FAKULTÄT FÜR !NFORMATIK

Meeting Scheduling Support using Mobile Clients

MASTERARBEIT

zur Erlangung des akademischen Grades

Magister der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Markus Niederer
Matrikelnummer 0126384

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer/Betreuerin: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Andreas Rauber
Mitwirkung: Univ.-Ass. Dipl.-Ing. Dr. Alexander Schatten

Wien, 12.10.2009

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Erklärung zur Verfassung der Arbeit

Markus Niederer
Siedlerstrasse 3
6972 Fussach

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 16. Oktober 2009 Markus Niederer

Abstract

This work describes the conception, modeling and prototypical implementation of an agent-based meeting scheduling system for mobile devices. This mobile agent system supports people on the move to locate feasible time slots to meet with each other, using their mobile devices. It negotiates the best suitable times integrating timetables and time preferences from the participants. Unlike to current approaches, the negotiation of the meeting time is fully executed on mobile devices. First, the basic prerequisites of meeting scheduling, as well as typical scheduling situations for business and spare time life are identified. The scenarios are used to evaluate the current scheduling system approaches, discussed in the second part of the work. Based on the results of the evaluation of the current approaches, a new meeting negotiation system for mobile devices will be designed. As mobile devices are limited in resources, the introduced protocol additionally allows to outsource the complex part of the negotiation to a self-contained scheduling service. From the theoretical design, a prototype based on the Android [45] platform was implemented. Finally, the new prototype will be verified against the business and spare time scenarios and future improvements will be discussed.

Zusammenfassung

In dieser Arbeit wird die Frage erörtert, wie die Suche nach geeigneten Zeiten für Besprechungen bzw. Treffen von zwei oder mehreren Personen schnell und effizient durch mobile Endgeräte unterstützt werden kann. Ziel ist die Konzeption, Modellierung sowie prototypische Implementierung eines Agenten-basierten Terminvereinbarungssystems für mobile Endgeräte. Mit diesem System sollen sich sowohl Termine von Personen, als auch von Ressourcen untereinander abstimmen lassen. Im Rahmen dieser Masterarbeit wurden bestehende Ansätze und Systeme für Gruppenterminvereinbarungen untersucht und anhand von zuvor definierten prototypischen Szenarien bewertet. Nicht alle Systeme sind uneingeschränkt für moble Geräte einsetzbar. Bedingt durch die knappere Verfügbarkeit von Ressourcen und eingeschränkte Bedienbarkeit, bedingt durch ihre Größe, haben mobile Geräte besondere Ansprüche an Applikationen. Die Anwendungen müßen diesen speziellen Anforderungen angepasst sein, um eine einfache und effiziente Bedienung zu ermöglichen. Die bestehenden Systeme wurden besonders im Hinblick auf die Anwendbarkeit auf diese Beschränkungen bewertet und verglichen. Aufbauend auf den Ergebnissen der Evaluierung wird ein Modell eines speziell für den mobilen Gebrauch zugeschnittenen Agenten-basierten Terminvereinbarungssystem entwickelt. Für die Kommunikation der Agenten untereinander wurde ein für mobile Systeme optimiertes Protokoll spezifiziert. Zur Untermauerung der Thesen und zur Überprüfung des Modells wurde ein Prototyp des Systems auf Basis der Android Plattform erstellt. Anhand der eingangs definierten alltäglichen Terminvereinbarungsszenarien wurde der Prototyp bewertet und Erweiterungsmöglichkeiten für zukünftige Entwicklungen aufzeigt.

Acknowledgements

I thank Dr. Alexander Schatten for his patient coordination and the helpful feedback. Thanks are also due to Prof. Dr. Andreas Rauber for supervising this thesis. Special thanks go to my family and my friends for their continuous support during all my years of studying.

In memoriam
Rudolf Niederer
(1946 - 2007)

Contents

List of Figures

Listings

List of Tables

1 Introduction

In the last few years the number of people owning and using mobile phones or personal digital assistants (PDAs) has increased significantly. To acquire new customers, telecommunication providers seem to submit new offers nearly every day. The prices for mobile communication including mobile Internet services decrease constantly. The results are obvious: People are potentially reachable anywhere at every time. SmartPhones and PDAs have become our permanent companions. On the other hand, time has become an invaluable property. Not only in business life the dictum "time is money" faces us every day. Scheduling meetings has become an increasing difficult activity for busy people. Several applications support people to find time slots to meet each other. The techniques to schedule meetings range from simple 'free-busy' calculations [34, 23] to preference-based approaches [6, 7, 11, 17]. It seems natural to bring scheduling applications to mobile devices, as they are permanently available and additionally often have calendar and address book functionality already in place. Nevertheless, mobile devices do still have limitations in computing resources, but most of all in visualization and usability. With every generation mobile devices become more and more powerful and open possibilities for new areas of application. This work will go into the matter how meeting scheduling can be supported by using mobile devices and is structured as follows: In chapter 1 the basic requirements, useful to help people find acceptable times to meet with each other, are explored. Chapter 2 describes current approaches, which are subsequently evaluated in chapter 3. From the results of the evaluation, a new scheduling protocol, fully designed to support meeting time negotiation on mobile devices, is defined in chapter 4. Based on this protocol a prototypical implementation called MeetMe was implemented. Chapter 5 describes the development of the MeetMe prototype and gives a system overview. Finally, chapter 6 evaluates the concept and gives an outlook for future research work.

The first prototype of MeetMe was submitted to the Android Developer Challenge (ADC) in April 2008 and reached the top 25 percent of the 1,788 submissions. Furthermore, a paper based on this work was accepted for the Second International Conference on the Applications of Digital Information and Web Technologies (ICADIWT) in 2009.

1.1 What is Scheduling

This section describes the basics of meeting scheduling as well as the two main types of scheduling techniques. The second part of this section introduces the actors involved during a scheduling process and describes their rules. In 1984 Kincaid *et al.* [27] analyzed several electronic calendaring systems and they defined the characteristics of automatic schedulers as:

> "The scheduler is a unique feature in that it provides a bridge between the calendar and the electronic mail facilities." [27]

Kincaid *et al.* showed that the major drawback of the schedulers relies heavily on the accuracy and completeness of individual calendars. They defined the following features as required for electronic schedulers:

- Users should be allowed to specify who is to attend (not necessarily themselves).
- The specification of a meeting's time range should be allowed.
- All possible meeting times should be presented, and the user permitted to select the most appropriate time.
- A warning should be given when conflicts arise, but conflicts should not prevent a meeting request from being sent.
- Each participant should be notified of the tentative meeting, and the system should request a response from each. The participants, however, should be allowed the option of postponing their reply to a later time.
- The person setting up the meeting should be allowed to automatically cancel or confirm the meeting, with a notification sent to all participants.
- It should be possible to book resources along with meetings.

1.1.1 Involved Actors and their Rules

Depending on the subject of the meeting, a great number of persons can be involved in scheduling an event. But independent of the quantity of persons invited or involved in the meeting scheduling process, these persons can be classified into three different types, each acting in a specific rule:

Initiator The person that initiates and hosts a meeting. The initiator is responsible for the meeting, owns the power of decision and therefore acts as a kind of supervisor. He or she decides where the meeting is held and who to invite. Moreover, in exceptional situations (e.g., no feasible time for each invitee can be found) the initiator decides how to proceed. Often the initiator refers the work of the negotiation to a second person, the coordinator.

Coordinator The person who coordinates the scheduling of the meeting. In most cases, specially in business life, the coordinating part is executed by the secretary of the initiating person. The coordinator is the central point, where the wishes of the attendees are collected. The secretary collects the preferences from the participants and subsequently matches the favorite times for the meeting. Frequently it is necessary to collect the preferred times from the invitees in several steps or negotiation rounds: The coordinator provides a time for the meeting, the invitees decide to accept or to reject. If the time is rejected by one mandatory person, the coordinator provides another time, waits for the responses etc. Mostly, the coordinator has to consult the initiator to solve unexpected situations during the meeting scheduling process.

Participants The attendee or invitee, a number of people that need to attend the meeting. Chun *et al.* classifies the attendees as those that "must", "should" or "can" attend the meeting. Those participants that "must" attend, all have to be available for the meeting to be confirmed. Otherwise no time slot can be found and the meeting is canceled [8]. "Should" attendees are the normal participants of a meeting. The coordinator and the initiator can also be participants.

Each of the actors can play multiple roles. The initiator can also be the coordinator and / or a participant of the meeting. The coordinator can be the initiator or the participant as well.

1.1.2 Techniques of Scheduling Support

Various systems support people to find feasible times to meet with each other. The techniques of meeting scheduling support systems can be classified in two main types of approaches:

- Shared personal calendars
- Agent-based meeting scheduling

In shared personal calendars based systems the core tasks of the scheduling job has to be done by humans, mainly by the meeting initializing person. A person, who wants to negotiate a meeting, selects all required participants and resources. The initiating person selects a time slot from the publicly available calendars of the participants and resources. Some systems suggest automatically the first free time slot. The chosen time then is sent to the participating persons, who can accept or reject the time, or — some systems offer this feature — they can suggest other, more favorable times.

If the initiating user accepts, a scheduling message is sent to the clients of the attendees. The participants have the possibility to respond to the scheduling request. If an attendee is not content with the suggested time, he or she can propose an alternate time slot and send it back to the initiating person. The initiator can view these responses and decide how to proceed, either to insist on the first suggested time or to accept the time proposed by the attendee by starting a new round of negotiation.

In agent-based approaches the main part of the indispensable work is delegated to personal scheduling agents, attending to its owner's interests. Chapter 2 introduces different applications using different types of agents. As big advantage agent-based systems discharge the human user from reiterating work, e.g. searching for time slots in the calendar, responding to the requests of the initiator and so on. On the other hand, these agents need big knowledge about the preferences of their users, which implies a long period of training.

1.2 Integration of Data

Beside the main information to schedule a meeting such as *time*, *date*, *attendants* and *location* various other data can be integrated to improve the scheduling results.

This part will describe additional information used by the various systems and how these information influence the scheduling process. Franzin *et al.* [15] showed the direct effect of privacy on the efficiency of the negotiation and the quality of the solution within meeting scheduling systems. The less information provided from each participant, the longer the negotiation process will take, with negative effects on the scheduling result. As a basic set of information, Haynes *et al.* [22] and Sen *et al.* [43] defined the required data for a meeting by a number of parameters:

- Set of attendees.
- Proposed length of the meeting.
- Priority assigned to the meeting.
- A set of possible starting times on the calendar for the meeting (e.g. sometime next week, Friday afternoon, etc.).
- A scheduling deadline.
- Any additional constraints (e.g. the location where the meeting will take place).

It is important for the participants to know where to meet. Especially in distributed settings, the location becomes particular important. Dependent on where the invitees stay within the regarding time when the meeting should be held, it can be favorable or impossible for them to participate. Chun *et al.* extended the set of information exchanged between the attendees. In their work, they defined a set of required data for their meeting scheduling agent system as well [8]:

Initiator The host or initiator of the meeting. A person might consider a meeting called by his/her immediate supervisor to be more important than the others, for example.

Rank The rank or position of the person calling the meeting. Values can be any rank or position within the organization.

Attendees A list of participants or invitees that need to attend the meeting.

Type The type of meeting which can be used to determine the priority of the meeting during scheduling. Meetings with higher priorities might take over time slots from previously scheduled meetings with lower priority which can be enforced to be rescheduled. Chun *et al.* list as possible values: "*general, departmental, group, strategic, inter-departmental, technical, marketing, sales, project, interview, etc.*".

Period The time period when the meeting should be held. The exact date and time is represented by other attributes. Chun *et al.* mention as example *"within the coming 2 weeks, within this week, within Friday, etc."*.

Duration The length of the meeting. Possible values may be a number of hours or minutes.

Part-of-day The part of the day that the meeting will be held, as a more nature in defining time preferences. As possible values Chun *et al.* list *"breakfast, morning, lunch, afternoon, dinner or evening"*.

Day-of-the-week Day-of-the-week that the meeting will be held, such as Monday, Tuesday, etc.

Additionally to the required data defined in [8, 43] Garrido and Sycara [16] specified three ranges wherein a meeting can be scheduled:

Date window specifies the range of days when the meeting can be scheduled.

Start-time window specifies the range of start times when the meeting can be scheduled.

Duration window specifies the range of time when the meeting can be scheduled.

As mentioned in section 1.1 Chun *et al.* additionally defined a further classification according to the priorities of the attendees:

- "Must attend": The invitee has to attend the meeting. All participants classified as "must" have to be available before the meeting can be confirmed, otherwise the meeting will be canceled.
- "Should attend": Is the standard classification of the participants where the invitee should attend the meeting.
- "Can attend": Invitees which are classified as "can attend" are casual observers of the meeting. Their availability will not affect the scheduling of the meeting.

1.3 Requirements for Mobile Devices

With every generation mobile devices become more powerful and hence open possibilities for new areas of application. It seems natural to bring scheduling applications to mobile devices, as they are permanently available and additionally often have calendar and address book functionality already in place. Nevertheless, mobile devices do still have limitations. Due to their size, mobile devices are limited in visualization, large icons and big fonts are prerequisite. The new tendency to touchscreen controlled devices expect not only big buttons. It is necessary to put special emphasis on the operability of the application. The user interface should be simple and intuitive, but powerful enough to support even difficult meeting scheduling situations. Furthermore, mobile devices have limitations in computing resources. Processes calculating sophisticated computations could allocate to much resources and probably cause inconvenient breakdowns of the mobile device. Essential functions like phone connections could be interrupted when the calculation takes too long on cellphones. These limitations should be kept in mind while developing applications for mobile devices. Therefore, complex and extensive calculations have to be avoided, which seems to be a natural border for the scalability of the system. Hence, the number of agents participating the negotiation process is limited.

Furthermore, it cannot be ensured, that a mobile device is permanently reachable for communication. Either cased by the lack of telecommunication network, or — not uncommon — intentionally, because the user does not want to be accessible. If meeting scheduling is executed with mobile devices, messages for unavailable devices have to be stored, until the device is reachable.

1.4 Scheduling Scenarios

In this section various scenarios scheduling meetings are described. When people want to meet, they contact each other to negotiate a time feasible for all participating persons. In some cases one person is instructed to play the coordinating part, contacting every invitee, record their preferred times and finally match the preferences to extract the best fitting time. In business life this part is often played by a secretary, minor by the initiating person itself. More and more electronic helpers — some of them are described in the second part of this work — support people to negotiate a good time to meet.

Life knows multiple situations, where people want to meet with each other. Very often it is not easy for people to find a suitable time to do so. Especially when the people, who want to meet, are very busy, work for different companies, or do life spatially divided, e.g. in different parts of a country.

The following scenarios should help to evaluate and categorize the existing systems in chapter 2 and make it easier to compare with the resulting prototype of this work in chapter 5. The scenarios are divided into two categories: The first scenarios describe typical business situations and the second type characterize some spare time scenarios. The two types each self are subclassified into scheduling situations, where persons want to meet, and situations where resources are required as well. Resources could be meeting equipment like meeting rooms, beamers, overhead projectors, or sport places for spare time scenarios.

1.4.1 Business Scenarios

The first part of this section describes four scenarios arising frequently in business. As mentioned in the introduction of this section, the organization of a meeting is often delegated to a third person, a secretary that itself mostly is not participating in the concerning meeting. She or he is asked by the initiator — in many cases a superior of the secretary — to organize the meeting within a specific time period, e.g. within the next days, within one week, etc. The secretary then contacts each of the persons requested for the meeting to invite them and gather their preferences of when to meet. After this first step the secretary matches the preferences of the invitees and chooses the best suitable time for all participants. Afterwards she or he has to contact the participating persons again, to ask if they are comfortable with the chosen time and if not, to offer an alternative time. When one of the invited persons is not satisfied with the result, the secretary has to choose another time and call them again for confirmation. As may be imagined, the more persons invited for a meeting, the search for a adequate time gets more sophisticated. The search will get more complex, when the negotiation of the meeting time is executed

only by secretaries, and these third persons — not part of the underlying meeting — have to query every request with the invited person.

The first scenario describes a simple business situation: a meeting has to be scheduled within one branch of one company. The second scenario describes the search for a meeting time within one company, where some traveling sales persons are invited. In the third scenario people working for different companies want to meet and the last situation describes the special requirements, when additional resources are needed for a meeting.

Business Scenario 1 - Within one Company

The first scenario describes the meeting scheduling between three persons, working in the same branch of one company. These employees are on the same hierarchic level, they are working inside the same building, but belong to different departments. When these workers like to meet, one of them will take the initiative and start the negotiation process, in order to find a good time to hold the meeting. He or she starts by inviting the others and asking them for their preferred times to meet. Furthermore, the initiating employee will match these preferences with his or her own, and generates one or more time proposals when to hold the meeting. Then, he or she will ask the others, if they accept one of these times and will match these preferences again. When they all agree with one of the proposed times, the time for the meeting can be fixed. Otherwise the negotiation process has to start again, with altered parameters, e.g. the time horizon to hold the conference can be enlarged.

To negotiate the meeting they either communicate by email, talk over phone or voice over IP, or they have a short meeting to talk when to hold the meeting. In this special case — because they work in the same building, with their departments not very far apart — this is a possible option. In chapter 2 several approaches for electronic meeting scheduling support will be introduced. A great number of companies offer their employees one of these supporting tools.

Business Scenario 2 - Same Company with Traveling Sales Person

The second scenario extends the first situation introduced above, by involving one or more employees working outside the office area. In this scenario a meeting should be held with one traveling sales person. The negotiation of the best fitting meeting time is coincident to the scenario between the employees within the same building. One participant will start by inviting the others and ask them for their preferred times to meet. These can be done by phone, voice over IP or email. In some cases — when the traveling employee is not permanently abroad — they could meet to talk about the time to have the meeting.

