scheduler uses the same procedure for each location. At the end, we have a two-dimensional table, with invitees on one dimension, and locations and time slots on the other. This table represents the sum of points that each time slot and location has been accumulated so far. Then, the scheduler finds the time slot and location, which have maximum accumulated points. If the two values corresponding to the time slot and location are greater than minimum threshold chosen by the host, and the number of attendees whose preferences satisfy this time slot and location is greater than minimum number chosen by the host, the meeting will be scheduled. The scheduled meeting will be announced to all of the invitees by e-mail. Even the attendees can be informed of the current status of the meeting through Internet. There is a web page designed for this purpose to give the status of the meeting at any time. This URL also represents that how many invitees have confirmed their attendance, and which time slots and locations they have chosen.

6 Implementation

6.1 *The programming language*

The system was implemented in a platform independent language (Perl). Perl was chosen as the programming language due to its strong text processing capabilities. Perl has readily adapted itself to the task of providing information using text-based protocols. The Web is driven by plain text. Web servers and web browsers communicate using a text protocol called HTTP, HyperText Transfer Protocol. Some parts of the program also are encoded in a text markup system called HTML, HyperText Markup Language. This grounding in text is the source of much of the Web's flexibility, power, and success.

Perl is by far the most widely used language for CGI programming. It contains many other powerful features. The advantages of Perl include:

- ❑ It is highly portable and readily available.
- ❑ It contains very simple and concise constructs.
- ❑ It makes calling shell commands very easy, and provides some useful equivalents of certain UNIX system functions

6.2 *Common Gateway Interface*

The Web is a client-server system. Client browsers request documents identified by Uniform Resource Locator (URL) from web servers. This browser-to-server dialog is governed by the HTTP protocol. Most of the time, the server merely sends back the contents of a file. Sometimes, however, the web server will run another program to send back a document that could be HTML text, an image, or any other document type. The server-to-program dialog is governed by the CGI (Common Gateway Interface) protocol, so the program that the server runs is a CGI program or CGI script. The server encodes the client's form input data, and the CGI program decodes the form and generates output.

The server tells the CGI program what page was requested, what values (if any) came in through HTML forms, and where the request came from. The CGI program's reply has two parts: headers to say "I am sending back an HTML document", "I am sending back a GIF image", or "I am not sending you anything, go to this page instead", and a document body, perhaps containing GIF image data, plain text, or HTML. The full CGI specification lays out which environment variable holds which data (such as form input parameters) and how it is encoded.

The CGI programs are called each time the web server needs a dynamic document generated. The CGI program doesn't run continuously, with the browser calling different parts of the program. Each request for a partial URL corresponding to the CGI program starts a new copy of the CGI program. The CGI program generates a page for that request, then quits.

A browser can request a document in a number of ways called methods. The GET method simply requests a document. The HEAD method is used when the browser wants to know about the document without actually fetching it. The POST method is used to submit form values. In our VS, we have used the POST method rather than the GET method. The reason is that POST requests cannot be cached, because each request is independent. The browser or the server may cache a GET request in the URL. This means that making a GET request for a particular URL once or multiple times should be no different. That is why the HTTP GET method is used in document retrievals where an identical request will produce an identical result, such as dictionary lookup. GET method also has limitation on the total size of the data requested. The HTTP POST method sends form data separate from the request. It has no such size limitation.

6.3 *Platform*

Our VS is implemented on Unix Solaris. Since the program is written in Perl language, VS can be implemented on PC, as well. The only restriction for the users is access to the Internet. It doesn't matter where and when they interact with VS.

6.4 Architecture

The architecture of the automated meeting scheduler is illustrated in Figure 2.

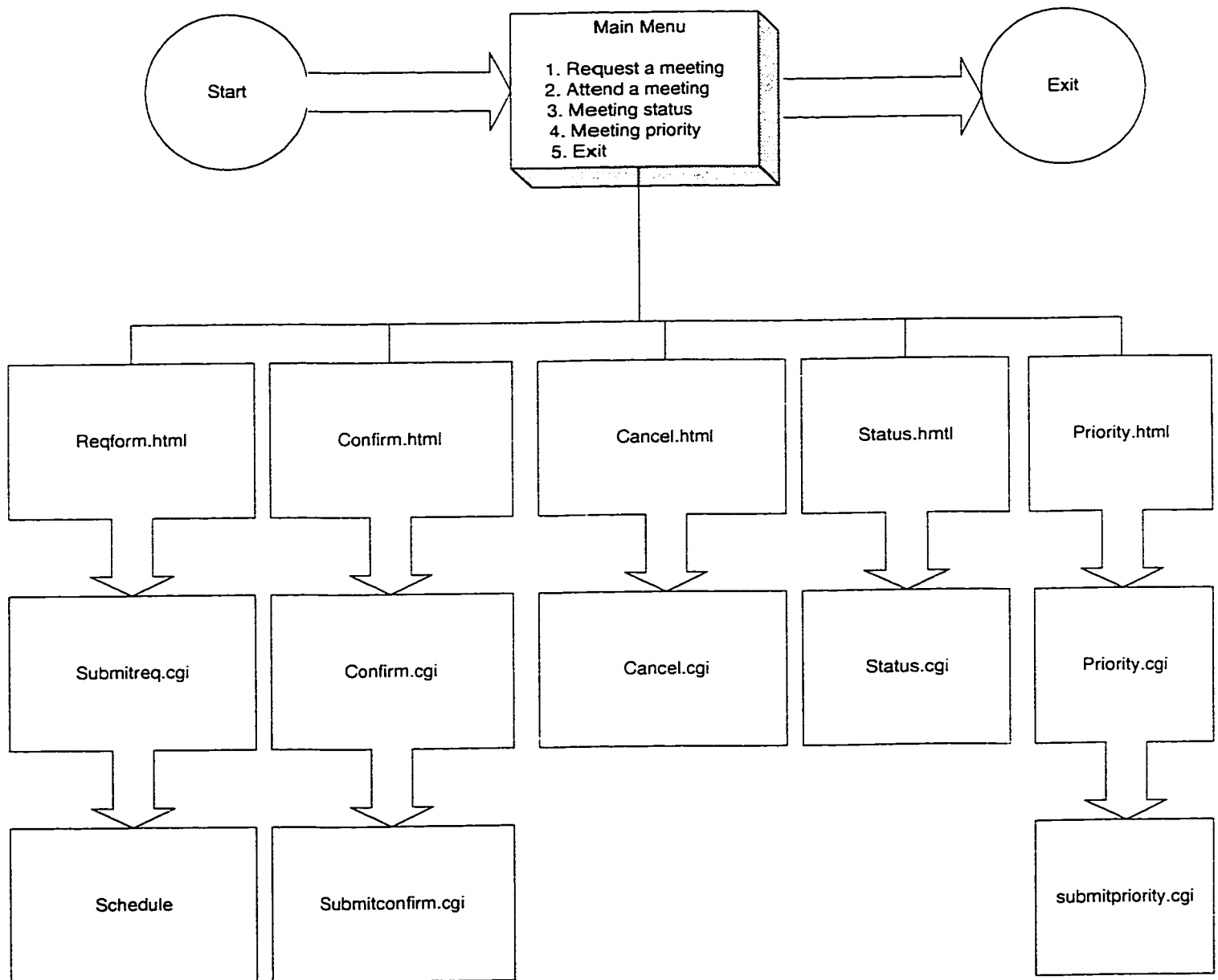


Figure 2

6.5 *The Modules*

To develop a system that is portable across the platforms, we developed an HTML-based GUI for user-VS interaction. The following sections explain each module used in VS application. The modules with extension (.html) are HTML scripts (HTML tags) and the (.cgi) are CGI scripts written by Perl.