Business Scenario 3 - Different Companies

The negotiation of a meeting time feasible for every participant will get more complex, when the invited persons work for different companies, situated on different locations. As we can imagine, these type of meeting scheduling in general take more time to be done and is more resource consuming than the scenarios shown before.

It is obvious that companies not closely related with each other, will not have a cooperative calendaring or any kind of groupware system. Furthermore, to get people together, they have to step back and use more widely established methods, like negotiation by phone, email or nowadays by voice over IP.

Business Scenario 4 - Different Companies considering Meeting Place

Additionally to the meeting scheduling situation outlined before, in this scenario the place where to meet has to be considered within the scheduling process. Particularly in business life meetings often are held in representative and specially equipped conference rooms. These kind of resource can be handled like a mandatory participant: When the conference room is not available because it is fully booked, the meeting has to be held at another time or — if available — in another room. To prevent the room from overbooking, it is not unusual to have an own calendar for the room. Furthermore, such resources in demand often do have an administrator to manage the appointment times. When it comes up to schedule a meeting considering a conference room, two different approaches can be identified: If only one conference room is available, the coordinating person will first visit the calendar of the conference room or contact the room's administrator to find out, when the room is free. Then, the coordinating person will send the limited timetable to the participating persons and ask them to choose the times that fit best for them. The remaining meeting scheduling will then be processed as described in the scenarios before. When more than one conference rooms are available, the coordinating person probably first ask the participants for their preferences and consult the conference rooms calendar afterwards.

Which strategy meet the requirements of the participating persons the best, depends on the situation, furthermore it is left to the coordinator's discretion to take the right approach.

1.4.2 Spare Time Scenarios

This section will describe the probably easiest scenario to negotiate a meeting, a scenario between persons who want to meet in their spare time. For the following scenarios we assume that the persons are very autonomous, they do not need third, uninvolved persons to keep their schedule. The people know each other quite well, they have a presentiment about the time tables of their friends and know about their preferences when to meet. The first scenario describes probably the simplest case, how two friends want to negotiate a meeting with each other immediately. The second scenario describes how a group of more or less good related friends could organize a special type of meeting: they want to meet to go in for sports. In this meeting scheduling situation a sport place has to be considered for the negotiation.

Spare Time Scenario 1 - Friends want to Meet

This part will describe the probably easiest scenario to negotiate a meeting, a scenario between two persons knowing each other very well. These two persons want to meet very shortly. Each participant knows the preferences of the other quite well, but we can assume, that they do not exactly know when the other person is

free or busy. While they want to meet in short term — in the evening this day — the communication and negotiation of the meeting time has to happen very quickly. It is obvious that the two friends will contact each other by phone or voice over IP. Another possibility would be the communication by email, but — as many people are not always online — this type of conversation could be too time-consuming, when one opponent or both do answer requests delayed. A forth possibility to communicate near real time is to negotiate a feasible meeting time using Internet Chat. One way to asynchronously interact between persons, that cannot be neglected is the Short Message Service or SMS. These type of communication often used between young people, comes up with the benefit that one or both sides can move wherever they want, not to be bound to a stationary service. The inviting person starts the negotiation by contacting the counterpart and asking him or her to meet. As mentioned above, we assume that the initializing person does have an idea about the invited persons preferences, so he or she can start by offering some time slots best fitting the companions preferences and his or her own. It is obvious, that the knowledge of the others preferences makes the negotiation much easier.

Spare Time Scenario 2 - Friends go in for Sports

The last scenario introduced, is very similar to the scenario described before. As mentioned above some friends want to meet, but in this case, they have a special intention: they want to go in for sports. For later evaluation of this scenario, it is not important to know, which type of sports they want to go in. A team sport can be assumed, where more than one attendee is needed. To find a feasible time to meet, these friends have to consider the availability of the sports place as well. Furthermore, in some cases the availability of the sports place is more important, than the availability of all players. If the sports place is not available, it is impossible for them to play, no matter if they want to play soccer in the park, squash at a squash court, or golf somewhere at the green.

2 Related Work: Current Concepts of Scheduling Systems

This chapter describes several current concepts of scheduling systems. The first part will survey the types of scheduling techniques. The second part will take a closer look at the scheduling methods used in the current concepts. The major task in meeting scheduling is to find the most feasible time. The third part will outline common negotiating strategies and protocols to negotiate best time slots for scheduled meetings and finally in the fourth section the underlying technical protocols to transfer the scheduling messages are debated.

2.1 Types of Scheduling

The major techniques of meeting scheduling have been already introduced in section 1.1 and can be classified in two types of approaches:

- Shared personal calendars
- Agent-based meeting scheduling

In systems based on shared personal calendars the core tasks of the scheduling job has to be done by humans, mainly by the meeting initializing person, whereas in agent-based approaches the main part of the indispensable work is delegated to personal scheduling agents. In the next sections the difference between these two scheduling techniques are discussed.

2.1.1 Shared Personal Calendars

Probably one of the most spread method in the business world to schedule meetings is used by Microsoft's Outlook [34]. To schedule a meeting efficiently, the initiating user needs to have access to all the participants' calendars.

A person who wants to initiate a meeting, selects all needed participants and resources. The algorithm of Outlook proposes the first time slot where all participants are free from the publicly available calendars. It is up to the initiator to accept or deny the suggested time. If the initiating user accepts, a scheduling message is sent to the Outlook clients of the attendees. The participants have the possibility to respond to the scheduling request. If an attendee is not content with the suggested time, he or she can propose an alternate time slot and send it back to the initiating person. The initiator can view these responses and decide how to proceed, either to insist on the first suggested time, or to accept the time proposed by the attendee and start a new round of negotiation.

Crawford and Veloso are calling this type of algorithm "Open calendar" scheduling:

> "The scheduling feature of Outlook rely largely on an open calendar system, where users are required to make their calendars publicly viewable within the organization." [11]

Crawford and Veloso pointed out a number of limitations and problems with this approach. They demonstrated that although the proposed time is marked free in a user's calendar, there could be an earlier request for the same time by another meeting. Furthermore, the shared calendar approach does not allow users to give detailed information about their preferences when they would like a meeting to occur and so it is quite possible that an unfavorable time will be chosen [10].

Another limitation of Outlook is that only one time slot can be proposed in every scheduling round. Outlook offers the opportunity to give counter-proposals, but every new suggestion has to be sent to all attendees again. Besides, the users can only accept or deny a proposal Outlook does not implicate a "soft" weighting of a proposed time slot. With Outlook it is possible to schedule the first business scenario, introduced in section 1.4. The scenario *Business Scenario 2* would be possible to schedule, if the traveling sales person is connected with the other participants by a mobile computer or does have access to the Internet. If the company, the participants are working for, does provide a Exchange Server [31] with OWA (Outlook Web Access)[1], the sales traveling person would be able to access its calendar and the calendar of the participating colleagues by a web browser. Full-featured access is only possible with Microsoft Internet Explorer 6 or newer, on other web browsers the OWA does have limited functions. If the traveling sales person does have access to the Internet, he or she would be able to participate the meeting scheduling process with his or her colleagues. The Web Access does provide the shared personal calendar feature of Outlook. Mobile devices are supported by Outlook Mobile [32], a lightweight mobile version of Outlook with limited functions. The mobile version of Outlook is runnable on different mobile devices, supporting the operating system Windows Mobile [33], but does not contain the meeting scheduling function of the desktop version of Outlook.

The disadvantage of shared personal calendar approaches is, as mentioned, that the initiating person needs to have access to the calendars of the participating colleagues. What is a desired feature within one company, could become an unwanted security problem in an enterprise overlapping setting. Although Microsoft Exchange and thereby Outlook provide domain traversing calendar access with different privacy levels, the sharing of personal calender data in the majority of the cases is reserved to the personnel inside a company. As a result the *Business Scenario 3* and *Business Scenario 4* are difficult to schedule with shared calendar approaches like Microsoft's Outlook.

Unlike companies, private persons possibly have less concerns sharing their personal calendar data with their friends. Therefore Outlook could be used to schedule meetings for the *Spare Time Scenario 1* and *Spare Time Scenario 2*, assumed the participating friends do have their desktop version of Outlook with them. As mentioned before, the mobile version of Outlook does not provide the shared calendar

[1] Outlook Web Access is a service integrated into the Exchanges Server 2007 and newer to access e-mail, calendars, contacts and tasks through web browsers, when the desktop version of Outlook is not available.

function.

Tullio *et al.* [47] claim that it is necessary to let users browse their colleagues' calendars to assess availability for meetings and that users must possess knowledge of how their calendars are accessed by others to effectively manage privacy settings. In 2002 they introduced Augur, an open model groupware calendaring system. Tullio *et al.* assume that the user's calendar information is hosted on the user's mobile device. The calendar data is automatically sent to a parsing module that synchronizes the data with a centralized relational database. An event matching module and a Bayesian network prediction module augments the data with information about attendance likelihood and events co-scheduled by colleagues. Additionally each user has a copy of the Bayesian network capable of learning their attendance habits over time on his or her mobile device. The augmented calendar information is accessible to every user by a separate web-service. With different visualization techniques Augur presents the user's events and the likelihood of colleagues' attendance to the meetings. Within his daily calendar view, the user is able to identify fellows who co-scheduled an event. In Augur it is essential for the user to have access to the colleagues' calendars. Shared calendar approaches allow only limited privacy for the user's data. High gradients of privacy will have negative effects on the result of scheduling meetings.

With its stationary and mobile configuration, Augur is capable to schedule the meetings in *Business Scenario 1* and *Business Scenario 2*, as well as the *Spare Time Scenario 1*. As mentioned before, *Business Scenario 3* and *Business Scenario 4* are difficult to schedule with shared calendar approaches. Besides the privacy reasons, Augur is designed to be used within one company and does not support the scheduling of resources. Therefore, it is not possible to schedule the meetings in *Business Scenario 3* and *4* and the meeting in *Spare Time Scenario 2*.

2.1.2 Agent-based Meeting Scheduling

To decrease the human communication overload while scheduling a meeting, the main part of the indispensable work could be delegated to personal scheduling agents.

Sen and Durfee define agents in a distributed meeting scheduling system as

> "The agents in a Distributed Meeting Scheduling (DMS) system exchange relevant information to build local schedules that fit into a globally consistent schedule." [41]

In their protocol, introduced in 1994, every meeting has a particular agent responsible for the scheduling process, called host. The host contacts the other attendees' agents (called invitees) to announce the meeting. The invitees offer information about their availability to the host agent, Sen and Durfee call this information bids. The bidding process is repeated several times before a mutually acceptable time interval is found, or the host agent terminates the scheduling, because no acceptable time slot can be found. The host tries to find the most favourable time slot by probabilistic analysis. Although the approach of Sen and Durfee provides several strategies for different meeting negotiation situations, they did not take account of resources. Each agent is assigned to a specific user, but not to resources. Therefore,

Business Scenario 4 and *Spare Time Scenario 2* described in section 1.4 cannot be negotiated with the distributed meeting scheduling system of Sen and Durfee.

As an extension of the shared personal calendar style approach of Microsoft's Outlook, Crawford and Veloso [10] designed an "Open-Negotiator", in order to schedule and re-schedule meetings in a fully automated and distributed setting. The Open-Negotiator tries to find possible time slots in a recursive way: First, all available time slots of the initiating user are marked as possible and are sent to the agents of the invitees. The agents of the participating people mark their available times as possible and send the times back. The initiating agent computes the intersection between the possible time slots of all participating agents. If the intersection is empty, the initiating agent marks one other of its available times as possible. In this second turn — if such an un-offered slot exists — the attendee agents mark an available slot as possible, otherwise an unavailable slot is marked possible. In the next step the initiating agent calculates the intersection between the possible slots again and — if one slot is left — the agent marks its favorite time from the intersection as pending. If this pending time slot is not marked as pending for other meetings, the attendees accept and mark the slot as pending ditto. Otherwise the attendees reject and mark one more time as possible. If all participants have marked a time as pending, the initiator confirms the time slot for the meeting. When the time conflicts with a newly confirmed meeting, the initiator marks the time slot as impossible, sends this to the other meeting's initiator, cancels the running negotiation and re-initiates the meeting scheduling procedure. As Crawford and Veloso discussed in their work, only the initiating agent of the meeting can express any preferences. If the intersection of possible time slots offers more than one time available to all attendees, the initiator chooses the time he or she favors the most. In the first round of scheduling the attendees have to send all available times at once without any possibility to express preferences about their favorable times. The protocol of Crawford and Veloso is designed to reflect the open calendar style approach of Microsoft Outlook discussed in section 2.1.1. Therefore, it is safe to assume that the same restrictions concerning the scheduling scenarios are valid for the Outlook system. This would mean, that *Business Scenario 1* and *2* as well as *Spare Time Scenario 1* can be scheduled, whereas the *Business Scenario 3* and *4* and the *Spare Time Scenario 2* cannot be negotiated with the Open-Negotiator.

Berry *et al.* [4] have introduced an personalized agent for time management and meeting scheduling called PTIME as a part of the cognitive agent system CALO, which is designed to assist in military situations [5]. The PTIME agent is a single-calendar scheduler dependent on other agents to coordinate shared calendar entities. It supports the collaboration between itself and its user and the collaboration with the PTIME agents of other users. The agents are designed to act on behalf of their users. They can provide information about upcoming events and learn about their user's preferences. Whereas CALO 1.0 requires multiple computers to run, the second version presented in 2006 is runnable on a single laptop. The main part of PTIME is its *Process Controller*, a SPARK[2] agent listening on possible interactions between its user and other agents. A so called *Constraint Reasoner* considers other information provided by CALO to ensure that the user is not over-committed.

[2]SPARK [38] is a Belief-Desire-Intention agent framework

PTIME also contains a *Preference Learner* as a support vector machine [17]. Unlike other agent-based approaches and calendaring systems, PTIME has access to rich information about the user's other activities [5]. Therefore, it is not fully comparable with other agent-based approaches. The advantage being part of the greater CALO framework, comes along with the disadvantage of being resource consuming and therefore currently not adaptable to be integrated on mobile devices. The approach of Berry *et al.* is focused to act on the behalf of its user. Therefore, it is safe to assume that *Business Scenario 4* and *Spare Time Scenario 2*, introduced in section 1.4, definitely cannot be negotiated. Due to the system requirements it is impossible to transfer the framework to run on small mobile devices. Consequently the meetings in *Business Scenario 2* and *Spare Time Scenario 1* cannot be negotiated either. However, the agents in PTIME transmit the calendar data in a peer-to-peer E-mail communication, which would allow meeting scheduling within a company or between different companies like in *Business Scenario 1* and *3*.

Chun *et al.* [8] introduced an agent-based negotiation algorithm within an environment they called "Mobile Agents for Office Automation" (MAFOA). To schedule meetings MAFOA uses two types of software agents:

Secretary Agent represents one person and has access to that person's calendar, preferences and constraints. The user's information is hidden from the other agents. The greater goal for this encapsulation is to enable the negotiation algorithm implemented in the agents to work on heterogeneous environments.

Meeting Agent acts as coordinator and manages the whole negotiation process. For every scheduling process there is one meeting agent. The meeting agent remains active to monitor events that might affect the meeting, even after the meeting is scheduled.

Chun *et al.* use a "propose - counter propose" protocol to negotiate the meeting time. To initiate a meeting process, a person sets up his or her secretary agent with the essential meeting data. The secretary agents create a new meeting agent to coordinate the negotiation process. The meeting agent first announces the new event to all attendee's secretary agents with the most preferred time of the initiating user as initial proposal. The secretary agents consider if the proposed time is acceptable or not. When they find a more favorable time slot, they send a counter proposal to the meeting agent. If this step does not provide a feasible solution, the meeting agent will send another proposal to the potential participants of the event. All proposals are stored in the "Proposal Table" at the meeting agent for later revalidation. Once the negotiation process yields to a result, this solution will be announced to all involved secretary agents. In another work Chun and Wong [9] introduced the "Nstar" (N*) negotiation algorithm for dynamic scheduling and rescheduling of meetings. The N* algorithm extends the MAFOA approach with announcements of *cancellation*, *availability checks* and replies to *proposal* and *availability*.

In the approaches of Chun *et al.* and Chun and Wong, the preferences of the initiating user could have big influences on the proposed time slots. The first announced propose is generated from the initiator's preferences and could lead to a premature result without having the other possibilities evaluated. The main type

of agent in the approaches of Chun *et al.* is acting on the behalf of its human user. This Secretary Agent is not capable to negotiate meetings for resources, like in *Business Scenario 4* and *Spare Time Scenario 2*. The agent based structure and the negotiation with strong privacy regarding the calendar data of the users would allow to negotiate meetings between companies as in *Business Scenario 3*. Although Chun *et al.* does not provide information about requirements of the technical infrastructure needed to implement the approach, it can be assumed, that *Business Scenario 1* and *2* as well as *Spare Time Scenario 1* are negotiable with this approach.

Hassine and Ho [21] introduced a model to solve dynamic meeting scheduling problems. They designed a distributed approach based on the DARC[3] model. Consistency algorithms are preprocessing algorithms that reduce futile backtracking [50]. In the DARC architecture the agents communicate peer-to-peer without a supervising instance. Hassine and Ho define two types of agents:

Interface agent corresponding with the user and the

Constraint agent negotiating the meeting with other users' agents. A Constraint agent can act as a **Proposer agent** to coordinate the negotiation process or as **Participant agent**.

Each agent has its own static and dynamic knowledge and a reasoning engine. In their approach, Hassine and Ho specify two main steps of negotiation. First, the Proposer agent transforms the initial meeting scheduling problem MS into another equivalent MS' by reinforcing local consistency. MS' is obtained as a result of the interactions between the Proposer agent and the Participant agents. In the second step the Proposer agent searches the best solution for its meetings, satisfying all of the participant's constraints.

A user can start the negotiation of a meeting through the Interface agent. The Interface agent activates the corresponding Proposer agent and makes it interact with all of the Participant agents. First the Proposer agent reduces the possible time slots for the meeting to its hard constraints and deletes all time slots where the meeting cannot be held (node consistency reinforcement). Then, the result is reduced from the times where other more important meeting are already scheduled (arc consistency reinforcement). Next the Proposer agent sends the final reinforcement result to the Participant agents to execute the node and arc consistency tasks and reduces the possible time slots according to their convenience. Finally, the Participant agents weights the possible dates according to its users' preferences by ranking the time slots in their users most preferred order.

If the reinforcement of the possible time slots delivers an empty result for a Participant agent, this agent sends a message to the Proposer agent to inform it about the non-possible meeting.

After all Participant agents have returned their weighted results, the Proposer agent tries to find the proposal with the best overall rank. The Participant agents then must check whether they can accept or reject the chosen time slot proposal A. If another meeting B has been scheduled in the meantime for the time slot, the Participant has to decide:

[3] distributed reinforcement of arc consistency

- When the weight of the other meeting B is higher than the weight of the meeting A, it has to reject the time for meeting A and ask the Proposer agent to relax its proposal.
- The weight of meeting B is less then the weight of A, the agent proceeds in two steps: First, it sends a positive answer to the Proposer agent of meeting A, then it invites the Proposer agent of meeting B to relax its preferences for the meeting.
- If the weights of both meetings are equal, the Participant agent should try to apply the metropolis criterion to decide which meeting has to be rescheduled.

The process resumes until an agreement among all of the participating agents is reached. If the negotiation leads to no result, the Proposer agent informs the participants that the meeting is canceled.

In the approach described by Hassine and Ho the negotiation is done autonomous by two types of agents. Whereas the agents decisions are based on their users' preferences, it is impossible for the user to influence spontaneously the result of a running scheduling process.

With its multi-agent design, it is fairly to assume, that with the approach of Hassine and Ho the meetings introduced in *Business Scenario 1* could be scheduled as well as the company traversing *Business Scenario 3*. The agents formalized from Hassine and Ho schedule meetings in the behalf of human users, but not for resources. However with some smaller changes the Interface Agent could be capable to negotiate meetings for resources, like in the *Business Scenario 4* and in the *Spare Time Scenario 2*. Without any information about the technical requirements of the system, it is difficult to appreciate whether or not the approach could be integrated on mobile devices. For that reason it is impossible to assess if the system is able to schedule the meetings of the *Business Scenario 1* or *2*, or the meeting of the *Spare Time Scenario 1*.

Franzin *et al.* [15] showed the relation between privacy, efficiency and solution quality within meeting scheduling in multi agent environments. They developed a preference-enriched scheduling system on an existing system introduced from Eugene C. Freuder [13] in 2001. In their approach the agents communicate in several proposal phases. In every phase one of the participating agent proposes a time slot to the other agents, which they can either accept or reject by estimating the slot with weights between 0 and 1. A zero weighted time slot is treated as a rejection, in comparison a time slot rated with 1 is most convenient for the agent's user. When one or more agents reject the time slot by rating it with 0, a new proposal phase is started. The overall rating of a time slot is the minimum of the single participants' ratings. If all participating agents accept a proposal, the time slot is stored and a new negotiation process starts, where proposals leading to a overall ranking lower than the stored time slot are set to 0.

Franzin *et al.* are trying to find the balance between privacy and efficiency of meeting scheduling. They showed that the information exchanged by the agents during the proposal phases can be used to build an approximation of the constraint set of the other agents. In addition, whenever a proposal is accepted by an agent, the proposing agent can infer that the user of the accepting agent does not have

a meeting at that time. As total privacy within meeting negotiation cannot be provided, the efficiency of the scheduling process increases in a provable extent.

In their work Franzin *et al.* enrich an existing meeting scheduling system with preferences, but do not describe the agents used in detail. Therefore, it is difficult to gauge which of the scheduling scenarios introduced in chapter 1.4 can be negotiated and which not.

Beside the personal agent acting on behalf of its user, Demirel and Erdogan [12] introduced a location agent taking part in the negotiation process holding information on resources — in their case on meeting rooms. Demirel and Erdogan developed a meeting scheduler on the Java Agent Development framework (JADE) [2] integrating these two kinds of agents. The negotiation process in their approach has two stages:

- First the organizing personal agent sends an appointment proposal to the location agent. The location agent looks up the resources' calendar to find out if the resource is not in use at that time. If the time interval is convenient, the location agent accepts the proposal and blocks the proposed time slot in the resources' calendar. If the resource is not available, the location agent tries to find a counter proposal and sends the counter proposal to the organizing agent.

- In the second stage the organizing agent sends an appointment proposal to the participating agents. The invitee agents involved can accept or reject the proposal by returning a new time interval as counter proposal. The responses are evaluated by the organizing agent and it decides to start a new iteration with a new proposal or to cancel the negotiation process.

The meeting scheduling process ends successfully when all agents accept a time interval. Besides the negotiation of meetings the protocol of Demirel and Erdogan supports the cancellation of already scheduled meetings. The multi-agent framework makes it fairly possible to schedule the meetings in the *Business Scenario 1* and *3*. For resources, specially for locations, Demirel and Erdogan introduced an additional agent type, which allows to schedule the *Business Scenario 4*. Although the system of Demirel and Erdogan has not primary been designed for mobile devices, the JADE framework is able to run on mobile systems [29]. With some medium changes it would be possible to adjust the personal and the location agent for portable devices. But in the current design, the meetings in *Spare Time Scenario 1* and *2*, as well as the meeting in *Business Scenario 2* are not negotiable.

2.2 Scheduling Methods

The major task in meeting scheduling is to find the most feasible time. This section will take a closer look at the scheduling methods used in current concepts. The scheduling concepts can be categorized into two different types of strategies:

- "Free/busy" calculations and more complex

- Preference-based approaches

In "Free/busy" calculations the users' availability is divided into times a user is available (free) and times a user is unavailable (busy). In these binary systems — "*tertium non datur*" — no other option exists in order to express a users' availability. But for users some free times are more desired than others, e.g. a user prefers to be scheduled in the morning, while another user prefers to have meetings in the afternoon. None of these preferences are respected in these calculations. In comparison, preference-based approaches integrate miscellaneous user preferences. The preferences used in different approaches, will be described in the second part of this section.

2.2.1 "Free/busy" Calculation

The "Free/busy" calculation is used by traditional group calendaring systems such as Microsoft Outlook [34] or Lotus Notes [23]. Their technology to schedule meetings is basically a simple representation of users' calendars. In Microsoft Outlook and Lotus Notes these schedules are represented as a listed set of free and busy times. Brzozowski *et al.* [6] indicated that in traditional group calendaring systems finding a time that a group of people can meet at the same time, is simply a matter of choosing a time that all users appear to be free. As mentioned in section 2.1.1 in shared personal calendar based systems, the initiating user needs to have access to all the participants' calendars to schedule a meeting efficiently. The schedules of the other users are displayed side by side or on top of each other and the initiating user chooses one of the empty time slots when every participant is free. The scheduler supports the user proposing free times by hopping from one time slot, when all participants are free, to the next. A user can restrict the available times in these systems by considering daily times of availability for the proposing (e.g., a user is only available between 9 AM and 5 PM).

As Brzozowski *et al.* [6] mentioned, the binary view of availability is often inadequate to describe users' preferences. Beard *et al.* [1] extended the view of "free/busy" times by allowing the user to mark any given time slot with five levels of grey shading to indicate priorities. The lighter an event is displayed in the calendar, the less important it is for the user. Beard *et al.* have called this type of illustration "Transparency view".

2.2.2 Preference-based Approaches

Brzozowski *et al.* [6] introduced an application for preference-based group scheduling called "groupTime". The first prototype of this group scheduling system was build as an Excel workbook where a week of the users' calendar was represented as a sheet. In their research the users could weight their preferred times to be scheduled. The times were symbolized by a cell, in an half-hourly granularity. Brzozowski *et al.* defined the weight of the preferences as "*Can't Make It*", "*Rather Not*", "*Is OK*" and "*Works Great*". The Excel prototype was tested in a study with twenty students. Based on this study Brzozowski *et al.* constructed a basic ontology of how people prefer to schedule meetings [6] in an educational environment and created a more powerful Web application. After a user starts the negotiation of a meeting with the Web-based application, every invitee is informed by an email message,

which contains a link to respond. With this link the participants can indicate their preferences by selecting one of the four weights for the time slots within the specified time horizon. After every user has responded, "groupTime" sends an email to every participant containing the time slot with the highest overall weight. If a user feels uncomfortable with the proposed time, he or she can change his or her preferences and make the system renegotiate the meeting time. In another study with forty students Brzozowski *et al.* [6] learned that users are very accommodative when they feel uncomfortable with a proposed time. Users not only change times they do not prefer to "*Can't Make It*", they additionally tend to change times they prefer to "*Works Great*".

In their work, Brzozowski *et al.* focused on how preferences superior simple calendar information. They created a web-based group scheduling system aimed mainly for people working in academical environments. Even this open approach enables to schedule meetings with attendees from heterogeneous environments, e.g. people working for different organizations or companies. Hence, "groupTime" is able to schedule *Business Scenarios 1* and *3* introduced in chapter 1.4. Furthermore, with the web-based setting, mobile devices are able to connect straightforward, which let the traveling sales person in the *Business Scenario 2* participate the negotiation of the meeting as well as the friends on the road in *Spare Time Scenario 1*. The system of Brzozowski *et al.* has primarlly been designed to schedule meetings between human users. "groupTime" does not support the scheduling with resources, e.g. meeting places, beamers, etc. Therefore it is not possible to schedule the meetings in *Business Scenario 4* and the meeting in *Spare Time Scenario 2*.

To make meeting scheduling more effective Sen *et al.* [43] introduced a plenty of preferences for involved users to adjust. In their approach each user first rates all other users of the system. In addition every user provides preferences for meeting topics, meeting lengths, hours of the day and days of the week. Finally, the user can also weight the preference dimensions, i.e. the users can define which preferences are more important than others. Every user creates his/her own particular weightage scheme. Therefore each user assigns a value between 0 and 1 for each option of each dimension. Furthermore, the user specifies a minimum threshold for every dimension. If a value falls below the minimum criterion the user will not be scheduled for that meeting.

The system in the approach of Sen *et al.* needs rich information about the preferences of a user. Before a user can take part in a meeting negotiation, he or she has to specify a big set of preferences in different dimensions to create his/her particular preference scheme. People may not have the passion to specify all these preferences just to schedule meetings.

In the approaches of Sen *et al.* [43], Sen and Durfee [42, 41] each agent is assigned to one specific user. It is not possible to integrate resources into the scheduling process. Therefore it is unable to negotiate the meetings in *Business Scenario 4* and *Spare Time Scenario 2* described in section 1.4.

Chun *et al.* [8] have designed a system called "Mobile Agents for Office Automation" (MAFOA) where the user preference model associates *priorities*, *preferences* and *rules* with negotiable variable attributes. During the negotiation process the agent uses the priorities, preferences and rules to evaluate whether a proposal is acceptable or not and — if necessary — which counter proposals to make. The

attribute priority defines the rank of a particular attribute, e.g. the location could be more important for one user than for another. The rank of the attributes can be changed by priority points. Each agent has the same total number of priority points to use and the total value will not change during negotiation. The *preference value* is the amount of preference for a particular value, e.g., if one user prefers one location to meet more than another, or a user prefers to be negotiated rather in the "morning" than in the "afternoon", the preference value for times before noon are higher than the preference value for times after lunchtime. In addition to attribute priorities and preference values, each user may also have a finite set of *preference rules* that defines in how far these priorities and preferences may change. The rule conditions are defined as pattern of values on *fixed* attributes, *variable* attributes or *global* attributes shared by all agents. Chun *et al.* adduce as an instance of such rule: "If there is already a meeting on Monday morning, then I don't want any other meetings in the afternoon"[8]. During the negotiation process every agent evaluates the proposed times with its users' attribute priorities and preference values and creates as result a *preference level* of that proposal. The preference level defines how satisfied an agent is with a proposal. A proposal with a higher preference level means it is more preferred. The priorities and preferences in MAFOA can be adjusted by the user through a graphical user interface. For each attribute, the user can assign a priority by adjusting the corresponding slider.

As illustrated in section 2.1.2 Chun *et al.* are using two types of agents to schedule meetings, a Secretary Agent representing one attendee and a Meeting Agent acting as coordinator. Both agents are acting on the behalf of human users. The Secretary Agent is not capable to negotiate meetings for resources, like in the *Business Scenario 4* and the *Spare Time Scenario 2*. Nevertheless, the agent based structure and the negotiation with strong privacy regarding the calendar data of the users would allow to negotiate meetings company overlapping as in *Business Scenario 3*.

The approaches of Sen *et al.* [43], Chun *et al.* [8] integrate a great quantity of user preferences to enhance the scheduling process. They collect information about the user's preferences by asking the user. Their approaches require a lot of user interaction to adjust all preferences. But as mentioned before, people may not have the passion to specify all these preferences to schedule meetings.

Crawford and Veloso [11] proposed an approach whereby the user's preference information is captured by simply observing the changes in the user's calendar. They argued, that data of this form is easy to collect. The approach allows to assume as little as possible about how the user will be willing to schedule their meetings. Crawford and Veloso model the user's dynamic time preferences by placing probabilities on transitions between possible states of the user's calendar. The user's time preferences are represented by a weighted-directed-acyclic graph with a set of possible states, where the edges represent the probability of transitioning from one state to the next. To schedule a meeting the graph is used to rank the possible times, according to their probability. First, the state corresponding to the current state of the calendar is identified, then the successor states of the current state are ranked, which corresponds to a ranking of the possible times.

One great advantage of this approach is that no specific negotiation procedure needs to be assumed. It allows to capture the dynamic aspects of the user's preferences in an easy way. But as Crawford and Veloso mentioned, changes in a user's

calendar depend not only on the preferences of the user, but also on the preferences of the people the user meets. In section 2.1.2 the system implemented from the approach of Crawford and Veloso [11, 10], was introduced. The system was designed to reflect the open calendar style approach of Microsoft Outlook. Therefore it is safe to assume that the system is able to negotiate the meetings in *Business Scenario 1* and *2*, as well as in *Spare Time Scenario 1*, whereas the meetings in *Business Scenario 3* and *4* and the meeting in *Spare Time Scenario 2* cannot be negotiated.

In 1993 Kozierok and Maes [28] introduced a learning interface agent for scheduling meetings. This agent keeps track of everything its user does and stores the raw data about what happened as situation-action pairs. Furthermore, the agents maintain a set of priority weightings for meeting topic keywords and weightings for the relative importance of other participants. In the approach of Kozierok and Maes the agent has initial, general knowledge about how to schedule meetings. First, most of the meeting negotiation has to be done by the user. The agent learns little by little the habits and preferences by observing the actions of its user. Faced with a new situation, the agent compares this new situation with each of the situations occurred before and decides which action to take. In their work Kozierok and Maes have focused on how to make meeting scheduling more efficient considering user habits and preferences. They have not considered resources for meeting negotiation. For that reason the meetings in *Business Scenario 4* and the meeting in *Spare Time Scenario 2* cannot be negotiated with this approach.

Lin and Hsu [30] are using a machine learning approach to acquire user preferences for scheduling meetings. Besides the five main attributes to specify an activity *name, participants, location, required resources* and *duration* Lin and Hsu introduced three types of restriction and three associated types of preferences. The restrictions have to be observed to schedule a meeting and the preferences indicate choices among alternative schedules. The *time interval restriction* identifies the time, when an activity can be scheduled. The *precedence restriction* determine the ordering of two activities and the *time margin restriction* constraints the time margin between two following activities. Similar to the three restrictions, Lin and Hsu have defined three types of preferences: The *time interval preference* illustrates the preferences of a user, when to have specific meetings. The *precedence preference* models the execution priority of two activities and the *time margin preference* reflects the preferred time margin of a user between two activities. Preferences can be violated, restrictions have to be obtained. Lin and Hsu are using decision trees to induce schemata which are learned respectively. The input for the decision trees are instances which are described by sets of attributes. Each node of a decision tree specifies a test of some attribute of the instance and each branch descending from that node corresponds to one of the possible values for this attribute. The output of the decision tree for restriction schemata is a classification of either *allowed*, or *forbidden*. The tests against the decision tree representing the preference schemata generate a classification from *highly preferred, preferred, normal, disliked* to *highly disliked*. The activities are tested against the decision tree for time interval schemata. The input for the precedence and time margin schemata are activity pairs (each schema is associated with two activities). The learned schemata for preferences and restrictions are frequently updated.

In their work Lin and Hsu sketched a theoretical agent to learn a user's scheduling criteria. They proposed this agent could work with other calendar scheduling software to automate the scheduling process. This would allow this agent to negotiate meetings with calendar data from heterogeneous systems like in *Business Scenario 3*. Dependent on the system requirements the future prototype the meetings in *Business Scenario 1* and *2*, as well as the meetings in *Spare Time Scenario 1* could be negotiated. Lin and Hsu do not mention the possibility to consider resources within the meeting scheduling process. Therefore it is unlikely, that *Business Scenario 4* and *Spare Time Scenario 2* are negotiable with this agent.