6.5.1 Menu.html

User starts interaction with the meeting scheduler through the main menu. The main menu is the front page of VS. This interface allows users to request for a meeting, cancel a meeting, view a meeting status, attend in a meeting, prioritize invitees and exit. Figure 3 illustrates the main menu.

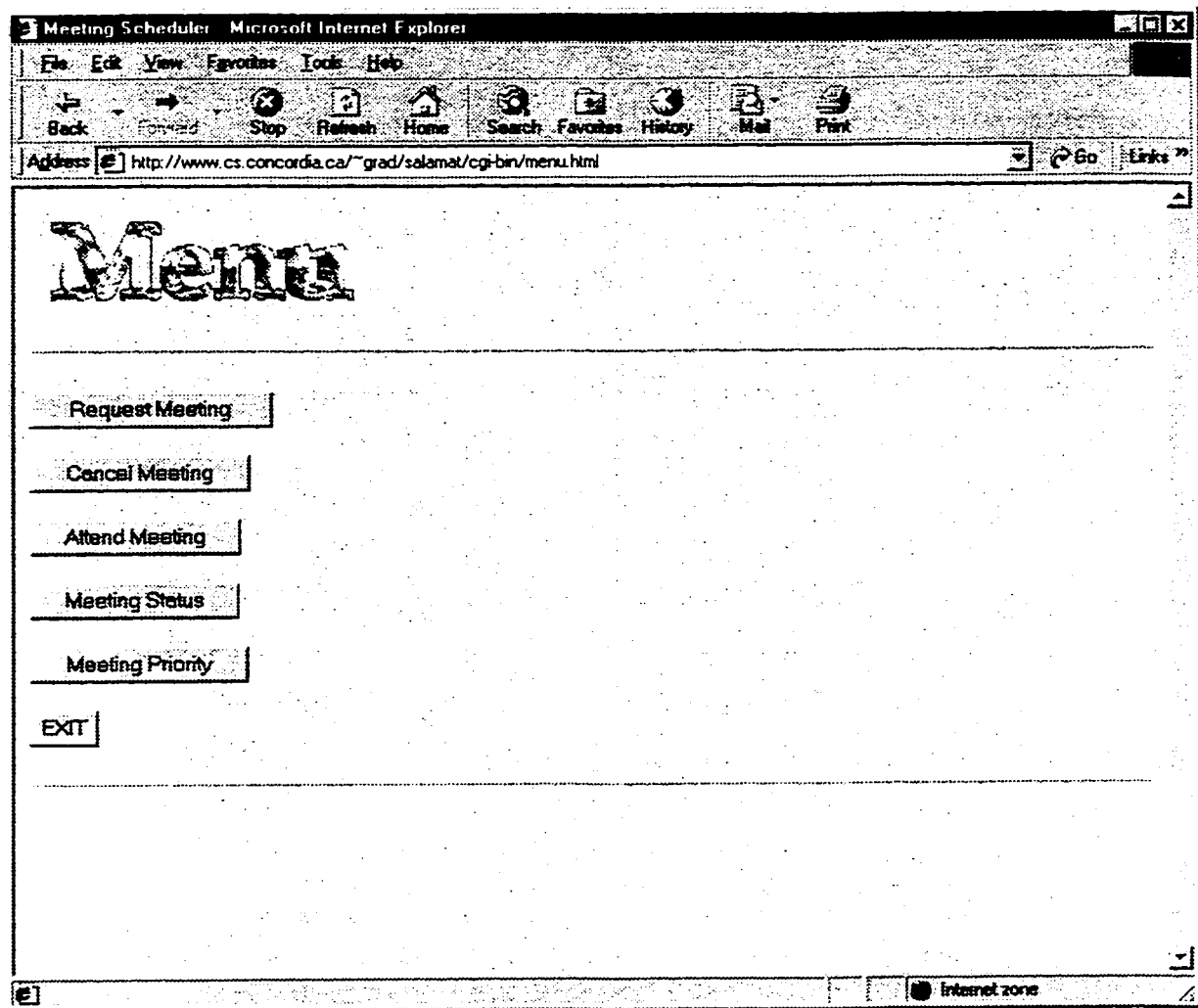


Figure 3

6.5.2 Reqform.html

The user can request a meeting by pressing the request-meeting button on the main menu. The reqform.html module creates a form, using HTML tags, where the host of the meeting can fill in all of the information related to a meeting that will be submitted to the server. The forms are composed of widgets, like text entry fields, check boxes, list boxes, text areas and radio buttons.

The request form contains:

- Text fields for the host name, the e-mail address, subject and length of a meeting.
- A text field for a meeting ID, which is a unique key to process the information of a meeting. The type of this text field is as password.
- The current time is represented dynamically as the user arrives at this page. Therefore, there is no need for the user to provide this text field manually.
- A text area for location(s) and a text area for invitees' e-mail addresses.
- A text field for deadline time and three list boxes for deadline date (day, month and year). VS starts scheduling off-line using deadline.
- One text field for time and three list boxes for date (day, month and year) for all six time slots. Time slots are the intervals that the host of the meeting offers to invitees. Each time slot is labeled using a check box. In this sense, the user should mark the corresponding check box if that time slot is to be offered to invitees.

After the host fills out the form completely, the host should press the "submit" button in order to submit the request to the "submitreq.cgi" module. The host can also clear the fields by pressing the reset button. The user can always go back to the main menu. The user will be forced to enter all the required information in the form if the user happens to forget to enter some of the items. In the case of time slots and locations, at least one item is required. Figure 4 illustrates the form used by the host to requests a meeting.

Request a new meeting Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print

Address http://www.cs.concordia.ca/~grad/salamat/cgi-bin/reqform.html? Go Links

Request a new Meeting

Host Name Hooman Salamat Email salamathooman@mpact.net

Meeting ID Meeting Subject Action plan for the project

Time of Request Thu Jan 14 10:28:22 EST 1999

Length of meeting (minutes) 120

Deadline

Time(hh:mm) 22:59 March 20 1999

Location(s) 1455 De Maisonneuve #1043-1, Concordia Uni.

Attendee's Email Addresses

Grogono@cs.concordia.ca
Vsalamat@alpn.net
Christopher.cottin@statcan.ca
Salamat@cs.concordia.ca

Possible Time Slots
Top-Down Priority

1	<input checked="" type="checkbox"/>	Time(hh:mm)	10:00	January	18	1999
2	<input checked="" type="checkbox"/>	Time(hh:mm)	16:30	January	20	1999
3	<input checked="" type="checkbox"/>	Time(hh:mm)	14:59	February	9	1999
4	<input checked="" type="checkbox"/>	Time(hh:mm)	11:25	February	17	1999
5	<input checked="" type="checkbox"/>	Time(hh:mm)	10:00	February	3	1999
6	<input type="checkbox"/>	Time(hh:mm)	00:00	March	Day	Year