Mitchell *et al.* [36] have described the design of the learning assistant CAP (Calendar APprentice) that learns its users scheduling preferences from experience. Whenever the scheduling assistant is used, it generates training data from every user interaction. Each night the agent learns general regularities from this training data. The acquired knowledge is used to increase the services offered by the software assistance. CAP learns rules that suggest the duration, location, day-of-week, and time of meetings. The knowledge is modeled as decision trees where each path through the decision tree is converted into a rule. Additionally statistics are maintained for each rule that summarize its performance. The rules are kept on a list, sorted by their past performance. When a new meeting is scheduled, advice will be generated by the topmost rule on the list that matches the new meeting attributes.

The agents in the work of Mitchell *et al.* are communicating by email messages. This widely spread message protocol would allow the agents to schedule meetings across organizational boarders, as in *Business Scenario 3*. The introduced agent is implemented as a desktop-based system which would prevent to schedule the meetings with participants abroad like in *Business Scenario 2* and *Spare Time Scenario 1*. The agents of Mitchell *et al.* are acting on the behalf of human users, resources cannot be integrated into the scheduling process. Therefore the meetings in *Business Scenario 4* and in *Spare Time Scenario 2* are impossible to negotiate.

Mynatt and Tullio [39] have created a calendar system extension called Ambush, using a Bayesian model to predict the likelihood of an user's attendance at an event. The Bayesian model was created manually from the results of interviews with students and faculty staff. The model specifies the decision to attend a meeting as a result of influences from the priority of the event, the priority of a conflicting event, if one exists, and the current availability of the potential attendee [39, 47]. The calendars of the users are scanned by autonomous components for new entries to teach the model. With transparency visualization techniques Ambush presents the user's events and the likelihood of colleagues' attendance to the meetings.

The system of Mynatt and Tullio [39], Tullio *et al.* [47] — closer introduced in section 2.1.1 — has a stationary and mobile configuration, which allows the system to schedule *Business Scenario 1* and *Business Scenario 2* as well as *Spare Time Scenario 1*. With shared calendar approaches *Business Scenario 3* and *Business Scenario 4* are difficult to schedule. The approach of Tullio *et al.* has been designed to be used within one company and does not support the scheduling of resources. Therefore, it is not possible to schedule the meetings in *Business Scenario 1* and *2* and the meeting in *Spare Time Scenario 1*.

2.3 Negotiation Algorithms

In this section different approaches of negotiation algorithms will be introduced: from "proposal - counter proposal" approaches, calendar information exchange with different graduates of privacy to approaches with relaxing rules over time.

Crawford and Veloso [11] have studied and discussed two types of negotiation strategies. In both negotiation strategies each meeting has an initiator, a list of attendees and a preferred day. First, all available time slots of the initiating user are marked as possible and are sent to the agents of the invitees. The agents of the participating persons mark their available times as possible and send the times back.

Offer-n-Negotiator in each negotiation round the initiator and the attendees add a set of times to their original offer to find an intersection of available times. After the initiating agent has received responses to its offer from all attendees' agents, it tries to create an intersection. If an intersection exists, the negotiation process switches into "pending mode". If no intersection can be found, the initiating agent starts the next round of negotiation by offering a new previously un-offered time for the specific day to the attendees. In pending mode the initiator sends a confirmation request to all participant agents. The attendees can accept or reject the time created from the intersection. When the time is accepted by all attendees, they can add the time to their calendar and the negotiation stops. If one or more attendees reject the confirmation request, the process switches back to "negotiation mode" and a new round of negotiation is started from the initiating agent.

Hold-Out-for-n-Negotiator is quite similar to the "Offer-n-Negotiator" strategy with following differences: In every round the initiator and the attendees only offer one new time. After offering two times the initiator 'holds out' and lets the attendees create their offers. If the attendees stop offering new times, the initiator proposes a new day. Times that are not in the attendees top two choices are not offered until a certain number of rounds.

In their work Crawford and Veloso are using a typical "proposal - counter-proposal" approach to negotiate meetings. In every negotiation round the initiator is offering an available time as proposal and the attendees respond with counter proposals. Under certain circumstances this type of negotiation can take many rounds to find a convenient solution.

As discussed in section 2.1.2 and 2.2.2, the approaches of Crawford and Veloso are designed to reflect the open calendar style approach of Microsoft Outlook discussed in section 2.1.1. The system is able to negotiate the meetings in *Business Scenario 1* and *2* as well as in *Spare Time Scenario 1*, but it is not able to negotiate the meetings in *Business Scenario 3* and *4* and the meeting in *Spare Time Scenario 2*.

In 2007 Wainer *et al.* [49] presented and discussed the efficiency of four types of protocols to negotiate and schedule meetings between agents. These four protocols provide user calendar information and preferences in different graduates: A protocol that provides full information to all agents, the approval protocol that shares only

the preference profile, the voting protocol where only free time can be shared and the suggestion protocol that shares only user proposals for the meeting. In their approach Wainer *et al.* assume that each involved user is using a different type of calendar system available to an agent that will schedule meetings for its user.

Full information protocol is used, when none of the participants has concerns about its personal calendar data and preferences. Each participants' agent sends their free time within the scheduling window and the corresponding preferences to the host. The coordinating host computes an intersection from all received times for all participants, and chooses the most favored time corresponding to the participants' preference information. If no intersection can be found, the participants are informed, that the meeting cannot be scheduled in the particular scheduling window.

Approval protocol In the approval protocol the coordinating host receives only the preferences for the scheduling time window from the attendees' agents. The host assumes that every participant is fully available and creates for every time interval an ordered list of the overall usefulness of the interval. Starting with the best interval, the host sends one time per round to the participants agents, which can accept or reject the proposal. The negotiation stops when all agents accept a time interval.

Voting protocol exchanges the free times of every participant. To schedule a meeting, every participant sends its free intervals within the scheduling window to the coordinating host. The host calculates sets of possible times from the intersection of the time intervals and sends these sets to the participants. Every agent ranks the set of intervals by its preferences and returns the sets back to the host. The host then approximates a preference function for each user from the rank of the sets and computes with these approximations the best overall time interval, which is returned as the scheduled time for the meeting. If no intersection can be found in the first step, the host informs the agents that the meeting cannot be scheduled.

Suggestion protocol is the protocol with the highest privacy. In this protocol whether information about free times nor preferences are exchanged. In every round of negotiation each agent must send a new time interval that the agent will accept as the time for the meeting. The coordinating host collects the suggestions, and sends the anonymized information to all agents. Each agent has full information about the suggestions made in previous rounds, and the agent takes the information into account, when it creates a new suggestion. If an interval in the list has been proposed by all agents, it is set as the time for the meeting. When more than one interval has been proposed by all agents, the host selects the earliest interval for the meeting. If all agents send a termination message in one round, the meeting cannot be scheduled.

Wainer *et al.* have pointed out that the full information protocol and the approval protocol are optimal. In the full information protocol the coordinating host has access to full calendar and preference information from all participants. The voting protocol and the suggestion protocol are complete but not optimal. In their work

Wainer *et al.* propose four theoretical protocols with different levels of privacy. It is easily conceivable to implement each of these protocols in applications for desktop and mobile devices. Therefore it can be assumed that the meetings in *Business Scenario 1* and *2* are able to be scheduled with the protocols as well as *Spare Time Scenario 1* introduced in section 1.4. Due to the different levels of privacy, it would be easy for business people working for different companies to negotiate meetings without exposing too much of their internal calendar data. The Suggestion protocol allows to schedule meetings in *Business Scenario 3* even if one of the companies has a high level of security concerning the calendar data of their employees. In their work Wainer *et al.* do not mention if the protocols are limited to schedule meetings only for human users, or if resources are supported. That is why it is unable to assess if the meetings in *Business Scenario 4* and *Spare Time Scenario 2* are able to negotiate.

The negotiation protocols presented by Chun *et al.* [8] and Chun and Wong [9] consist of iterations of proposing and counter proposing.

NWOPI Negotiation through relaxation — After being informed from the initiating agent about the start of the negotiation of a new meeting, every participating agent creates a list of possible time slots. The proposals are sorted according to the preference levels of the agent's user and stored in a preference vector. In the NWOPI algorithm the initiating agent chooses the "best first" times — the times with the highest preference level from the preference vector — as proposals to send to the participating agents. The participant agents determine whether any of the proposals are feasible or not. Then they select proposals with higher preference levels than the proposed by the initiator from their preference vectors and send them back to the initiator as counter proposals. If none of the proposals are acceptable and no counter proposals have been made by the participating agents, the negotiation process stops and the meeting is canceled. When all participants have accepted one proposal, the meeting is confirmed. If none of the proposals are accepted, the initiating agent selects the next proposals from its preference vector to start a new round of negotiation.

NWPI Negotiation based on relaxation plus preference estimation — is similar to the NWOPI algorithm above, but with the estimation of a global preference level. The initiating agent stores the proposals and counter proposals in a preference estimation vector and then selects the proposals that are more likely to be accepted. This should generate higher quality solutions. Therefore the participating agents send their preferences for proposal and counter proposal to the initiator. In an earlier work Chun and Wong [9] introduced a similar algorithm they called Nstar (N*).

N* In the first step the initiating agent generates a list of possible proposals, based on the initiator's calendar and preference model. The initiating agent sends the event data together with the proposed time slots to each participating agent. The participating agents create counter proposals according to the calendar and preference model of their user and return the counter proposals together with their preferences for proposals and counter proposals to the

initiator. The initiator evaluates all collected proposals and preferences, and stores them into the preference estimation vector. The preference estimation vector is dynamically sorted after every update. The initiating agent selects the next best proposals from the preference estimation vector and sends them to the participants. The participant agents evaluate each of the received proposals and returns a reply of either accept or reject. Additionally, they select proposals with higher preference levels than the proposed by the initiator from their preference vectors and send them back to the initiator as counter proposals. If none of the proposals is acceptable and no counter proposals have been made by the participating agents, the negotiation process stops and the meeting is canceled. When all participants have accepted one proposal, the meeting is confirmed. If none of the proposals are accepted, the initiating agent selects the next proposals from its preference vector to start a new round of negotiation.

As Chun and Wong mentioned the NWOPI algorithm is similar to the negotiation process defined in Sen and Durfee [42]. The preference estimation vector used for the NWPI algorithm is similar to the public preference model of Garrido and Sycara [16]. Chun and Wong introduced a scheduling agent system on the mobile framework MAFOA (mobile agents for office automation). This system is using two types of agents to schedule meetings, a Secretary Agent representing one attendee and a Meeting Agent acting as coordinator. In the second work of Chun *et al.* [8] they named the agents Initiating and Collaborating Agent. As illustrated in section 2.1.2, both agents (independent from the naming) are acting on the behalf of human users. The Secretary or Collaborating Agent is not capable to negotiate meetings for resources, like in *Business Scenario 4* and *Spare Time Scenario 2*. Nevertheless, the agent based structure and the negotiation with strong privacy regarding the calendar data of the users, would allow to negotiate meetings company overlapping as in *Business Scenario 3*.

Tsuruta and Shintani [46] formalized "Distributed Valued Constraint Satisfaction Algorithm" an algorithm to solve over-constrained meeting scheduling problems for agent based systems. When no solution can be found during a negotiation process due to too many constraints, the algorithm tries to reach a consensus by relaxing a portion of all constraints, but still satisfying as many of the important constraints as possible. In the first step of the algorithm the initiating agent generates a set of windows, in which the meeting may be scheduled, based on the constraints for the starting and ending time. Together with the information about the meeting and an initial threshold of weights of constraint (integer between 1 and 9), the generated set of proposals is sent to all participant agents. Each participating agent returns a replay containing impossible times for the meeting. The constraints weight of the excluded times is higher than the weight of the initial proposals. When the coordinating agent receives all participants' replies, it uses a tree search — a modified version of forward checking introduced by Haralick and Elliot [20] — to find a time interval satisfying the constraints of all participants. If a solution cannot be found, the next round of negotiation starts and the coordinating agent sends a new message containing a new set of proposals and its threshold of weights to the participating agents. In every round the threshold is increased by 1. If

the new threshold reaches 10, the coordinating agent cancels the meeting and a failure message is sent to all participants. If the tree search delivers a solution, the coordinating agent sends the result to the participants. When more than one solution is found, the proposing user selects what time to use.

As for many algorithms using weights to define user preferences and constraints, the approach of Tsuruta and Shintani can be manipulated by an agent. If, for example, an agent wants to assert its preferred time, it can declare all times, except of its favorite time slot, as impossible and set the weight of the constraint to the highest level. And if there would be only manipulating agents, which like to assert their preferred times, the algorithm would be as efficient as without the threshold of weights.

Tsuruta and Shintani have implemented a group schedule management system, which can support personal schedule management, group sharing calendars, and meeting scheduling based on the algorithm described above. With this desktop-based system it would be difficult to schedule one of the mobile scheduling scenarios introduced in section 1.4. Therefore, it is safe to say that the meetings in the *Business Scenario 2* as well as in *Spare Time Scenario 1* are unable to negotiate. Tsuruta and Shintani do not mention, if their application and protocol does support other than human users to be scheduled, resources like meeting rooms. It has to be assumed, that the meeting in the *Business Scenario 4* and the meeting in the *Spare Time Scenario 2* are not negotiable with their system. Furthermore, the group sharing application is mainly built to support closed user groups, company spanning negotiation of meetings seems not to be able, which eliminates *Business Scenario 2*.

2.4 Scheduling Protocol

This section takes a closer look at the technical protocols used in different current approaches to negotiate meetings. The approaches described in the sections before use following types of protocols:

- Electronic Mail (email)
- File Transfer Protocol (FTP)
- Hypertext Transfer Protocol (HTTP)
- Extensible Messaging and Presence Protocol (XMPP)
- Proprietary protocols

Due to the wide spread availability since the early days, most approaches are using electronic mail to negotiate meetings. Newer approaches also utilize other protocols as FTP or the messaging protocol XMPP, originally designed for chat system. Commercial applications often use own proprietary protocols.

2.4.1 Electronic Mail

Electronic mail (email) is very popular to negotiate meetings due to the wide spread availability. To let agents negotiate appropriate times to meet over email is probably the most human like type to schedule meetings.

Sen *et al.* [43], Haynes *et al.* [22] use email to let the agents communicate with each other. They implemented a message constructor/decoder component that serves as the interface with the email system. In their approach the agents negotiate the time of a meeting based on bids and proposals. Every time a proposal is to be sent, the messaging module constructs a mail message with a specially formatted subject and body, and invokes the mailing software to send out the email to the recipient agents. The constructor/decoder module also observes continually the system mail box to check if a meeting scheduling message has arrived. To identify the meeting messages, Sen *et al.*, Haynes *et al.* use special header strings. The subject starts with "*MS*" followed by the login ID of the user originating the meeting request and a time stamp, representing the time the request was made. The information in the body is formatted as "*meta data* : *data*" pairs separated by a colon. The body contains the type of message, the message direction, the meeting ID to assign the messages and an internal iteration counter. Furthermore, the message contains the ID of the user initialized the meeting, the meeting topic and a comma separated list of the invitees to the meeting. Finally, the message also contains preferred time intervals being proposed for the meeting, the expected duration of the meeting, a suggested priority of the meeting, the deadline by which a decision on this meeting has to be made and the set of acceptable time intervals for this meeting.

To gain information about the availability of its users, the PTIME agents in the CALO system introduced by Berry *et al.* [4] are communicating peer-to-peer by email. For next deployment Berry *et al.* have promised to offer email communication peer-to-peer as well as communication by email with the users.

In the approach of Brzozowski *et al.* [6] communication over email is used in another form: They have implemented a web-based service called groupTime to visualize group calendars and negotiate meetings in an educational environment. When a new meeting is initialized, each invitee receives an inviting email, containing a link to respond. Brzozowski *et al.* have implemented a simple agent-user-information protocol. The big part of the negotiation work is done by the groupTime web-interface.

Faulring and Myers [14] use email messages to gain information about pending meetings. They have presented RhaiCAL (Rich Human-Agent Interaction for Calendar Scheduling Agents), a system where agents watch for emails containing meeting scheduling information and try to assist users to schedule these meetings. When an agent identifies a message containing meeting scheduling information, the user is asked to check the understandings of the agent, then the agent proposes actions to help the user to schedule the meeting.

Kozierok and Maes [28] proposed an agent that communicates with other agents and users as well by semistructured email messages. When a new meeting is initiated, the agent will send an invitation message to the calendar mailboxes of the invited users. The messages can either be handled by the agent or by the user.

Modi *et al.* [37] implemented a personal assistant agent for calendar management

called CMRadar. The multi agent interaction in calendar meeting scheduling occurs by email message exchange. Modi *et al.* defined grammar rules for converting emails to a normalized format representing the meeting request or reply to a request. An extractor module is parsing incoming email messages into these templates. This extractor can interpret messages from other agents or messages from users in natural language.

2.4.2 File Transfer Protocol

To augment the information in shared personal calendars, Tullio *et al.* [47] have implemented an open model groupware calendaring system called Augur. Augur is not really a typical scheduling system as it was designed to visualize the attendance likelihood and events co-scheduled by the users colleagues. The calendar data is hosted on mobile Personal Digital Assistants (PDAs) based on Palm OS [40]. Every time the calendar is updated, the information is automatically synchronized with a network computer using the File Transfer Protocol (FTP). A parsing module on the networked computer reads the calendar information and updates new events with a database. The event matching module is looking for equal events scheduled by colleagues. A Bayesian network prediction module augments the data with information about attendance likelihood. Every user has a copy of the Bayesian network capable of learning their attendance habits over time on their mobile devices. To visualize the likelihood of attendance Tullio *et al.* use different types of visualization techniques. The augmented calendar information is accessible to every user by a separate web-service. Augur is not a meeting negotiation system in the proper sense, the idea of Tullio *et al.* is to support interpersonal communication via predictive models, intelligent text processing and visualizations. The File Transfer Protocol is used to collect all relevant data from distributed mobile devices.