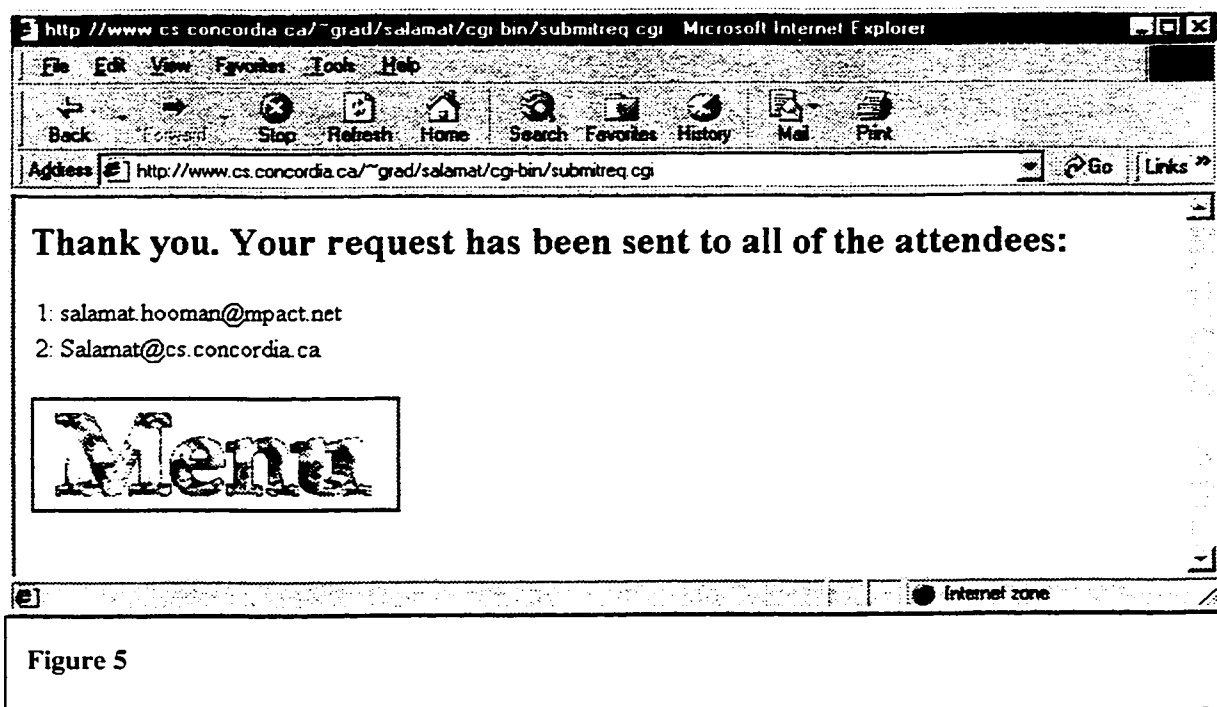
Submit Request Reset

Internet zone

Figure 4

6.5.3 Submitreq.cgi

The Submitreq.cgi ensures whether or not the form is filled out completely. If not, it prompts the user which field has not been filled yet. Then it checks out if there is another meeting with this meeting ID. If so, it forces the user to change the meeting ID. Figure 5 illustrates the HTML page that the Submitreq.cgi returns.



6.5.3.1 Storing meeting profile

After the above validations, this module creates six files. If the meeting-id is meeting-file, then those files will be: "meeting-file.pro", "meeting-file.sch", "meeting-file.mm", "meeting-file.loc", "meeting-file.tim", and "meeting-file.sloc". For keeping information of the initial user and task model, we used the file system rather than the database. The reasons are:

- The users should have access only to the information related to their own meeting. The system takes a risk by allowing users to update the database through CGI script.

- ❑ As the volume of the meeting increases, the speed of communication decreases due to updating or retrieving information from the database. Since every time the users interact with the system, the user or meeting profile should be retrieved or updated.

6.5.3.1.1 *meeting-file.pro*

The “meeting-file.pro” contains all filtered information passing from reqform.html to “submitreq.cgi” as text format.

6.5.3.1.2 *meeting-file.sch*

In the “meeting-file.sch”, each record (line) has two fields. An e-mail address of an attendee and a time slot code indicating the attendee’s preference in terms of possible time slots that the invitee could attend in a meeting. The time slot code contains at most six digits (012345) corresponding to the maximum six possible time slots, which the attendee can choose at the time of confirmation from the list box. The “meeting-file.sch” is a hash table, which provides the essential information to the scheduler when finding the common time slot.

6.5.3.1.3 *meeting-file.mm*

The “meeting-file.mm” contains the e-mail address of all invitees.

6.5.3.1.4 *meeting-file.tim*

The “meeting-file.tim” contains two fields: the digits (0-5) and possible date and time slot. This is a kind of array, which maps the time slots to the digits, which generate the time slot code in the “meeting-file.sch” file.

6.5.3.1.5 *meeting-file.loc*

The “meeting-file.loc” contains two fields: the digits (0-5) and locations. This is a kind of array, which relates the locations to the digits, which generate the location codes using in the “meeting-file.sloc” file.

6.5.3.1.6 *Meeting-file.sloc*

In the “meeting-file.sloc”, each record (line) has two fields. An e-mail address of an attendee and a location code indicating the attendee’s preference in terms of possible locations that the invitee could attend in a meeting. The location code contains at most six digits (012345) corresponding to the maximum six possible locations, which the attendee can choose at the time of confirmation from the list box. The “meeting-file.sloc” is a hash table, which provides the essential information to the scheduler when finding the common location. It is worthwhile to mention that if only one location is offered by the host, VS doesn’t use this file.

The “submitreq.cgi” basically extracts the information for later retrieval and scheduling purposes, and distributes it in various files. Then it sends an invitation to each attendee via e-mail. This mail contains the name of the host, meeting ID, which is essential for the

confirmation. It also asks the attendees to confirm their attendance at the host site. At the end, it runs the “schedule” module as off-line.

6.5.4 Confirm.html

The Confirm.html script receives the meeting ID as a password and an identification to retrieve the required information related to the meeting and passes it to the “Confirm.cgi” module. All the invitees start with this web page to confirm their attendance and express their preferences regarding time, dates and locations. Figure 6 illustrates this HTML page.

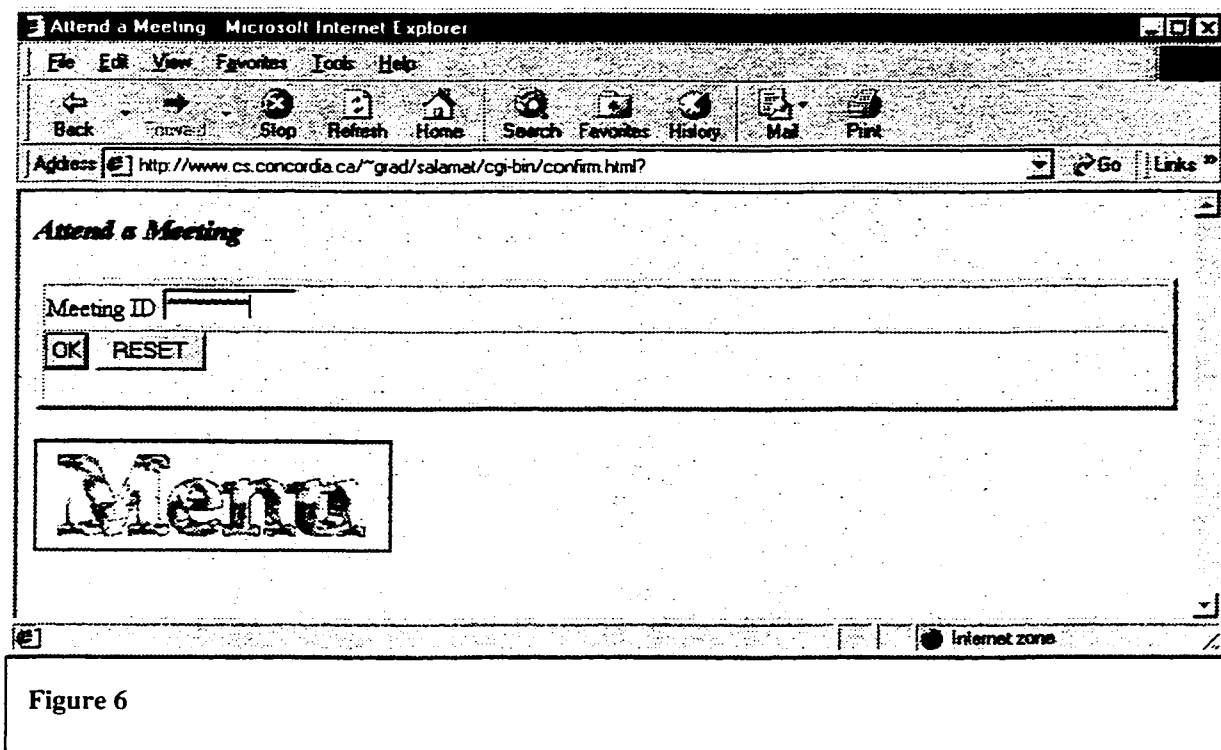


Figure 6

6.5.5 Confirm.cgi

The Confirm.cgi retrieves the user profiles from the files "meeting-file.pro", "meeting-file.loc", and "meeting-file.tim". The invitee enters the name in a text field on the form. The invitee also chooses its own e-mail address, the time intervals and locations that can attend from list boxes on the interface. The time of confirmation is represented dynamically as time at which the user arrived at this page. The time slots and locations are represented as multiple selection list boxes. Acceptable time slots and locations can be specified by clicking on the items. Reset and Submit buttons are provided to clear all entered information, and to submit the meeting request. Figure 7 illustrates the form that the invitees fill out in order to send their proposals to VS.