2.4.3 Hypertext Transfer Protocol

Augur, the approach of Tullio *et al.* [47] presented in the section above, uses different visualization techniques to illustrate the likelihood of attendance on events. The calendar information — collected in a centralized database over FTP and augmented by a matching module and a Bayesian network — is accessible to every user by a web-service based on Java Server Pages (JSP) and dynamic HTML (DHTML). With different visualization techniques Augur presents the user's events of the day or the week and the likelihood of colleagues' attendance to the meetings. Within his daily calendar view, the user is able to identify fellows who co-scheduled an event. As mentioned above, Augur is not used to negotiate meetings. It uses HTTP to visualize a user's calendar and the likelihood of attendance to events co-scheduled by colleagues.

2.4.4 Extensible Messaging and Presence Protocol

The Extensible Messaging and Presence Protocol (XMPP) is used by Tungle [48], a new approach of a "Free/Busy" calculation agent. Tungle is a plug-in for Microsoft Outlook to share personal calendars between individuals and calendaring applications. The schedules of the invited users are displayed side by side and the initiating

user chooses one of the empty time slots, where every participant is available to schedule the meeting. The new created event is transported over XMPP to the Tungle clients of the invited users. The participants' clients automatically add the new event to their calendars.

2.4.5 Proprietary Protocols

Most of the commercial systems communicate and negotiate events and meetings through own proprietary protocols. Some of the businesses keep the used protocols secret, therefore it is not easy to gather information about the underlying concepts. To communicate with the shared calendars of other users, Microsoft Outlook [34] clients exchange information over the Microsoft Messaging Application Programming Interface (MAPI) [35]. While scheduling meetings, Outlook retrieves the calendar information of each user, invited to a meeting, from the central Exchange Server. As mentioned in section 2.2.1 Outlook provides a simple "free/busy" calculation by sharing the calendar information of the involved users. The schedules of each invited user is visualized side by side to the others and the initiating user can choose one of the empty time slots as proposal for the meeting. Finally, the invitations for the meeting are sent to each user by email.

3 Research Question: Enabling Meeting Scheduling for Mobile Devices

In this chapter the result of the analysis of the approaches described in chapter 2 will be summarized and discussed. In the first part the approaches again are examined for their usability for mobile devices. In the second part the result of the Business Scenario and Spare Time Scenario evaluation are summarized. And finally the discussion in part three will show that none of the current approaches are fully able support meeting scheduling on mobile devices.

3.1 Usability for mobile devices

Many of the current approaches, discussed in detail in the previous sections, are already implemented and exist as real applications. Some of these applications have also been tested on mobile devices. In this section the usability of approaches for mobile devices will be discussed.

One application designed to be used on mobile devices was introduced by Mynatt and Tullio [39]. They implemented a system to visualize the attendance likelihood and events co-scheduled by the users colleagues. The calendar data is hosted on mobile Personal Digital Assistants (PDAs) based on Palm OS [40]. Every time the calendar is updated, the information is automatically synchronized with a network computer using the File Transfer Protocol (FTP). A parsing module on the centralized network computer reads the calendar information and updates new events with a database. The event matching module is looking for equal events scheduled by colleagues and a Bayesian network prediction module augments the data with information about attendance likelihood. Every user has a copy of the Bayesian network, capable of learning their attendance habits over time on their mobile devices. To visualize the likelihood of attendance, Tullio *et al.* use different types of visualization techniques. The augmented calendar information is accessible to every user by a separate web-service.

In May 2008 Tungle [48] announced to provide its scheduler service on Smart Phones[1]. Tungle is a 'Free/Busy" calculation agent, and can be integrated into Microsoft Outlook as plug-in to share personal calendars between individuals and calendaring applications. The meetings are scheduled similar to Microsoft Outlook, but — different to Outlook — the calendar data of the invitees in Tungle is exchanged through XMPP.

The system introduced by Demirel and Erdogan [12] is implemented on the Java Agent Development framework (JADE) [2]. Their approach has not primarily been designed for mobile devices. The JADE framework — an open source platform for

[1] http://www.tungle.com/Home/svc/Press, *Mai 2008*

	Business Scenario 1	Business Scenario 2	Business Scenario 3	Business Scenario 4
Microsoft Outlook [34]	yes	yes [1]	no	no
Crawford and Veloso [10]	yes	yes	no	no
Tullio *et al.* [47]	yes	yes	no	no
Sen and Durfee [41]	yes	yes	yes	no
Berry *et al.* [4]	yes	no	yes	no
Chun *et al.* [8]	yes	yes	yes	no
Hassine and Ho [21]	yes	unknown	yes	no
Franzin *et al.* [15]	unknown	unknown	unknown	unknown
Demirel and Erdogan [12]	yes	no	yes	yes
Brzozowski *et al.* [6]	yes	yes [2]	yes	no
Kozierok and Maes [28]	yes	unknown	unknown	no
Lin and Hsu [30]	yes [3]	yes [3]	yes [3]	no
Mitchell *et al.* [36]	yes	no	yes	no
Wainer *et al.* [49]	yes	yes	yes	no
Tsuruta and Shintani [46]	yes	no	no	no

[1] via desktop version
[2] via web access
[3] system not implemented

Table 3.1: Evaluation of the Business Scenarios

peer-to-peer agent based applications — is able to run on mobile systems. Lima *et al.* [29] showed patterns to implement mobile agent-based applications using the JADE framework.

Most of the concepts discussed in chapter 2 are already deployed as applications. Except for the works analyzed above, none of the other concepts are designed to be used on mobile systems. Either they are implemented for workstations [43, 22, 15] or they are accessible through web interfaces [6]. Other approaches require a large set of services [4, 5, 3, 17] to negotiate meetings. Currently no agent based application exists to schedule meetings fully on mobile system.

3.2 Evaluating Scenarios

As outlined in section 2.1, the current scheduling approaches can be divided into two categories: the "Shared Personal Calendars" and "Agent-based" meeting scheduling. The approaches of both types are described in detail in the sections 2.1.1 and 2.1.2.

3.2.1 Business Scenarios

Table 3.1 shows the summary of the Business Scenario negotiability of the approaches illustrated in the chapter 2. In combination with the Microsoft Exchange Server [31] Outlook allows people to schedule meetings with other people from the same company. Furthermore, it is possible to include resources, required for the meeting. This allows Outlook to negotiate the meetings in *Business Scenario 1.*

As discussed in chapter 2 Microsoft follows its own strategy regarding accessibility of their groupware solution: The majority of the Outlook calendaring and meeting negotiation functions work only in combination with a centralized Exchange Server. This mailing and groupware tool allows company persons to move outside to access their calendar data by a web interface. Moreover, the so called Outlook Web Access (OWA) offers the full Outlook features for Internet Explorer 6 or higher. Other web browsers have limited functions. If a traveling sales person — like in *Business Scenario 2* — has web access and has an Internet Explorer web browser, he or she would be able to participate in the negotiation of a meeting, even if Outlook is not available. Outlook is a typical "shared calendar" system, where the inviting person must have access to the calendars of the meeting attendees. What is a desired feature within one company, could become an unwanted security problem in an enterprise overlapping setting. The Exchange Server is able to provide domain traversing calendar access, but for many companies the distribution of internal calendar data is still a strict no-go area. As a result *Business Scenario 3* and *Business Scenario 4* are difficult to schedule with shared calendar approaches like Microsoft's Outlook.

Designed to reflect the shared calendar approach of Outlook, the protocol introduced from Crawford and Veloso faces the same restrictions: No company overlapping negotiation and limited access for people outside the company. It is safe to assume that *Business Scenario 1* and *2* can be scheduled, whereas *Business Scenario 3* and *4* cannot be negotiated with the Open-Negotiator.

The Augur system introduced from Tullio *et al.* is the only approach in the evaluation that allows to schedule meetings on mobile devices. Therefore it is capable to negotiate the meetings in *Business Scenario 1* and *Business Scenario 2*. Nevertheless, Augur is a shared calendar system, that makes it difficult to schedule the meetings in *Business Scenario 3* and *Business Scenario 4*.

In their approach Sen and Durfee provide several strategies for different meeting negotiation strategies, that allow to schedule the meetings in *Business Scenario 1*, *2* and *Business Scenario 3*. Each agent is assigned to a specific user, Sen and Durfee do not consider resources. Therefore the *Business Scenario 4* cannot be negotiated.

Another agent-based system, focused to act on the behalf of human users, was introduced by Berry *et al.*. With these agents it is not possible to negotiate the meeting in *Business Scenario 4*. Furthermore, the tremendous requirements of this system do not allow to transfer the framework to run on small mobile devices. Consequently the meetings in *Business Scenario 2* cannot be negotiated either. The agents transmit the calendar data by email. Assumed the security specification of the companies allow the transfer of internal calendar data, it would be possible to schedule meetings with people from different companies like in *Business Scenario 3*.

Chun *et al.* have introduced two types of agents, both acting for human users. None of the agents is capable to negotiate meetings for resources like in *Business Scenario 4*. In the approach Chun *et al.* emphasize strong privacy regarding the exchanged calendar data of the users. Combined with the agent-based setting, it would be possible to negotiate meetings with people from different companies as in *Business Scenario 3*, even if the security and security level in one company is very high. Although Chun *et al.* do not provide information about requirements of the technical infrastructure of the system, it is likely to adopt, that *Business*

Scenario 1 and *2* are negotiable.

The multi-agent design of the approach of Hassine and Ho allows to negotiate the meetings in *Business Scenario 1* and *Business Scenario 3*. The agents are formalized only to act on the behalf of human users, which disallows to schedule the meeting in *Business Scenario 4*. Hassine and Ho do not give information about the technical requirements of the system, therefore it is difficult to assess whether or not the approach could be integrated on mobile devices, as the meeting in *Business Scenario 2*.

Franzin *et al.* extend a existing meeting scheduling system with preferences. In their work they do not describe the agents used in detail. Therefore it is unable to gauge, which of the business scenarios are possible to negotiate.

In their work Demirel and Erdogan introduced an additional agent type to integrate resources (specially locations) into meeting negotiations. Furthermore, the multi-agent framework allows to schedule the meetings in *Business Scenario 1*, *2* and *Business Scenario 4*. Although the JADE framework, the system of Demirel and Erdogan was built on, is able to run on mobile systems [29], no implementation for portable devices exist, which eliminates the meeting in *Business Scenario 2*.

"groupTime", the web-based group scheduling system, introduced from Brzozowski *et al.*, is mainly built to assist people working in academical environments. The open approach enables to schedule meetings with attendees working for different organizations or companies. Hence, "groupTime" is able to schedule the meeting in the *Business Scenarios 1* as well as the meeting in *Business Scenario 3*. The web-based setting allows traveling sales persons to connect with their mobile devices and participate in the negotiation of the meeting in *Business Scenario 2*. But"groupTime" does not support the scheduling with resources. Therefore it is impossible to schedule the meetings in *Business Scenario 4*.

In their work Kozierok and Maes do not consider resources for meeting negotiation. They are focused on how to make meeting scheduling more efficient considering users habits and preferences. For that reason the meetings in *Business Scenario 3* and *4* cannot be negotiated.

The agent sketched from Lin and Hsu was proposed to work with different calendar scheduling software to automate the scheduling process. This would allow this agent to negotiate meetings with calendar data from heterogeneous systems like in the *Business Scenario 3*. At present the approach is not implemented as prototype, but as for all agent-based systems, it should be able for a future implementation to schedule the meeting in the *Business Scenario 1*. *Business Scenario 2* probably could be negotiated with the future agent, if the requirements of the system are low enough to run on mobile devices. Lin and Hsu do not mention the possibility to consider resources within the meeting scheduling process. Therefore, it is unlikely, that the *Business Scenario 4* is negotiable with this agent.

The agents in the work of Mitchell *et al.* use the well established email protocol to communicate and negotiate meetings. This widely spread message protocol allows the agents to schedule meetings across organizational boarders, as in *Business Scenario 3*. At present the approach is implemented as a desktop-based system, no scheduling for mobile devices like in *Business Scenario 2* is supported. Furthermore, the agents introduced from Mitchell *et al.* act on the behalf of human users, references cannot be integrated into the scheduling process, which would make it

	Spare Time Scenario 1	Spare Time Scenario 2
Microsoft Outlook [34]	yes [1]	no
Crawford and Veloso [10]	yes	no
Tullio *et al.* [47]	yes	no
Sen and Durfee [41]	yes	no
Berry *et al.* [4]	no	no
Chun *et al.* [8]	yes	no
Hassine and Ho [21]	unknown	no
Franzin *et al.* [15]	unknown	unknown
Demirel and Erdogan [12]	no	no
Brzozowski *et al.* [6]	yes [2]	no
Kozierok and Maes [28]	unknown	no
Lin and Hsu [30]	yes	no
Mitchell *et al.* [36]	no	no
Wainer *et al.* [49]	yes	no
Tsuruta and Shintani [46]	no	no

[1] via desktop version
[2] via web access

Table 3.2: Evaluation of the Spare Time Scenarios

impossible to negotiate the meeting in *Business Scenario 4*.

Wainer *et al.* introduced four theoretical protocols to negotiate meetings. The different privacy levels of the protocols allow to schedule the meetings in *Business Scenario 1*, *2* and the meeting in the company overlapping *Business Scenario 3*. In their work Wainer *et al.* do not mention, if resources are supported by their protocol. Therefore the with this approach it is impossible to negotiate the meeting in *Business Scenario 4*.

The desktop-based schedule management and group sharing system, introduced from Tsuruta and Shintani, is unable to negotiate the meeting in the *Business Scenario 2*. Tsuruta and Shintani do not mention if their application and protocol supports resources to be scheduled. Therefore it has to be assumed, that the meeting in *Business Scenario 4* is not negotiable with their system. The group sharing application is mainly built to support closed user groups. Company spanning negotiation of meetings seems not to be able, which eliminates *Business Scenario 2*.

3.2.2 Spare Time Scenarios

Table 3.2 shows the negotiability of the Spare Time Scenario meetings with the approaches illustrated in chapter 2.

Unlike companies, private persons possibly have less privacy doubt to share their personal calendar data, particularly when sharing personal calendars with friends. Therefore Outlook could be used to schedule meetings for *Spare Time Scenario 1* and *Spare Time Scenario 2*, assuming the participating friends do have their desktop version of Outlook with them. Microsoft offers a lightweight mobile version of Outlook [32], but the mobile version is limited in functions and does not contain

the meeting scheduling function of the desktop version.

The protocol of Crawford and Veloso is designed to reflect the open calendar style approach of Microsoft Outlook Microsoft Corporation [34] with the same restrictions for scheduling meetings. Therefore it can be assumed that *Spare Time Scenario 1* is able to be scheduled whereas *Spare Time Scenario 2* cannot be negotiated with the Open-Negotiator.

Augur, the meeting scheduling system introduced by Mynatt and Tullio is able to run on desktop computers and mobile devices. Therefore, Augur is capable to schedule the meetings in *Spare Time Scenario 1*, but it does not support the scheduling of resources, which disallows negotiate the meeting in *Spare Time Scenario 2*.

In the approach of Sen and Durfee [41] each agent is assigned to a specific user, resources are not considered to be scheduled. Their approach is unable to negotiate the meetings in the *Spare Time Scenario 2*.

Another agent-based system, designed to act only on the behalf of humans, was introduced by Berry *et al.*. It is safe to assume, that the *Spare Time Scenario 2* cannot be negotiated. Furthermore, the system requires a laptop computer at minimum, which makes it impossible to transfer the framework to run on small mobile devices. Therefore, the meeting in the *Spare Time Scenario 1* cannot be negotiated either.

The two agents introduced by Chun *et al.* both act only on the behalf of human user. The Secretary Agent is not capable to negotiate meetings for resources like in *Spare Time Scenario 2*. At present the theoretical protocol is not implemented. Although Chun *et al.* do not provide information about the requirements of the technical infrastructure needed to implement the approach, it is likely to adopt that the meeting in *Spare Time Scenario 1* is negotiable with.

The agents formalized by Hassine and Ho negotiate meetings on the behalf of its human user, but not for resources. However, with some smaller changes the Interface Agent could be capable to negotiate meetings for resources, like in the *Spare Time Scenario 2*. Without any information about the technical requirements of the system, it is difficult to assess whether or not the approach could be integrated on mobile devices. For that reason it is impossible to assess if the system is able to schedule the meetings of the *Spare Time Scenario 1*.

In their work Franzin *et al.* have not described the agents in detail, which makes it difficult to gauge, if one of the spare time scenarios can be negotiated or not.

Although the system of Demirel and Erdogan is not primary designed for mobile devices, the JADE framework is runnable on mobile systems. With some undefined changes it should be possible to transfer the personal and the location agent on portable devices. But in the current implementation no mobile devices are supported. Therefore the meetings in the *Spare Time Scenario 1* and *2* are not negotiable.

Mobile devices are able to connect straightforward to the web-based group scheduling system of Brzozowski *et al.*. Therefore, the friends on the road in *Spare Time Scenario 1* are able to negotiate their meeting. On the other hand, the system is primarily designed to schedule meetings between human users. “groupTime” does not support the scheduling with resources, e.g. meeting or sport places. Therefore it is not possible to schedule the meeting in *Spare Time Scenario 2*.

Kozierok and Maes have focused on how to make meeting scheduling more effi-

cient considering users habits and preferences. They have not considered resources for meeting negotiation. For that reason the meeting in the *Spare Time Scenario 2* cannot be negotiated with this approach.