Attend a meeting Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print

Address <http://www.cs.concordia.ca/~grad/salamat/cgi-bin/confirm.cgi> Go Links

Attend in the meeting :

Attendee's Name Email address


Time of Confirmation

Meeting Subject

Length of meeting (minutes) DeadLine

Please choose all of the time slots that you can attend :

Please choose all of the locations where you can participate :



Internet zone

Figure 7

6.5.6 Submitconfirm.cgi

The Submitconfirm.cgi first receives the information from confirm.cgi. The information basically contains the time intervals and locations that the invitees prefer to attend at a meeting. This module codifies the invitee's preferences. Then it updates the "meeting-file.sch" and "meeting-file.sloc" files by entering the attendee's e-mail address and the time slot and location codes. In this module, the invitee would be forced to choose at least one location and one time slot. Figure 8 illustrates the HTML page that the CGI script returns.

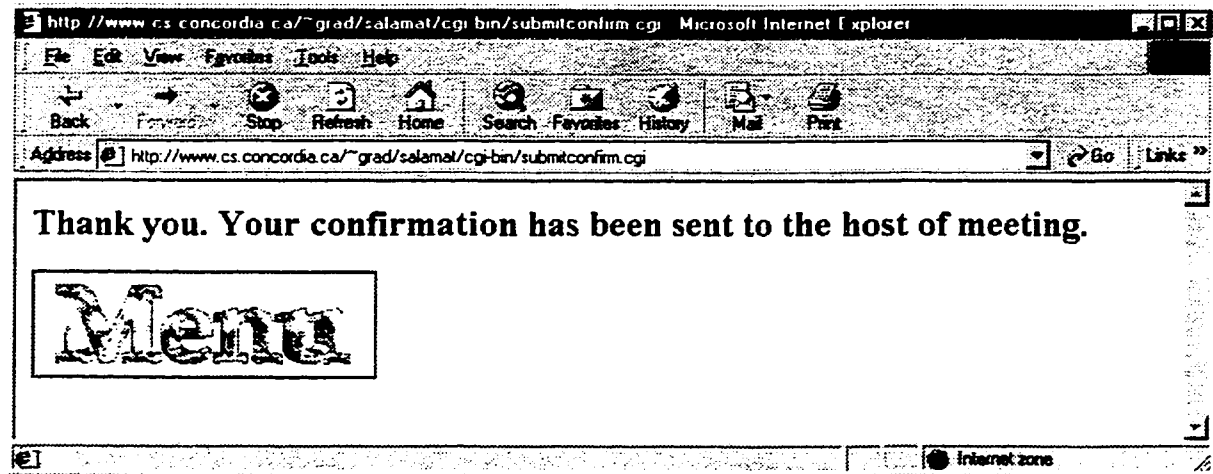


Figure 8

6.5.7 Status.html

The Status.html script receives the meeting ID as a password and a meeting identification to the meeting and passes it to the "status.cgi" module. The host of the meeting can get the status of a meeting at any time. Figure 9 illustrates this HTML page.

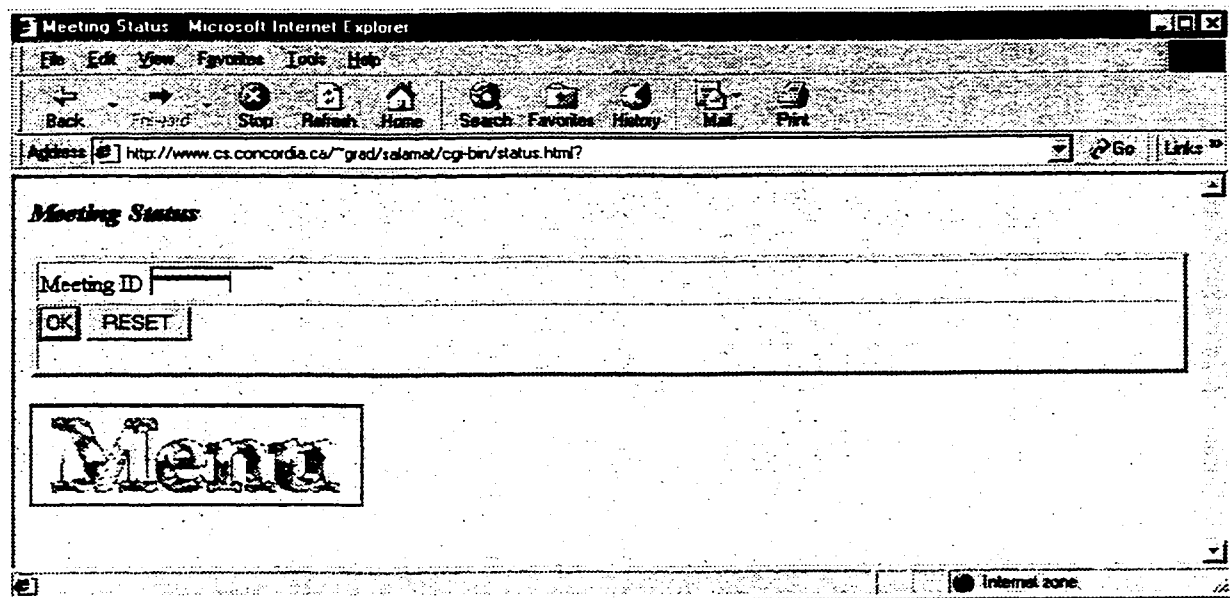


Figure 9

6.5.8 Status.cgi

The Status.cgi script presents the status of the meeting at any time. It basically contains two tables saying which locations and time slots has been chosen by each invitee so far. This module also says when and where the meeting has been scheduled. This is useful information for the invitees that they want to get information about their meeting without e-mails. It also helps for the host of meetings to make a better decision. In particular, when it comes to prioritizing the invitee, this would be helpful. Figure 10 illustrates that this HTML page returns.

Meeting Status: Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print

Address http://www.cs.concordia.ca/~grad/salamat/cgi-bin/status.cgi

Go Links

Meeting Status

The Locations has been chosen by each invitee so far

salamat1@cs.concordia.ca	2200 Ward st. Apt#503	1455 Blvd. Rene Levesque #1043-1	1155 Blvd. Rene Levesque Suite# 2200
salamat@cs.concordia.ca	2200 Ward st. Apt#503	1155 Blvd. Rene Levesque Suite# 2200	
salamat.hooman@mpact.net	1455 Blvd. Rene Levesque #1043-1	1155 Blvd. Rene Levesque Suite# 2200	
salamat@cs.concordia.ca	1455 Blvd. Rene Levesque #1043-1		

The Time Slots chosen by each invitee so far

salamat1@cs.concordia.ca	Wed Jan 13 10:00:00 1999	Wed Jan 13 12:30:00 1999	Sun Feb 14 13:55:00 1999	Mon Feb 15 20:00:00 1999	Mon Mar 1 10:00:00 1999	Mon Apr 5 08:15:00 1999
salamat@cs.concordia.ca	Wed Jan 13 10:00:00 1999	Sun Feb 14 13:55:00 1999	Mon Mar 1 10:00:00 1999			
salamat.hooman@mpact.net	Wed Jan 13 12:30:00 1999	Sun Feb 14 13:55:00 1999	Mon Mar 1 10:00:00 1999			
salamat@cs.concordia.ca	Wed Jan 13 10:00:00 1999					

Internet zone

Figure 10

6.5.9 Priority.html

The hosts of meeting can prioritize their attendees by arriving at the “Priority.html” page by pushing the “Priority” push button on the main menu of VS. This feature is optional for the hosts of meeting. This HTML page asks the host for the meeting id and password and passes them to the Priority.cgi. The next section explains in detail how the Priority.cgi accomplishes the task of the invitees’ prioritization. Figure 11 illustrates this HTML page.

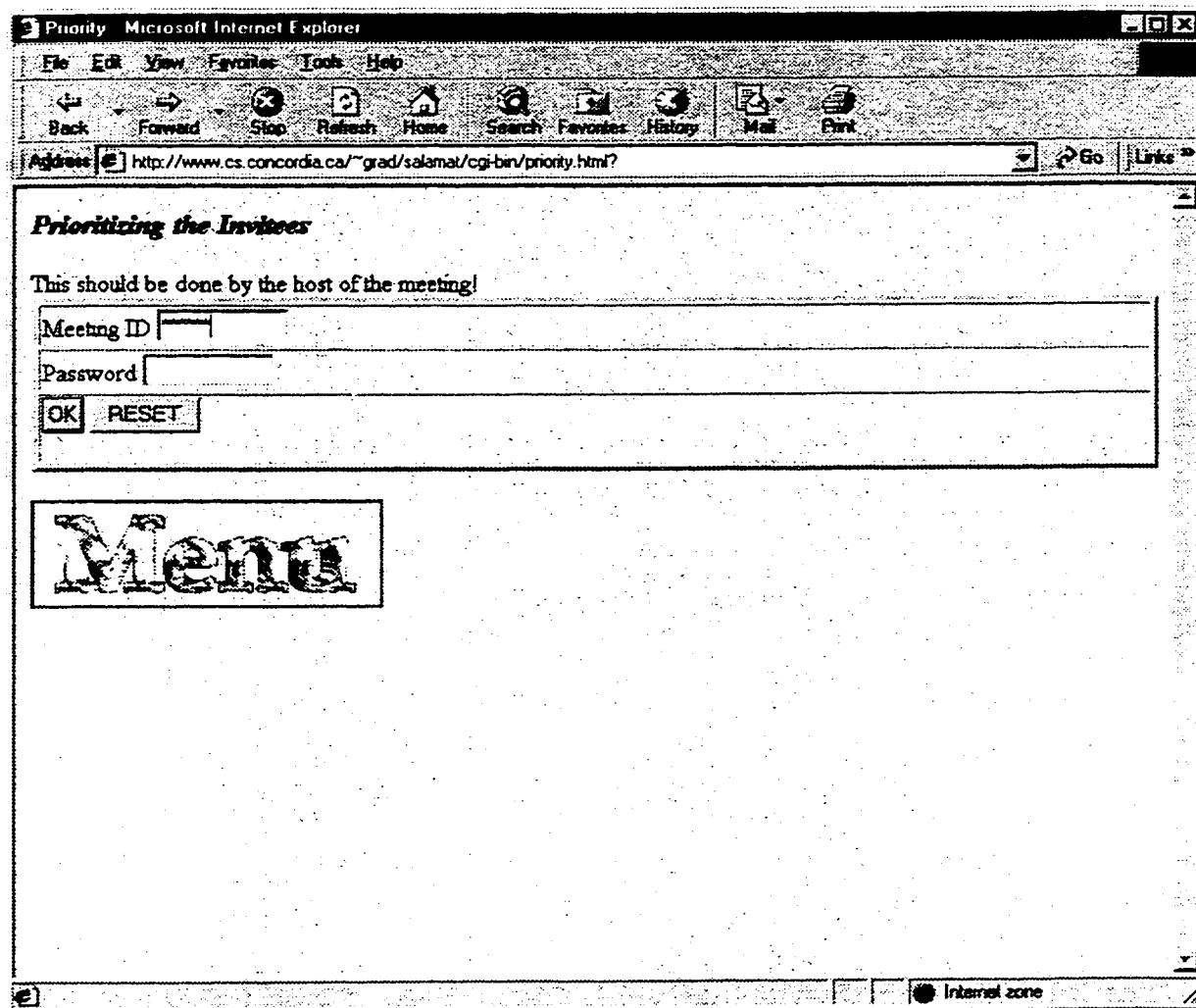


Figure 11

6.5.10 Priority.cgi

The host of the meeting prioritize the invitees. Basically VS provide four categories for invitees.

1. Non-mandatory: Those invitees whose presence doesn't affect the scheduling at all.
2. Low: Those invitees whose presence wouldn't really affect the scheduling.

3. High: Those invitees whose presence is important, but the meeting could be scheduled in their absence.
4. Mandatory: Those invitees whose presence is mandatory, otherwise the meeting won't be scheduled. Typical mandatory invitees include chair, secretary, vice-president, etc.

The host can also choose the minimum number of attendees required in order to scheduling a meeting.

Each of four categories mentioned above has its corresponding score. Non-mandatory is 0, Low is 10, High is 20 and Mandatory is 100. The host can also prioritize the invitees by giving the minimum accumulated points required to schedule a meeting. This threshold value is used for making decision on location and time slot according to the invitee's priority, when there is no common time slot and location.

It is important to point out that the prioritization procedure is optional. If a host doesn't ask for prioritization, the scheduler assumes that it should find the absolute common time slot and location. But if the host prioritize the invitees, the scheduler will find the most common time slot and location according to the priorities. Figure 12 illustrates the HTML page that this CGI script returns.

Priority Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print


Address http://www.cs.concordia.ca/~grad/salamat/cgi-bin/priority.cgi Go Links

Prioritizing the Invitees

salamat@cs.concordia.ca	Non-Mandatory <input type="radio"/>	Low <input type="radio"/>	High <input checked="" type="radio"/>	Mandatory <input type="radio"/>
salamat.hooman@mpact.net	Non-Mandatory <input type="radio"/>	Low <input checked="" type="radio"/>	High <input type="radio"/>	Mandatory <input type="radio"/>
The value of each priority Non-Mandatory = 0, Low = 10, High = 20, Mandatory = 100				

The minimum number of attendees required to schedule this meeting

The minimum points required to schedule this meeting



Internet zone

Figure 12

6.5.11 Submitpriority.cgi

The Submitpriority.cgi module stores the invitees' priority from the host's point of the view to a file ("meeting-file.pri"). This information will be referred during scheduling.

Figure 13 illustrates the HTML page that this CGI script returns.