Lin and Hsu introduced a theoretical agent to learn a user's scheduling criteria. Even no implementation of the approach exist at present, the protocol is able to negotiate the meeting in *Spare Time Scenario 1*. Lin and Hsu do not mention the possibility to consider resources within the meeting scheduling process. Therefore it is unlikely that *Spare Time Scenario 2* is negotiable with this agent.

The agent introduced by Mitchell *et al.* [36] is implemented as a desktop-based system, which would prevent to schedule the meetings with participants outside the home like in *Spare Time Scenario 1*. Furthermore, with this agents, references cannot be integrated into the scheduling process. Therefore, the meeting in *Spare Time Scenario 2* is unable to negotiate.

The four theoretical protocols introduced by Wainer *et al.* are predestined to be implemented in applications for mobile devices. Therefore, it can be assumed that the meeting in *Spare Time Scenario 1* is able to be scheduled with the protocols. Wainer *et al.* do not mention if the protocols are limited to schedule meetings only for human users, or if resources are supported. Hence, it is unable to assess, if the meeting in *Spare Time Scenario 2* is able to negotiate.

The desktop-based group schedule management system from Tsuruta and Shintani is unable to schedule one of the mobile scheduling scenarios. Therefore, it can be assumed that both meetings in *Spare Time Scenario 1* and *Spare Time Scenario 2* are unable to negotiate. Furthermore, Tsuruta and Shintani do not mention, if their application and protocol does support other than human users to be scheduled, like the sports place in *Spare Time Scenario 2*.

3.3 Enabling Meeting Scheduling for Mobile Devices

The evaluation of the current approaches in the section before has shown that none of the systems is capable to schedule each meeting of the six scheduling scenarios defined in section 1.4. Furthermore, none current application is able to schedule the meeting in *Spare Time Scenario 2*, where some friends want to meet to go in for sports. The difficulty in this set-up is to consider the external sports place into the meeting negotiation. With some approaches it would be possible to integrate the resource "sports place" into the negotiation, but these applications (Outlook [34], Demirel and Erdogan [12]) are not runnable on mobile devices like mobile phones or PDAs (Personal Digital Assistants). Except to Tullio *et al.* [47] the current approaches have fully disregarded the increasing need of meeting attendees to be unbound to fixed desktop systems. None of the approaches analyzed in chapter 2 is designed to accomplish the vital need to be used on mobile devices. Furthermore, an approach specially designed to schedule meetings on mobile devices, should be able to perform every Business Scenario as well as each Spare Time Scenario. Therefore, it is necessary for this system to be accessible independent from the organizational structure the attendees belong to, i.e., a meeting should be negotiable although the participants are working for different companies. The evaluation of the current approaches has indicated that shared calendar

systems are inappropriate for organization overlapping meeting scheduling. Agent-based systems have shown much better performance in scheduling meetings in a distributed setting. Furthermore, small independent agents seem to be the best approach to support meeting scheduling for mobile devices. Attendees of different enterprises perhaps do not feel comfortable, if a system exchanges to much of their private or company internal calendar data. Hence, the calendar data exchanged between the participants should be as private as possible but public enough as necessary to schedule the meeting fast and efficient. Furthermore, to consider the preferences of all attendees, the exchanged calendar data has to be enriched with preference information in different levels. The agent-based system should communicate through a wide spread and easy adaptable protocol. To use a well known protocol has big advantages: Most firewalls have standardized roles for this protocols, which securely allow the required data to pass. Furthermore, it is easy for IT administrators to configure their firewalls to only transfer needed information. Current approaches use email [43, 22, 6, 14, 28, 37], FTP (File Transfer Protocol) [47], HTTP (Hyper Transfer Protocol) [47, 6] and proprietary protocols [34]. Email, FTP and HTTP are established protocols for many years, but they are designed to support the communication between humans, transfer files or — in case of HTTP — to present content for the Internet. Each of these protocols would have to be adopted heavily to make the agents communicate properly. In 2004 the Internet Engineering Task Force formalized the communication protocol for messaging and presence called XMPP (eXtensible Messaging and Presence Protocol) as Proposed Standard [26]. Initially design for instant messaging between humans, the protocol is also applicable for automated data exchange. It allows near-real-time transfer of nearly every type of data. Tungle Corporation [48] was one of the first meeting scheduling support systems using XMPP. In the meantime Tungle has reconsidered its operating area and has changed the system architecture to a web service. To schedule the meeting considering the conference room in *Business Scenario 4* and the meeting taking the sport place into account in *Spare Time Scenario 2*, it is necessary for the system to depict resources. The agents should act on behalf of human users and of meeting relevant resources of any type. To include as many users as needed, the agents should be runnable platform independent and even on heterogeneous systems. Mobile devices do have special requirements. Due to their size, the user interface needs to be adjusted, the agents should be assessable with a few finger entries, elaborate input should be avoided. On the other side, the application should be powerful enough to schedule each of the meetings illustrated in Business Scenarios as well as the meetings in Spare Time Scenarios. In the next chapters a new agent-based system specially designed to enable meeting scheduling for mobile devices will be introduced. First, the protocol based on the extensible messaging and presence protocol (XMPP), is formalized. Therefore the XMPP protocol has to be extended with particular meeting scheduling and coordination messages. Chapter 5 suggests a meeting scheduling prototype for the new platform for mobile devices called Android [45]. This prototype is implementing the scheduling protocol and tries to meet all requirements determined above.

4 Introduction of a Mobile Scheduling Protocol

In this chapter a protocol, specially designed to support meeting scheduling on mobile devices, will be introduced. In the first part the whole negotiation process — from the initial request, to the final result — will be defined. Furthermore, it will be illustrated how the different agents are orchestrated to interact with each other, while negotiating a meeting time. The second part of this chapter will take a closer look at the information integrated in this approach to schedule meetings. The data required for the negotiation is exchanged as properties in XMPP messages. To get the agents communicating with each other, the XMPP protocol has to be extended with special scheduling messages. In the third part, these messages required to negotiate meetings, are specified. In the last part, the formal algorithm of the time slot negotiation will be defined.

4.1 Orchestration of the Scheduling Process

Figure 4.1 shows the orchestration of the scheduling process between the initiating agent (also acting as participant), the coordinating agent and one participant agent. The shown process is only prototypical and can be enlarged as desired. The participant on the left is the initiator of the scheduling process. The initiating agent on the left wants to schedule a meeting with the participant on the right, while the agent in the middle is coordinating the negotiation. As shown later in this chapter the coordinating role can be held either by the initiating agent, one participant, or any other agent (a specialized coordination service agent for example). In Figure 4.1 a separate agent is coordinating the meeting negotiation.

- The initiator records all the required data discussed in chapter 4.3, like *time horizon* (e.g. *this week*, *next week*, *in the next two weeks*, etc.), *duration* and *location* of the meeting and the user names of the participants. After entering the data, the initiator starts the scheduling process. The initiators' agent sends a coordination request to the coordinating agent.
- When the coordinating agent accepts the coordination of the meeting, it sends a timetable query request to all participants' agents.
- The participants' agents now calculate the times of availability of their users. Therefore, the agents contact their calendar services to retrieve the necessary availability data for the requested time horizon. With these data, the agents calculate a weighted timetable, depending on the preferences of their user.
- After receiving the times from the participant agents, the coordinating agent matches the timetables and calculates possible time slots for the meeting. If

no time slot can be found, the process stops with an error and the initiator can decide to expand the time horizon for the meeting, e.g. by another week, or cancel the negotiation process.

- When the previous step has generated one or more possible time slot to meet, the times are presented to the participants and they can estimate these time slots. The preferences chosen are matched again by the coordinator. When there are no time slots left, because the users did not like one of the time slots or they prefer different time slots, the initiator gets an error message and he can decide to cancel the process or start all over by choosing another planing horizon.
- When one or more time slots are left, it is the coordinator's decision to determine the time of the meeting. The participants again have the possibility to decide whether they accept or deny the final result.
- When one of the participants deny the final meeting time, the initiator gets an error message and he can decide to cancel the process, or start all over choosing another planing horizon.
- After all participants have accepted the time, the meeting is fixed and the users can add the meeting to their calendars.

4.2 Scheduling States

To negotiate a meeting, the scheduling process runs through different states. When a higher state is reached, the state can be changed back to every lower state. If, for example, the scheduler is in the *Waiting for preferences* state and receives in the meantime a new timetable from a participant's agent, the state is set back to *Calculate time slots* to recalculate the best suitable time slot. Following states are possible:

Idle The scheduling process is idle and ready to be initialized to schedule an event. This is the initial state where the scheduler is instantiated and waits to receive new event data to negotiate.

Initialized After the user has recorded all necessary data for a new meeting to schedule, the state of the scheduler changes to initialized. In this state the data of the meeting and the timetable requests are sent to the participants.

Waiting for timetables The process is waiting for the requested timetables of all invited users. Each participant's agent contact its users calendar services and calculate by integrating different information and user preferences, a weighted timetable for its users. The weight of a time slot can be a floating point value between -1 and 1, where -1 means the user can't make that time and 1 means, it is the best choice for the user. The values are normalized from the coordinating agent, when the collected timetables are aggregated to protect the scheduling process from manipulation.

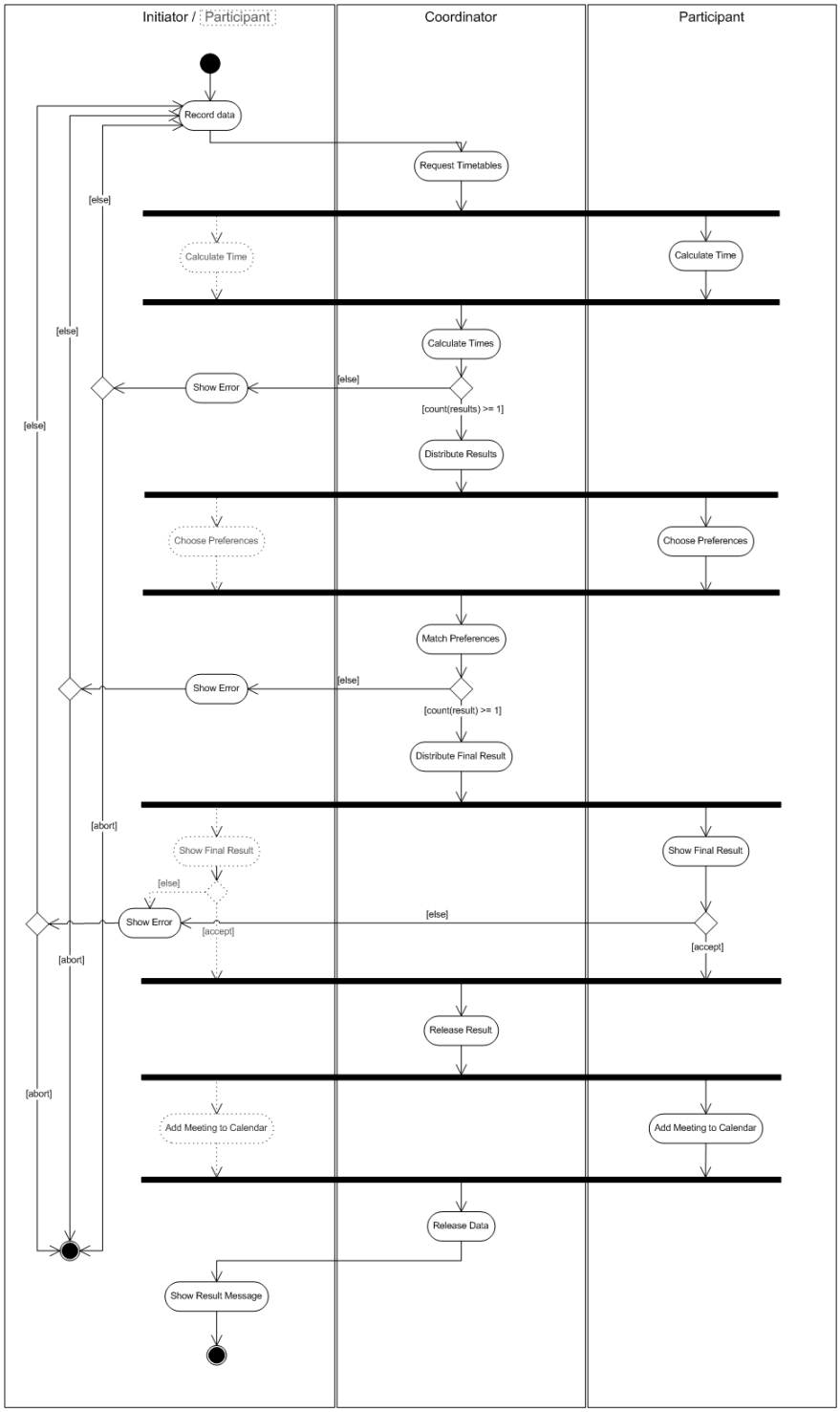

Figure 4.1: Orchestration of the negotiation

Calculate time slots The time slots of the participants are matched and the best times are calculated. After the coordinating agent has received the weighted timetables from each participant's agent, it starts to match the timetables and calculates the most apposite time slots for all participants. Therefore, the time horizon is split into adjustable time intervals. Next the weight for each interval is calculated by counting the total of each participant's weight for the interval. Then, the intervals are connected again to extract the most favorable time slot for all invitees.

Distribute time slots The calculated time slots are distributed to all participants. An adjustable number of time slots is sent to each participant. The participants are asked to choose their preferred time slot.

Waiting for preferences The process waits to receive the preferred time slots of the participants. Each user can set his or her preferences to the best time slots computed in the *Calculate time slots* state. As defined by Brzozowski *et al.* [6], each participant can estimate the preference of each time slot from *Perfect*, *Ok*, *Rather Not* and *No Chance*. The time slots are marked as *Ok* by default. Brzozowski *et al.* chose a four point rating scale because it offers good precision and forces users to take a stand since there is no neutral choice.

Match preferences All participant's preferences are matched and the best time will be selected as final result. After the participants have defined their preferences, their agents send the preferences back to the coordinating agent. When the coordinating agent has received the preferences from each user, the total of all time slots is calculated and the best first time slot will be sent as final result to all participants.

Distribute result The result is distributed to all participants. The final time slot computed in *Match preferences* is sent to each participant's agent.

Waiting result The scheduling process waits for the decisions of the participants to accept or reject the final result. Each participant can choose to accept, if he or she feels comfortable with the final time for the meeting, or reject the result, if the time is not feasible. The choice is sent back to the coordinating agent.

Release result When all participants have accepted the scheduling result, the event is released and a message is sent to the participant's agents to add the event to their user's calendars. If one or more participants have rejected the result, the initiating user receives a message and can choose to restart the negotiation process again with modified parameters or stop the negotiation.

Add result Adding the scheduling result to the calendar. Each participant's agent adds automatically the new event to the participant's calendar.

Release data Finalize the process and destroy data. After the event is added to each participants calendar, the negotiation process stops and the collected data will be destroyed.

4.3 Integration of Data

In this section the data integrated into the negotiation process will be described. In the first part, general principles are specified, i.e., actors involved during the negotiation of a meeting and their rules. Next the information exchanged between the actors is characterized, i.e., the data of the meeting, the predefined time horizon values and possible user preferences.

As mentioned in section 1.1.1 different types of actors take part, while scheduling a meeting. This protocol differs three types of actors:

Initiator is the person that hosts the meeting and invites the required people. The initiator is responsible for the meeting and the negotiation of the best feasible time. He or she decides where the meeting is held and who to invite. Moreover, the initiator acts as a kind of supervisor in cases of error during the scheduling process, e.g., he or she decides how to proceed, when no time slot can be found in the *Calculate time slots* state, or when one or more participants reject the final result. The initiator can also be the coordinator and / or a participant of the meeting. Only the initiator can cancel the meeting scheduling process.

Coordinator The host who technically coordinates the scheduling of the meeting. This is where all messages and responses intersect during the scheduling process. The coordinating agent requests the timetables from the participants, calculates the best fitting time slots, distributes the calculation result, matches the preferences and instructs the participants agents to add the final result to their users calendars. Furthermore, the coordinating agent keeps the initiator up-to-date by sending status messages and answering status requests.

Participant An invitee, that need to attend the meeting. For each meeting there can be on or more participants. The coordinator and the initiator can be participants as well. Participants are classified as *Mandatory important*, *Mandatory*, *Minor* or *Info*. Attendees characterized as *Mandatory important* or *Mandatory* have to be available for the meeting to be confirmed. When no time slot can be found, when all of the mandatory participants are free, the initiator receives an error message. The meeting cannot be scheduled either, if one or more of the *Mandatory important* or *Mandatory* participants reject the final result. The difference between these two classifications is the timetable of the *Mandatory important* attendees are weighted higher and therefore do have more influence on the negotiation of the best time slots. *Minor* participants can attend a meeting. Users classified as *Info* will only receive an information about the meeting from the coordinator when the meeting is accepted by every participant.

4.3.1 Participant

A participant is specified by the following data:

- Identification (ID) of the participant
- Real name of the participant

- Priority weight of the participant for this meeting

The coordinator, the initiator and each participant can be identified by his or her ID and real name. Additionally a weight of priority for each participant defines the importance of the participants' time table on the negotiation result, i.e., the higher the weight of priority, the higher the influence of the participants' preferences on the time slot of the meeting. The priority weight for a participant is defined for each meeting and is graduated into:

Info attendees do not influence the time negotiation. When a time slot is found and accepted by all participants, the *Info* attendees receive an information about the scheduled meeting.

Minor attendees can attend a meeting. Their influence on the result of the meeting is very low, and the meeting can be scheduled even if one or more participants are not able to attend the meeting or reject the result.