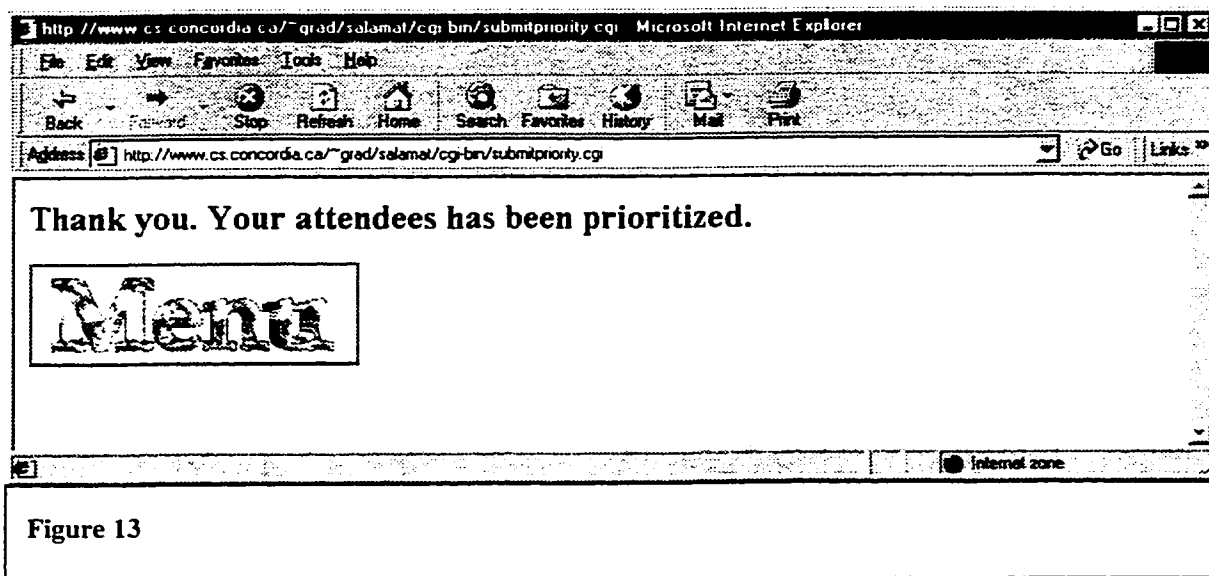


Figure 13

6.5.12 Cancel.html

The Cancel.html receives the meeting ID as a password and an identification to the meeting and passes it to the "Cancel.cgi" module. The host of the meeting can cancel a meeting at any time. Figure 14 illustrates this HTML page.

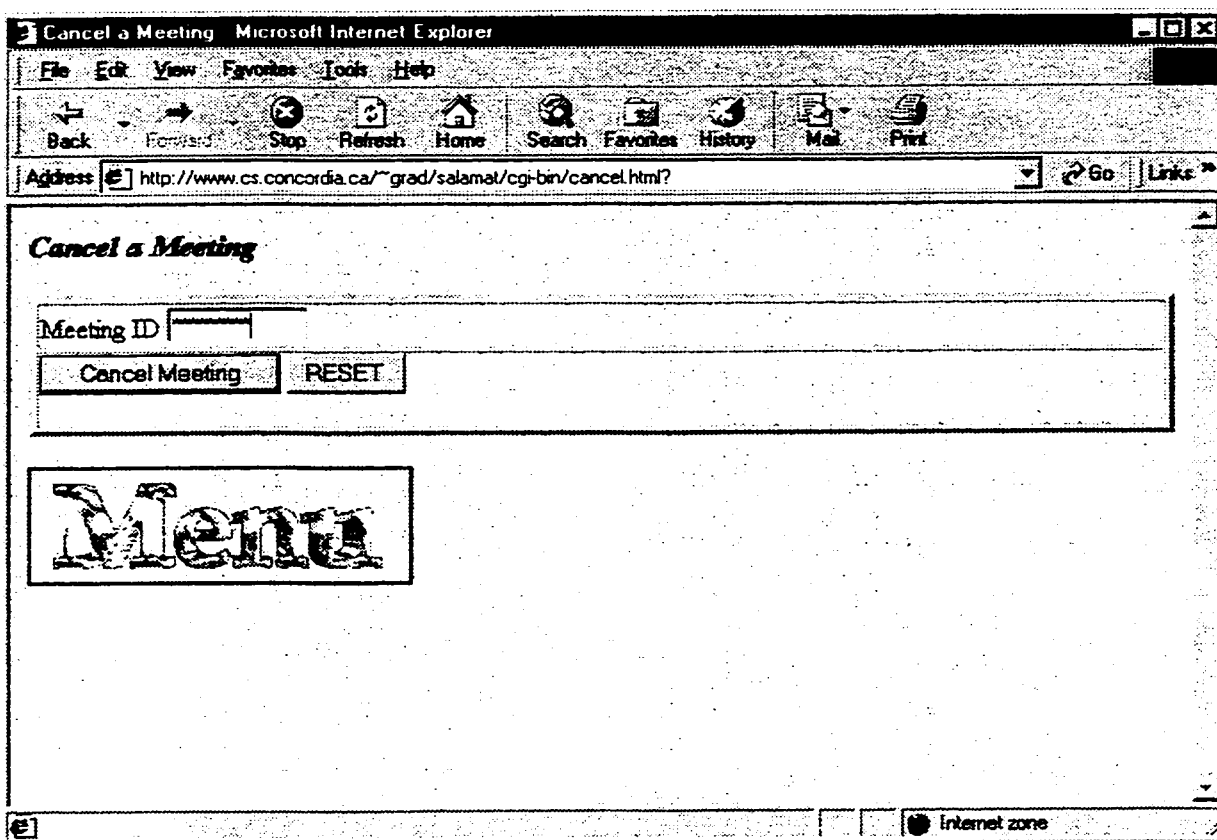


Figure 14

6.5.13 Cancel.cgi

The Cancel.cgi removes all of the files related to the meeting from the file system. In general we have eight files for each meeting (meeting-file.sch, meeting-file.loc, meeting-file.sloc, meeting-file.mm, meeting-file.pro, meeting-file.tim, meeting-file.fin, meeting-file.pri). Figure 15 illustrates the HTML page that this CGI script returns.



Figure 15

6.5.14 Schedule

The Schedule module actually does the whole process of scheduling. When this module is called by "submitreq.cgi" module, its process sleeps until the deadline arrives. The deadline is basically a date and time indicating when VS should wake up the sleeping process and schedules the meeting. By the time the deadline kicks off, the process automatically wakes up as an off-line process on a particular machine. The "schedule" module gets the meeting profile, extracts the time slots offered by the host of the meeting, counts the number of invitees and compares with the number of attendees who have confirmed their attendance so far. If there are enough members (all the invitees who has been invited, do they confirm their attendance) or if there is a common time slot and common location according to their priority, then the meeting will be scheduled. The

schedule module uses the “meeting-file.sch” for finding common time slot and the “meeting-file.sloc” for finding common location (if there is more than one).

Consequently, VS informs all of the attendees of exact date, time and location of a meeting. If there is no common time slot and location, the scheduler checks out if there is a file called “meeting-file.pri”. If not, VS informs the host of the meeting of the failure. If yes, VS goes to the following steps to find the most common time slot according to the host’s prioritization.

1. Find the most common time slot.
2. Count the number of the attendees for this time slot.
3. For each attendee, find its priority (and its corresponding point) and then calculate the total points of all attendees for this time slot.
4. Find the slot time which has the maximum total points.
5. If the number of the attendees is greater than the minimum number and the maximum total score is greater than minimum point required to schedule a meeting, the meeting will be scheduled at the most common time slot, if there is a common location.
6. If there is no common location, then VS calculates the total point for each location.
7. Find the location which has the maximum total point.
8. If the maximum total point for that location is greater than the minimum required score, the meeting will be scheduled, otherwise, VS informs the host of the meeting of the failure.

9. Create a file "meeting-file.fin", which contains the result of the scheduling. The content of this file is used when the users like to get the status of the meeting through the Internet. This is useful, when the attendees don't have access to their mailbox.

6.5.15 E-mail

VS calls the e-mail subroutine in order to invite the attendees or inform the host of the meeting of the cancellation or success of the scheduling. It is important to note that the users never send any mail to anybody in our implementation. The users receive at most two e-mail messages from VS: the invitation, and the report of success or failure of the scheduling.