Mandatory attendees have to be available for the meeting to be confirmed. When no time slot can be found, where all of the mandatory participants are free, the initiator receives an error message. The meeting cannot be scheduled as well, if one of the *Mandatory important* participants reject the final result.

Mandatory important participants are treated similarly as those characterized as *Mandatory*. The difference between these two classifications is the timetable of the *Mandatory important* attendees that are weighted higher and therefore do have more influence on the negotiation of the best time slots.

This classification is extending the classification of Chun *et al.* [8] with the superordinate instance of *Mandatory important* attendees and the *Info* classification for users not participating the meeting, but wants to be informed about when the meeting will occur. In the approach of Chun *et al.* an attendee additionally can be classified by its rank or position within an organization, such as CEO, CFO, CTO, VP R&D, VP Sales etc. [8]

As this protocol is designed not only for organizations, the rank is forgo and the weight of the user is extended to identify persons with extra ordinary importance.

4.3.2 Event

At the beginning of the scheduling process, the initiator specifies the event to be negotiated. Therefore, he or she records all the required data. An event can contain the following information:

Event title A short description of the event to be negotiated. The title is a simple identification for the event, e.g. "Budget meeting", "Project meeting" etc.

Location A textual description of the location where the event should be held. In future research the location could be extended with geographical data to define the location of the event precisely.

Content A detailed description of the event. The content description is a continuous text, e.g. the agenda of the meeting.

Duration The proposed length of the event in minutes. The length can be specified as *30 minutes*, *1 hour*, *1:30* etc.

Horizon An approximate time horizon when the event should occur. The time horizon is internally transferred into a start time and an end time to freeze the time horizon for the negotiation process.

Initiator The ID of the user initiated the meeting negotiation. Error messages due to unexpected situations occurring during the negotiation process are transferred to the supervising initiator. The initiator than has to decide, how to proceed.

Coordinator The ID of the agent coordinating the negotiation process. The coordinator is the central point where all messages and responses intersect during the scheduling process.

Participants The set of participants for the event. Each participant record contains the data described in section 4.3.1, i.e., *ID*, *Real name* and *Priority weight* of the participant.

Chun *et al.* [8] additionally define *Day-of-the-week* as part of the event information. This subjective preference predefined by the initiator has deep influence on the result. The meeting can only be scheduled on the specified day of the week. When a meeting for example should be held on Tuesday because one of the attendees is only available on Tuesday, a meeting will never be scheduled on another day, if the participant is classified as *Mandatory* or higher, even if the invitee is available.

Besides the data defined by the initiator, an event contains further internal information required for the negotiation process. The event can contain the following internal data:

ID The identification number of the event, respectively the scheduling process. Each event and its negotiation process is identified by a unique number. Every message exchanged has to contain the ID to refer the event.

Start time of time horizon The start time stamp of the time horizon. After the initiator has assigned the time horizon, it is internally transferred into a starting and an ending time to freeze the time horizon for the negotiation process.

End time of time horizon The end time of the time horizon. Internally, the horizon is transferred into a starting and an ending time.

Time out deadline A time stamp, until the meeting has to be scheduled. When the time out time is reached, the negotiation is canceled by the coordinating agent and the initiating user receives an error message.

Time of creation The time stamp of the creation of the event.

The internal data are created after the initiator has recorded the required event data, before the event is sent to the coordinating agent.

Time Horizon

The time horizon specifies the period when the event can be scheduled. To make it easier for the user to decide the interval when the meeting should take place, the classification is adapted to a human like graduation of time periods. The horizon can be one of the following time intervals: *Tomorrow, Within the next two days, Within the next three days, Within the next four days, Within the next five days, Within the next six days, Within one week, Within next week, Within the next two weeks, Within the next three weeks, Within one month, Within two months*. In the background the time horizon converted into a starting time stamp and an ending time stamp. The starting time stamp is equivalent to the meeting time out.

After the initiating user has recorded the information described above, together with a new scheduling request, the event data are sent to the coordinating agent. The coordinator then spreads the event data to all participants with a priority weight higher than *Info*.

4.3.3 Time Slot Preferences

In the *Calculate time slots* state the timetables of the participants are matched and the most apposite time slots for all participants are calculated. When the calculation generates one or more possible time slots to meet, the times are presented to the participants and they can weight each time slot by their personal preferences. Each participant can rate each of the time slots with one of the following preference weights:

- "Perfect"
- "Ok"
- "Rather not"
- "No chance"

The classification of the time slot preferences is similar to the classification of Brzozowski *et al.* [6]. They argue the four point rating scale to offer more precision than a binary free/busy scheme, and — because there is no neutral rating — still forcing users to take a stand.

4.4 Message Design

The data described in section 4.3 is exchanged as properties in XMPP messages. To transport all necessary information between the participants the XMPP protocol has to be extended with different messages. Each message has to contain the *ID* of the event respectively the scheduling process and the type of message. The type could be a *Request*, *Response* or an *Error*. Listing 4.1 shows the basic architecture of the messages with the essential information message type *MESSAGE_TYPE* and the identification number *SCHEDULING_ID* of the scheduling process. The data is embedded into the XMPP message using XMPP properties. Listing 4.1 shows the two possible message types (Scheduling- and Coordination-Request and -response)

and the possible subtypes listed between the squared brackets and separated by the pipe symbol.

```
<message xmlns="http://www.excellentis.org/meetme/message"
         id="um95T-5" to="mozart@musicians.com" from="beethoven@musicians.com">
  <properties xmlns="http://www.jivesoftware.com/xmlns/xmpp/properties">
    <property>
      <name>MESSAGE_TYPE</name>
      <value type="string">[scheduling_request|scheduling_response
      |coordination_request|coordination_response]</value>
    </property>
    <property>
      <name>REQUEST_TYPE|RESPONSE_TYPE</name>
      <value type="string">[timetable_query|preferred_timeslots|show_result|
              release_result|cancel_negotiation|
              negotiation_request|state_request|update_request|
              timetable_response|preference_response|
              result_acceptance|release_result_response|
              negotiation_response|update_response|state_report]</value>
    </property>
    <property>
      <name>SCHEDULING_ID</name>
      <value type="long">[unique identification number]</value>
    </property>
      <property>
      <name>TIMEOUT</name>
      <value type="string">[UTC time stamp]</value>
    </property>
  </properties>
</message>
```

Listing 4.1: Basic message architecture with all message types, response and request types listed

4.4.1 Scheduling Requests

The communication between the coordinator and the participants is based on request – respond messages. Each step in the negotiation process is initiated by the coordinator by sending one of the requests described in this section. The participants are answering these requests with response messages described in the next section.

Timetable Query

The first message sent by the coordinator at the beginning of a new negotiation process, is the request for the timetables of the participating users. Besides the essential information (*scheduling id* and *time out*) this request contains all information of the projected event to schedule (*title, location, time horizon,* etc.). If a participant accepts to negotiate the event, his or her will agent respond with a *Timetable Response* message described in section 4.4.2. When the user does not want to be scheduled, he or she rejects the timetable query and the coordinator receives an error message discussed in section 4.4.6.

Preferred Time Slots

After the coordinating agent has aggregated the weighted timetables of the participant and extracted the best fitting time slots, these time slots are sent to the participants' agents in the *Preferred Time Slots* request message. The agents show the result to their users to weight the time slots by their preferences. The request contains a list of time slots, ranked by their overall weight.

Show Result

In the next state the preferred time slots are matched by the coordinating agent. When this task produces a result, this time slot is sent to the participants agents again in the *Show Result* request message to be presented to their users. The participants can then choose either to accept the final time slot, or to reject. When a user has accepted the result, its agent sends the *Result Acceptance Response* message described in section 4.4.2 to the coordinating agent. If the participant rejects the time slot, because the time is inconvenient, his or her agent responds with an error message described in section 4.4.6.

Release Result

At the end of the negotiation process, when the final time slot is accepted by every participant, the coordinating agent sends a *Release Result* request to every participant's agent, to announce the end of the negotiation and instructs the agents to add the result to their users' calendar. No response is expected on this request. When an agent could not contact its users' calendar service, it sends an error message described in section 4.4.6 to the coordinating agent.

Cancel Negotiation

The initiating user can cancel the negotiation of a meeting at any time during the scheduling process. Therefore, the initiating agent sends a *Cancel Negotiation* message to the agent acting as coordinator, which spreads the message to all participants' agents (see also section 4.4.5).

4.4.2 Scheduling Responses

As illustrated in the section before, the agents communication with requests and responses. Except to the *Release Result* request, every request initiated by the coordinator is answered by the participating agents with one of the response message, described in this section.

Timetable Response

The participants' agent responds to the *Timetable Query* request with the *Timetable Response* message, containing a weighted time table of its user. Therefore the agent request the time table from the users' calendar service and weights the time slots corresponding to the preferences of its user. More preferred time slots are rated with higher weights. Unwanted times receive a negative weight. The coordinating

agent collects the timetables of all invited users and chooses the time slots for the next negotiation step with the highest overall weights.

Preference Response

When a participant's agent has received a list of time slots in the *Preferred Time Slots* request message, it asks its user to rate the time slots corresponding to his or her individual preferences. After the user has ranked the time slots, the agent sends the users preferences back to the coordinating agent within the *Preference Response* message.

Result Acceptance Response

When the coordinating agent has calculated a final result, the time slot is sent to the participants' agents requesting the acceptance of their users. The final result is shown to the participant and each user can choose either to accept or to reject the final time slot. When the user has accepted the result, its agent sends the *Result Acceptance Response* message to the coordinating agent. If the participant rejects the time slot, because the time is inconvenient, the agent responds with an error message described in section 4.4.6.

Release Result Response

At the end of the negotiation process, after a time for the given event is found, accepted by the participating users and added to their calendars, the coordinating agent sends a release result request to make the participating agents clean up their data store. After the agents have deleted the no longer required scheduling data, they have the possibility to send last information about the cleaning to the coordinating agent. But this *Release Result Response* message is not mandatory.

4.4.3 Coordination Requests

As mentioned in section 4.3 every agent can act as initiator, coordinator and/or as participant. The splitting of the rules has the advantage to be more flexible regarding the most efficient allocation of resources for the negotiation process. As this protocol is designed to be used on mobile systems, the coordinating task can be outsourced to a more powerful system. Therefore, it is necessary to augment the scheduling protocol with coordinating messages extracted between the initiating and the coordinating agent. In this section request messages of the coordinating protocol are described. The next section will illustrate the corresponding responses.

Negotiation Request

When an initiator does not want his agent to coordinate a meeting, he or she can choose any other accessible agent to execute the coordinating task. Therefore, the initiator's agent sends a *Negotiation Request* message to the elected agent. This request contains the information of the event to be negotiated, i.e. *title, location, description, time horizon,* etc., the initiator and a list of participants.

The designated agent can respond with an acceptance or a rejecting error message described in section 4.4.6.

State Request

To reduce unnecessary message traffic, no information belonging the state of the negotiation process are sent from coordinating agent to the initiator. The actual state of the negotiation process can be requested by the initiator with the *State Request* message.

Update Request

The values of an event in negotiation can be changed by the initiator in every state of the process. When, for example, the *Match preferences* task failed to extract a time slot, where every mandatory participant is available, the initiator is asked to change the basic values of the event (widen the time horizon for instance) before retrying to schedule the meeting. When the values of an event have to be changed, the initiating agent sends an *Update Request* to the coordinating agent. This request contains either the changed event or an updated list of participants. The coordinating agent matches the new event data and participant list with the data in negotiation and decides in which state to continue the scheduling process. Either the negotiation restarts by requesting the time tables from the participants, or — if the changes are marginal — the process is continued at a higher state.

Cancel Negotiation

The negotiation of a meeting can be stopped by the initiating user at any time during the scheduling process. The initiating agent therefore sends a cancellation message to the agent acting as coordinator, which spreads the cancellation with the coordination *Cancel Negotiation* message to all participants' agents (see also section 4.4.5).

4.4.4 Coordination Responses

As outlined in the section before, the coordinating task of the meeting negotiation can be outsourced to any agent accessible over XMPP. In this section the responding messages of the coordinating protocol are described.

Negotiation Acceptance Response

When an agent is requested to coordinate a meeting scheduling, it can accept the request and assume the coordinator's role, or — if the agent or its user does not want to coordinate the event — reject the request. The designated agent can respond with an *Negotiation Acceptance Response* message to accept the coordination or an rejecting error message described in section 4.4.6.

Update Response

To change the data of an event in negotiation, the agent of the initiating user sends an *Update Request* to the coordinating agent. If the user of the coordinating agent is not content with the new key data, he or she can reject the update, otherwise he or she can accept the update. The agent sends the decision within the *Update Response* message to the initiating agent.

State Report Response

The coordinating agent by itself does not send any state messages to the initiating agent to reduce unnecessary message traffic. The actual state of the negotiation process can be requested by the initiator with a *State Request* message. When the coordinating agent receives such message, it responds with a *State Report Response* message containing information about the present state of the negotiation process.

4.4.5 Scheduling Cancellation

An initiating user can stop the negotiation of a meeting at any time during the scheduling process. Therefore, the initiating agent sends a *Scheduling Cancellation* message to the agent acting as coordinator, which spreads the message to the participants' agents. Listing 4.2 shows an example of a cancellation message:

```
<message xmlns="http://www.excellentis.org/meetme/message"
  id="yUBo4-32" to="beethoven@musicians.com" from="mozart@musicians.com">
  <properties xmlns="http://www.jivesoftware.com/xmlns/xmpp/properties">
    <property>
      <name>REQUEST_TYPE</name>
      <value type="string">cancel_negotiation</value>
    </property>
    <property>
      <name>SCHEDULING_ID</name>
      <value type="long">1242337143179</value>
    </property>
    <property>
      <name>MESSAGE_TYPE</name>
      <value type="string">coordination_request</value>
    </property>
  </properties>
</message>
```

Listing 4.2: Scheduling Cancellation

4.4.6 Error Messages

During the negotiation process, different types of errors can occur. Errors affecting the negotiation of the event, are sent to the initiating user. Listing 4.3 shows the basic architecture of an error message:

```
<message xmlns="http://www.excellentis.org/meetme/message"
  id="aF5Tc-5" to="beethoven@musicians.com" from="mozart@musicians.com">
  <properties xmlns="http://www.jivesoftware.com/xmlns/xmpp/properties">
    <property>
      <name>RESPONSE_TYPE</name>
      <value type="string">timetable_response</value>
    </property>
    <property>
      <name>ERROR_MESSAGE</name>
      <value type="string">
        <SchedulingError>
          <errorMessage>No calender found!</errorMessage>
          <timeOfCreation>2009-01-01T20:09:32.79Z</timeOfCreation>
          <errorNumber>5001</errorNumber>
        </SchedulingError>
      </value>
    </property>
    <property>
      <name>SCHEDULING_ID</name>
      <value type="long">1230398028731</value>
    </property>
  </properties>
</message>
```

Listing 4.3: Scheduling Error

The message contains a short, human readable description of the error (*Error_Message*) and a number (*Error_Number*) to identify the error automatically by the agents. Error messages can be sent by participant, coordinating or initiating agents. Participating agents always send their error messages to the coordinating agent, that forwards the message to the initiating agent. The initiating agent displays the message to its user and the user can decide how to proceed. Error messages produced by the coordinating agent are directly routed to the initiating agent. Errors within the initiating agent are handled within the agent.

Negotiation Errors

To find the best feasible time for every participant to meet, the negotiating process is running through different states. In some states one of the following unexpected errors could occur:

No event received is triggered every time a message misses the required event data. When the coordination initializing *Negotiation Request* or the coordinating *Update Request* (see 4.4.3 for both) do not contain the event information, the designated coordinator responds with this error. Furthermore, this error is returned when the participant receives a *Timetable Query*, *Show Result* or *Release Result* (see 4.4.1) request without the required event information.

No matching time slots found After the coordinating agent has received the weighted timetables from all participants, the schedules are matched and the time slots with the highest overall weights are distributed to the participants to choose their preferences. If none of the mandatory participants are available in the time horizon and the matching process could not extract any time slot, the coordinating agent sends this error to the initiating agent.

No time slots received This error is generated by the participants agent, when the *Preferred Timeslots* request does not contain any required time slot to estimate. In that case it is impossible for the participating agent to display any time slot to the user.

Could not choose preferences due to technical problems in the user interface service, the participants' agent could not view the time slots to its user. By the system architecture the user interface service is integrated into the negotiation component as a plug-in. When the service is not registered at the negotiation component, the participating agent informs the initiating agent with this error message.

Could not match preferences Similar to the error above, this error is generated, when the preferences of the participating users could not be matched by the responsible plug-in.

No matching preferences When the coordinating agent has collected the time slot preferences from each participant, the preferences are matched and if there are no time slots left — because the users did not like one of the time slots or they prefer different time slots — the coordinating agent sends this error message to the initiating agent. Then the initiator can decide either to cancel the negotiation or to restart the process by choosing another planing horizon.

Could not display result is sent to the initiator, when the participants' agent was not able to execute the *Show Result* request and display the result to its user. This error occurs either due to an error in the user interface or the user interface service is not registered at the negotiation component. As described before, the user interface service is separated from the negotiation component.

No result received This error is generated by the participants agent, when the *Show Result* request does not contain the required time slot. In that case the participating agent is not able to display the negotiation result to the user.

Scheduling timed out At the beginning of the negotiation process, the initiator defines a time, before the meeting should be generated. When this time is reached without result, the scheduling process is stopped and the coordinating agent informs the initiator with this error message.