Our e-mail subroutine uses the Unix Mini-mailer utility for handling mails through the Internet. When it comes to WWW, the mailing strategy is different from standalone programs for security purposes. If we wish to send an electronic mail from a Web browser, we need a mail gateway to actually send the message. A mail gateway is the machine that connects two or more electronic mail systems, and transfers messages between them. Mail User Agent (MUA) is the program that allows the user to compose and read electronic mail messages. The MUA provides the interface between the user and the Message Transfer Agent (MTA). MTA is the program responsible for delivering e-mail messages. Upon receiving a message from a MUA or another MTA, it stores it temporarily locally and analyses the recipients and either delivers it (local addressee) or forwards it to another MTA. In either case, it may edit and/or add to the message headers. The Unix MTA supports mail transport via TCP/IP using SMTP. MTA is normally invoked in the background via a MUA such as the miniature mailer (Minimailer).

6.5.16 Security

Our SVS (Secure Virtual Secretary) is the automated meeting scheduler that provides the users with high security in addition to the user authentication. VS2 uses PGP (Pretty Good Privacy) for encrypting the e-mails before sending to the users. PGP is a public key encryption package to protect e-mail and data files. It lets VS communicate securely with its users, with no secure channels needed for prior exchange of keys. PGP is well featured and fast, with sophisticated key management, digital signatures, data compression, and good ergonomic design. Chapter seven explains public key encryption techniques, our infrastructure for security implementation, and other security issues. This section just explains how SVS applies PGP to send the e-mails containing the passwords to the users.

It is a necessary requirement for SVS to have the user identification of all its users and their public keys. The user identification is an ASCII string used to identify a user. Our SVS assumes the user ID is the same as the user e-mail address.

The following are the steps required to be done in order to use PGP in SVS:

- ❑ The users and SVS generate their own unique private and public keys.
- ❑ SVS should have the users' public key. The private keys are secret to the users.
- ❑ SVS adds the users' public key to its key ring. The key ring are basically two files (pubring.pgp and secring.pgp) containing two sets of names for public keys and private keys respectively.

- ❑ SVS encrypt the message using the user's public key and digitally sign the message using its private key.
- ❑ On receipt, the users should save the message into a file and decrypt the cipher message using their own private key.

7 Security

7.1 Introduction

CGI programs let anyone run a program on the system. This allows an anonymous user from any where to send it unexpected values, and to try to trick it into doing the wrong thing.

All CGI programs must be placed in a distinct directory. In our work, the directory path is `"/home/grad/www/salamat/cgi-bin"`. The most important reason for this is system security. By having all the programs in one place, a server administrator can control and monitor all the programs being run on the system. However, there are directives that allow programs to be run outside of these directories, based on the file extension. The following directive, when placed in the `"srm.conf"` configuration file, allows the server to execute files containing `.pl`, `.sh`, or `.cgi` extensions:

```
AddType application/x-httpd-cgi .pl .sh .cgi
```

However, this could be very dangerous. By globally enabling all files ending in certain extensions, there is a risk that novice or malicious programmers might write programs that violate system security (e.g., printing the contents of important system files to standard output).

7.2 *Risk*

When it comes to WWW security, there are basically three types of risk:

1. Bugs or misconfiguration problems in the Web server that allow unauthorized remote users to:
 - Steal confidential documents not intended for their eyes.
 - Execute commands on the server host machine, allowing them to modify the system.
 - Gain information about the Web server's host machine that will allow them to break into the system.
 - Launch denial-of-service attacks, rendering the machine temporarily unusable.
2. Browser-side risks, including:
 - Active content that crashes the browser, damages the user's system, breaches the user's privacy, or merely creates an annoyance.
 - The misuse of personal information knowingly or unknowingly provided by the end-user.
3. Interception of network data sent from the browser to the server or vice versa via network eavesdropping. Eavesdroppers can operate from any point on the pathway between the browser and server including:
 - The network on the browser's side of the connection.
 - The network on the server's side of the connection.
 - The end-user's Internet Service Provider (ISP).
 - The server's ISP.
 - Either ISPs' regional access provider.

It is important to realize that the "secure" browsers and servers are designed only to protect confidential information against network eavesdropping. Without system security on both browser and server sides, confidential documents are vulnerable to interception.

7.3 CGI scripts

The problem with CGI scripts is that each one presents yet another opportunity for exploitable bugs. CGI scripts should be written with the same care and attention given to the Internet servers themselves, because, in fact, they are miniature servers. Unfortunately, for many Web authors, CGI scripts are their first encounter with network programming.

CGI scripts can present security holes in two ways:

- They may intentionally or unintentionally leak information about the host system that will help hackers break in.
- Scripts that process remote user input, such as the contents of a form or a "searchable index" command, may be vulnerable to attacks in which the remote user tricks them into executing commands.

CGI scripts are potential security holes even though users run the server as "nobody". A subverted CGI script running as "nobody" still has enough privileges to mail out the

system password file, examine the network information maps, or launch a log-in session on a high numbered port (it just needs to execute a few commands in Perl to accomplish this). Even if the server runs in a chroot directory (The chroot is a directory which can be run only by privileged users and is used to give a process such as FTP or HTTP access to a restricted portion of the file system), a buggy CGI script can leak sufficient system information to compromise the host.

7.4 Forms

One of the critical issues is when the CGI scripts deal with forms and they don't check the form data. A malicious user can embed the shell meta-characters (characters that have special meaning to the shell) in the form data. If a malicious user entered the following as the value of the user:

```
; rm * ; mail -s "Ha Ha" malicious@crack.net </etc/passwd
```

This might remove all the files in the current directory, and it would also mail the /etc/passwd file to the malicious user.

7.5 User authentication

In addition to domain-based security, most HTTP servers also support a more complicated method of security, known as user authentication. When configured for user authentication, specified files or directories are set up to allow access only by certain

users. A user attempting to open the URLs associated with these files is prompted for a name and password.

The username and password are checked by the server, and if legitimate, the user is allowed access. In addition to allowing the user access to the protected file, the server also maintains the user's name and passes it to any subsequent CGI programs that are called. The server passes the user name in the REMOTE_USER environment variable. A CGI script can therefore use server authentication information to identify users. Our VS is supported by user authentication.

Since our VS should create some text files for scheduling purpose to store information of a meeting remotely (such as meeting profiles and user profiles), these files and the directory which they belong to, should be world readable and writable. In this sense, to protect our directory and server from viruses, the system administration of the computing service that this application is supposed to be installed, should make that directory password protected. Therefore, only privileged users are allowed to use the application.

Server authentication doesn't provide complete security, since the user name and password are sent unencrypted over the network, it is possible for a "snoop" to look at this data. Therefore, it is obvious that our approach is not the best and most secure way of the system protection.

7.6 *Platform*

Unix systems, with their large number of built-in servers, services, scripting languages, and interpreters, are particularly vulnerable to attack because there are simply so many portals of entry for hackers to exploit. Less capable systems, such as Macintoshes and special-purpose Web server boxes, are less easy to exploit.

In the real world, of course, many sites will want to run a Windows NT or Unix server in order to gain the performance advantage of a multitasking operating system and the benefits of database and middleware connectivity. Security holes have been found in both Unix and Windows NT server systems, and new security holes are being found on a regular basis. On the whole Windows NT systems seem to be more vulnerable at the current time, partly because the OS is relatively new and the big bugs haven't been shaken out, and partly because the NT file system and user account system are highly complex and difficult to configure correctly.

7.7 *Our suggestions*

The following are our suggestions in order to improve VS in terms of security:

7.7.1 Using SSL

Secure Socket Layer is the scheme proposed by Netscape Communications Corporation. It is a low level encryption scheme used to encrypt transactions in higher-level protocols such as HTTP, NNTP and FTP. The SSL protocol includes provisions for server authentication (verifying the server's identity to the client), encryption of data in transit, and optional client authentication (verifying the client's identity to the server). SSL is currently implemented commercially on several different browsers, including Netscape Navigator, Secure Mosaic, and Microsoft Internet Explorer, and many different servers. including ones from Netscape, Microsoft, IBM, Quarterdeck, OpenMarket and O'Reilly and Associates.

7.7.2 *Using SHTTP*

Secure HTTP is the scheme proposed by CommerceNet, a coalition of businesses interested in developing the Internet for commercial uses. It is a higher level protocol that only works with the HTTP protocol, but is potentially more extensible than SSL. Currently SHTTP is implemented for the Open Marketplace Server marketed by Open Market Inc, on the server side, and Secure HTTP Mosaic by Enterprise Integration Technologies on the client side.

7.7.3 *Using Personal Certificates to control server access*

SSL can also be used to verify the user's identity to the server, providing more reliable authentication than the common password-based authentication schemes. To take advantage of this system each user will have to obtain a "personal certificate" from a CA

(Certifying Authority). The VeriSign Corporation was the first and still most widely used certifying authority.

A certification authority is a central, trustworthy entity whose certification key is commonly known and trusted. Every user in the domain of the certification authority can exchange its key with the certification authority in a secure manner in order to have it digitally certified by the authority. Other users can then verify the authenticity of the key by checking the signature of the authority.

7.7.4 Using Public-key encryption

The Public-key encryption scheme is introduced by Diffie and Hellman in 1976. Each user gets a pair of keys, called the public key and the private key. Each user's public key is published while the private key is kept secret. Messages are encrypted using the intended recipient's public key and can only be decrypted using his private key.

In the last three years, encryption utilities like Pretty Good Privacy (PGP) and Privacy Enhanced Mail (PEM) have matured to a point where they have begun to receive widespread acceptance among users of electronic mail on the Internet and Intranets. To achieve a maximum benefit from these security measures, though, we have to provide an infrastructure for our user, which includes trusted key servers, a key certification authority and a definite policy to establish such an authentication infrastructure.

We now explain the infrastructure that we have planned for VS:

A public key is used for authentication purpose using a private key accessible only to the users and a public key known to their VS. The users who want to decrypt an e-mail from VS has to publish their public key to VS. Fortunately there is a secure way to exchange keys. The users can digitally sign their public key. The digital signature basically is extra data appended to a message, which identifies and authenticates the sender and message data using the public key encryption.

There are three methods to produce a user's key pair:

- The user generates its own key pair, and its secret key is never released to another entity.
- The key pair is generated by a third party, which makes it necessary that the user receives its secret key in a secure way.
- The key pair is generated by a certification authority, a special third party, which fulfills appropriate security requirements.

7.8 *Result*

The task of establishing an authentication infrastructure is challenging and requires some time and effort. In this chapter we described only some protocols that can be added to make VS more secure. The cryptographic security services don't specify a standard of any kind and discussion of this issue is unlimited. We tried to introduce the importance of security and its relationship to our work. In this sense, VS could be extended to provide greater security. This extension can be provided by creating a new infrastructure or by the

trust that can be made by CAs. Finally, the question may arise as to which solutions should be supported. Should we go for using the encrypting utilities such as PGP, PEM, S/MIME, etc., or how can we implement our own infrastructure? These are future work!

8 Conclusion

8.1 *In General*

Automating meeting scheduling is important. VS cannot only save time and effort, but also may lead to more efficient scheduling. This dissertation has explained how VS implements the task of scheduling a meeting. By utilizing the invitee's preferences and the host's priorities, VS can provide more effective scheduling than manual processes used by human secretaries. Our VS has focused on representing and using user preferences to categorize acceptable meeting proposals. We have also shown how to produce a consensus for meeting times and location using quorum while balancing different user preferences and priorities. Our VS has been designed so as to minimize the amount of supervision the user must provide.

The benefit of such a software agent is to allow users to concentrate on more productive tasks. VS not only saves time and effort, but also improves the quality of information processing by preventing errors that might be introduced by secretaries.

8.2 *Future Work*

Although in the first version of VS we have focused on one secretarial task, e.g. meeting scheduling, we expect that eventually VS will take care of most secretarial services.

Some of the extensions that we are actively pursuing include learning user preferences and priorities by observing users, providing more secure environment for the users of VS, receiving the user preferences and priorities as many as possible, and scheduling the meetings accordingly. We are also interested in extending the protocol to multi-agent systems where the virtual secretaries cooperate with each other. This would be useful when VS doesn't have enough knowledge to do a task, and then it would use the other virtual secretaries in order to accomplish that task.

The Internet has made remote conferencing a widespread reality. This has added a new and challenging dimension to the problem of scheduling meetings and conferences. The stress on a participant due to the geographical time difference needs to be taken into account while scheduling a meeting. In a global event, participants may locate in different time zones. People in different parts of the world have different local calendars having different working hours. All participants will naturally prefer a meeting to be scheduled within their working hours. This is one of the issues that would make VS more powerful.

The ultimate objective is, the users should give the order of what to do, while VS figures out how to do and carry out the tasks.

9 References

[Cao et al, 1996] W. Cao, C. G. Bian & G. Hartvigsen: Cooperator-Base + Task-Base for Agent Modeling: the Virtual Secretary Approach. In Proceedings of Agent Modeling Workshop, 13th National Conference on Artificial Intelligence - AAAI'96, pp. 105-111, AAAI Press, Portland, Oregon, USA, 1996.

[Cao et al., 1997] W. Cao, C. G. Bian & G. Hartvigsen: Achieving Efficient Cooperation in a Multi-Agent System: the Twin-Base Modeling. In Proceedings of Cooperative Information Agents - DAI meets Database Systems (CIA'97), Lecture Notes in Artificial Intelligence, Vol. 1202, pp. 210-221, Springer-Verlag, Kiel, Germany, 1997.

[Hackos and Redish, 1998] JoAnn T. Hackos & Janice C. Redish: User and Task Analysis for Interface Design, 1st ed., John Wiley & Sons Inc, 1998.

[Hartvigsen et al., 1995] G. Hartvigsen, A. Helme & S. Johansen, A Secure System Architecture for Software Agents: The Virtual Secretary Approach. In Proceeding of the Second Broadcast Open Workshop, 1995.

[Kosba and Wahlster, 1989] A. Kosba W. Wahlster: User Models in Dialog systems. Springer-Verlag, 1989.

[Maes, 1994] Pattie Maes. Agents that reduce work and information overload. Communications of the ACM, 37(7):31-40, 1994.

[Nanard at al, 1993] Jocelyne Nanard, Marc Nanard, Anne-Marie Massotte, Alain Djemaa, Alain Joubert, Henri Betaille, Jaques Chauché: Integrating Knowledge-based Hypertext and Database for Task-oriented Access to Documents. DEXA 1993: 721-732

[Sen and Durfee, 1991] Sandip Sen and Edmund H. Durfee. A formal study of distributed meeting scheduling: Preliminary results. In Proceedings of the ACM Conference on Organizational Computing systems, pages 55-68, 1991.

[Sen and Durfee, 1994] Sandip Sen and Edmund H. Durfee. On the design of an adaptive meeting scheduler. In Proceedings of the Tenth IEEE Conference on AI Application, pages 40-46, March 1994.

[Sen and Durfee, 1996] Sandip Sen and Edmund H. Durfee. A contracting model for flexible distributed scheduling. Annals of Operations Research, 65:195-222, 1996.

[Sen and Durfee, 1998] Sandip Sen and Edmund H. Durfee, "A Formal Study of Distributed Meeting Scheduling," accepted for publication in Group of Decision and Negotiation Support Systems, 1998.

[Sen et al., 1997] Sandip Sen, Thomas Haynes, and Neeraj Arora, "Satisfying User Preferences While Negotiating Meetings," *International Journal of Human-Computer Studies*, vol. 47, pages 407-427, 1997 (special issue on Group Support Systems).