User Caused Errors

The error messages described above, occur during the negotiation within the scope of the coordinating agent. Furthermore, unexpected states of negotiation could also be generated, when users answer the requests given to them with "no". Two different types of scheduling error messages can be generated and sent to the initiator, when a user rejects either to be scheduled, or a user rejected the resulting time of the negotiation:

User rejected negotiation is sent to the initiator, when a user is invited to participate in a meeting, does not want to participate. The error message indicates the rejection of the participation request.

User rejected result At the end of the negotiation process, the final result is presented to the participants and they are asked either to accept or reject the resulting time slot. When a mandatory participant rejects the result, because the time slot is not acceptable for him or her, this error message is sent to the initiator. The initiator than can decide how to proceed.

The error messages can be generated and sent to the initiator, when a user rejects either the *Negotiation Request* or the *Timetable Query* request.

User refused negotiation is generated and sent to the initiating agent, when the user of the agent — asked to coordinate the negotiation of a meeting by a *Negotiation Request* message 4.4.3 — does not want his agent to act as a coordinator. The error message indicates the rejection of the coordination request. When the initiator receives this error message, he or she has to search for another agent, assuming to coordinate the meeting.

User refused update When the data of an event changes during the negotiation process (e.g. the initiator enhances the time horizon, when the meeting should be held), the user of the coordinating agent — if he or she is not content with the new data — could reject the update request. In this case, the *User refused update* error message will be sent to the initiating agent.

Calendar Provider Errors

The calendar provider module is the interface between a participating agent and the calendar service of its user. It requests and receives the calendar information from the calendar service and generates the timetable of its user to send it to the coordinator. After a meeting is scheduled, it adds the new event to the users' calendar as well.

No calendar found occurs when no calendar is specified for the calendar provider. This error is sent from the participant's agent to the initiator to inform the initiator that no timetable can be generated. The initiator then has to decide how to proceed.

Could not connect calendar service When the calendar provider module cannot connect with the users' calendar service, the user receives this error message. Either the connection to the service is broken, or the service is not available.

Calendar service login failed The calendar provider plug-in could not log in the calendar service, because the credentials (user ID and/or password) are not correct. When the login failed, the agent presents this error message to its user.

Could not add result to calendar When the calendar provider cannot add the negotiated and accepted event to the calendar, this error is viewed to its user to inform him or her that a meeting is scheduled, but not added to the calendar. The user then has to decide how to proceed.

Connection Errors

To communicate with other agents, every agent has to be connected with one XMPP server. If the agent cannot connect or login to this server, error messages are produced and viewed directly to the agent's user and the user can decide how to go on:

Not connected to server The agent could not connect with its server, either because the Internet connection is down, or the server is not available, or the address to the server is not correct. When no connection could be established with the server, the agents' user receives this error message.

Login failed The agent could not log into the server, because the credentials (user ID and/or password) are not correct. When the login failed, the agent presents this error message to its user.

Component Errors

By architecture the agents are composed of a bunch of different components (see section 5.1), bound to the centralized negotiation module as plug-ins. When one of the essential plug-ins is missing or an error occurs in these modules, one of the following error messages are presented to the agents' user:

No timeslot calculator set This error occurs, when no calculating module is registered at the coordinating agent. The coordinating agent sends the error message to the initiator.

No preference calculator set The preferences are matched by a separate calculator module. When this preference calculator module is not registered at the coordinating agent, the agent sends the error message to the initiator.

No calendar provider set When no calendar provider module is set, the participants' agent cannot create the requested timetable of the user. With this error message the participants agent informs its user.

Application Caused Errors

In the sections before, unexpected states caused by the involved users and errors occurring within the different components have been illustrated. In this section the various unforeseen exceptions and the application side will be introduced. These errors are generated, when information received from the coordinating agent could not be handled, either because it could not be displayed to the user, or it could not be processed internally.

Could not display negotiation request After a potential coordination agent has received a negotiation request, the application has to ask the user, either to accept or reject the coordination of the meeting time negotiation. In cases the agent cannot display this message - without consideration of the circumstances - this error message is transported to the initiating agent.

Could not display update request occurs in a similar case as the error above: When the update request of an initiating agent, could not be displayed to the coordinating user, the agent drops this error message.

Could not display state request as the two unexpected cases above, this message is sent to the initiating agent, when the state request could not be handled and displayed by the coordinating agent.

Could not display acceptance is created, when a participating agent could not handle and present the resulting time to the user. When this error is triggered, the coordinating agent will be informed about the circumstance, that one user was not able to accept or reject the resulting time.

Could not cancel negotiation As illustrated in section 4.4.5 the initiating user can stop the negotiation of a meeting at any time during the scheduling process. Therefore, a cancellation message is sent to the coordinating agent, which spreads this message to all agents invited to participate. This error message can be generated by each of the agent (the initiating, the coordinating and every participating agent) that cannot handle the cancellation request.

Could not add timetable to chart After the coordinating agent has requested the weighted timetables from the participating agents, it is waiting for responses and collects the received information to an internal memory for later evaluation. When the coordinating agent is not able to store a timetable from a participating user, this error message will be dropped and sent to the initiating agent.

Could not add preferences to chart is provoked, when the coordinating agent, trying to collect the preferences about the meeting time from the participants, could not add the data to its internal store.

Could not add decision to chart Similar to the last two exceptions, this error message is generated, when the coordinating agent could not add the decision of a participating user to its data store.

In heterogeneous environments connecting agents from different platforms and versions, the correctness of the messages could not always be guaranteed. Particularly in a mobile setting, when the messages are transported over wireless connections, the integrity of the message could suffer from the quality of the connection. The intent of these error messages, is to indicate the initiating user, that the agent which created the exception, was not able to handle one of the messages and to prevent the negotiation process from a deadlock. With this error message exchange, the protocol includes a basic protection to escape from stalemate situations. In the first step it should be the agents choice to demand the non-interpretable message again, until it can handle it, but the last step has to be in the hand of the initiating or coordinating user to decide how to avert the deadlock.

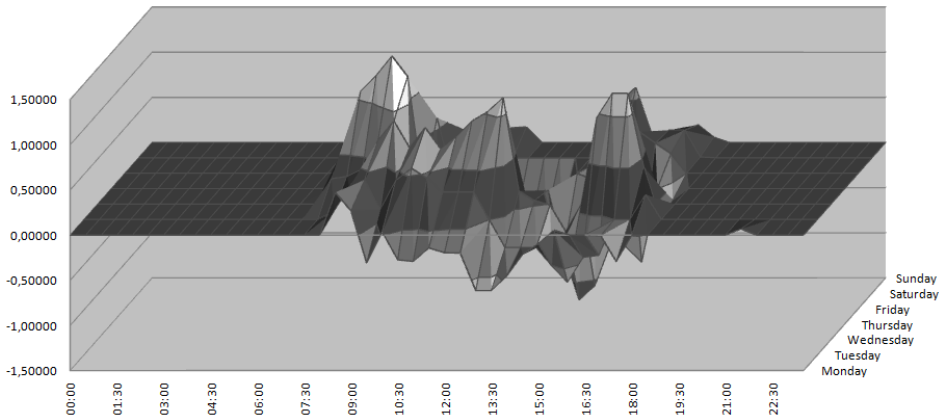

Figure 4.2: Aggregated and weighted time slots as graph

4.5 Time Slot Negotiation

In this section a time slot negotiation algorithm is introduced. As described in the sections before, the coordinating agent requests the timetables from each participant. Each time slot from the timetable is augmented with an individual weight representing the participants' preference to schedule a meeting. When a user initiates a new meeting to negotiate, he or she assigns each participant with an individual priority for this meeting (*Info*, *Minor*, *Mandatory* or *Mandatory important*). The weighted timetables are matched by the coordinating agent considering the priority level of each participant. Therefore, time slots are divided into predefined intervals. The weights for these intervals then are normalized and aggregated taking account of each participant's priority. Finally, the intervals are combined again and the algorithm selects the time slot with the highest overall weight, suiting the estimated duration of the meeting. If there are more than one time slots with the same weight, the earliest time slot is selected. Figure 4.2 shows the example of an aggregated, weighted time slots in a graph with a one week planing horizon.

4.5.1 Formal Definitions

Similar to Wainer *et al.* [49] a meeting m is defined as a 6-tuple $\langle A, \bar{a}, c, d, H, O \rangle$, where

- $A = \{a_1, a_2, \ldots, a_n\}$ are the participants agents,
- $\bar{a}$ is the initiator of the meeting. Commonly $\bar{a} \in A$ but the protocol allows that a initiator is not participant of a meeting. In this case $\bar{a} \notin A$,
- c is the coordinator of the meeting. This is the agent that performs the orchestration between the agents A and the initiator $\bar{a}$ and performs the negotiation task of the meeting scheduling. In most cases $c \in A$ and $c = \bar{a}$. The coordinating task can also be outsourced to an scheduling service and in this case $c \notin A$,

- d is the planed duration of the meeting,
- $H = \{h_s, h_e\}$ the planing horizon defined as the start and the end of the time horizon to schedule the meeting,
- $O = \{o_1, o_2, \ldots, o_n\}$ set of the participants priority for the meeting to schedule.

4.5.2 Time Slot Calculation

First the participants' agents A are requested to generate a set of weighted time slots $U(a_i, m)$ according to their users preferences, where

- $U(a_i, m) = \{u_1, u_2, \ldots, u_k\}$ set of weighted participant time slots for the meeting.

After each participants' agent A has executed the timetable generation, the coordinating agent c transforms the weighted timetables U of the participants A into a set of time slots T of the length v, where T between $h_s \in H$ and $h_e \in H$ and

- $T(a_i, m, U, v) = \{t_1, t_2, \ldots, t_g\}$ set standardized time slots of the length,
- v length of the interval of T.

Finally, to receive the most feasible time slot for the meeting m the coordinating agent c executes for the number of participants N the function

$$W_j(m, T_j) = \sum_{i=1}^{n} \frac{1}{N} \cdot |T_j(a_i, m, U_i, v)| \cdot o_i \tag{4.1}$$

The higher the value of $W_j(m)$ the more preferred the meeting m to schedule during the time t_j. The time slots with the highest overall weights $\max_{W(m,T)}$ are chosen as most feasible times for all participants Q, where

- $Q = \{q_1, q_2, \ldots, q_z\}$ set of z times with the highest overall weight $\max W_j(m, T)$.

If some time slots have the same weight, the earliest time slot will be chosen. Q is sent to the participants' agents A as result of the calculation.

4.5.3 Preference Matching

After the possible time slots for the meeting are calculated, the coordinating agent c sends the time slots with the highest overall weight Q to the participants' agents A and requests them to estimate their preferences $R(m, Q)$. When the coordinating agent has received the rankings $R(m, Q)$ from the participants A, it executes the preference matching function $S(m, R)$ to find the most preferred time slot for the number of participants N:

$$S_j(m, R_j) = \sum_{i=1}^{n} \frac{1}{N} \cdot |R_j(m, Q_j)| \cdot o_i \tag{4.2}$$

The higher the value of $S_j(m, R)$ the more the participants A prefer to schedule the meeting m at the time q_j. The time with the highest overall ranking $x = \max_{S(m,R)}$ will be distributed as final result.

The algorithm is extending the approach of Wainer *et al.* with the participants priorities for the meeting. Depending on the importance of the invitee, the preferences have greater or less influence on the final result. In the approach of Wainer *et al.* the preferences of each attendee are treated equally.

5 Meet Me: Developing a Scheduling Agent for Mobile Devices

This chapter deals with the architecture and technical design of the scheduling agent is presented. To demonstrate the ability of the protocol introduced in chapter 4, a prototype based on the mobile device platform Android [45] has been developed. The first section of this chapter demonstrates the system design of the prototype and how the prototype is integrated into the Android platform. The second section will demonstrate the data structure used to implement the new protocol in the mobile prototypical agent. Finally, the third section will show two scheduling examples, to illustrate how the prototype negotiates meetings.

5.1 System Design

Figure 5.1 shows the general architecture of MeetMe. The Java-based agents communicate with XMPP [26] the Extensible Messaging and Presence Protocol to find the most suitable time slots for users. The implementation is based on four layers. The two layers below are implemented independent from the underlaying platform and can be adopted into various types of systems:

Smack API Open Source XMPP (Jabber) client library for instant messaging and presence. The Java library contains the full implementation of the XMPP protocol and enables to create anything from a full XMPP client to simple XMPP integrations [24].

MeetMe API Is the platform independent implementation of the meeting schedul-

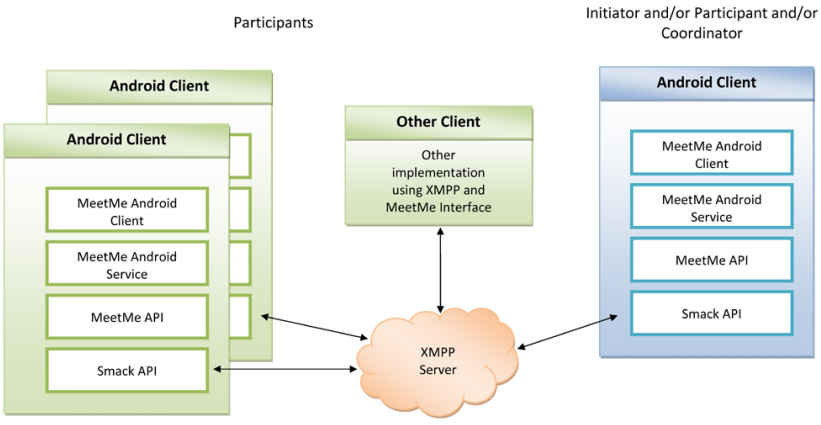

Figure 5.1: General Architecture

ing protocol and algorithm introduced in the chapter 4. The Java library is applicable for mobile devices as well as for desktop systems, or server applications. It contains the basic requirements for the protocol, e.g. the definition of messages, the interfaces of the data classes and implementations of the negotiation algorithms described in section 4.1. The API is applicable for each of the three rules: the initiating, the coordinating and the participating rule.

The *MeetMe API* depicts the protocol defined in section 4. It is a collection of predefined Java interfaces of the messages. Furthermore, it contains interfaces for future extensions, which can be integrated as plug-ins or replace the present basic implementations of the preference calculating module or the time slot matching module. In addition it is possible to extend the system with new calendar plug-ins, to integrate calendar from multiple calendar systems data into the negotiation process. The prototype contains a basic calendar interface to the Google Calendar service [18].

The implementation of the prototype is based on the Android [45] platform for mobile devices. The two upper layers are Android specific:

MeetMe Service The background service coordinating the scheduling of the events and listening for incoming meeting scheduling requests. The service is the core of the agent. It can act as initiator, sending the coordination request to the coordinating agent, as coordinator organizing the negotiation of the feasible meeting time, or as participant agent, answering the scheduling requests of the coordinating agent.

MeetMe Client The user interface of the agent. It forwards the entered information from the user to the agent, shows meeting scheduling relevant information messages during the negotiation and waits for the user to choose preferences. The client itself does not interfere into the negotiation process, it acts as messenger between the user and the service.

5.2 Data Structure

The Android platform includes an embedded SQL database engine SQLite [44]. The MeetMe prototype stores the meeting relevant data in a database. In this section the modeling of the handled data will be described. Figure 5.2 shows the modeling of the data. The system knows eight different data objects. The objects are referenced with an unique scheduling identification number (*schedulingId*):

Process is the main data object, containing the general information of the scheduling process.

Event stores the information about the planed meeting, like *title*, *location* of the event, information when and how long the meeting will take (*duration* and *horizon*) and extended *description* of the meeting.

Participant refers a user to a meeting scheduling process. The participant object contains an index for the importance of the user for the event the user is invited to. As described in section 4.3 the protocol knows several types of importance ratings. This categorization is expressed as *float*.

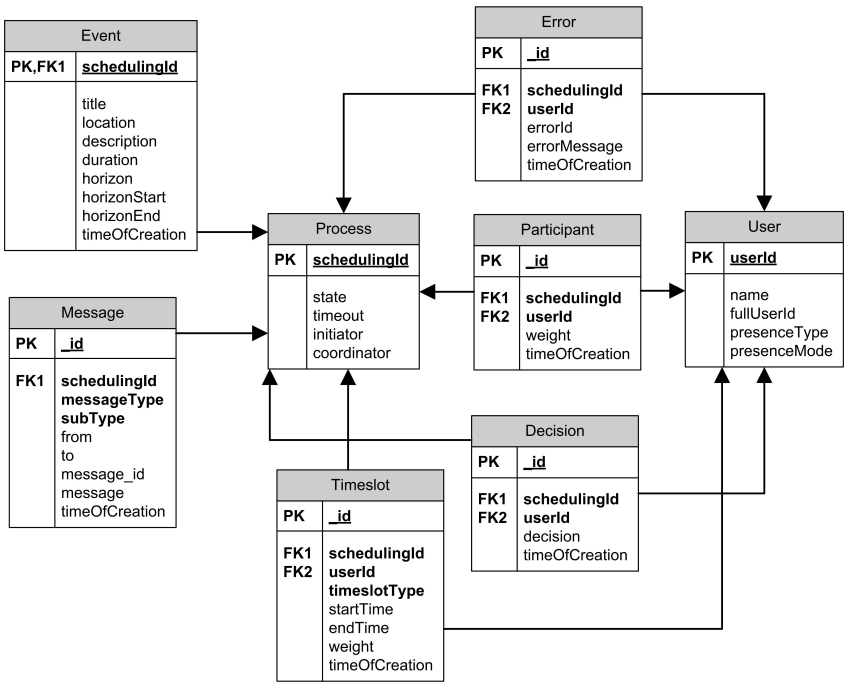

Figure 5.2: Data Structure

User contains the data to identify one user, i.e. the users *name*, his or her identification (*userId* and *fullUserId*) and the type and mode of his or her presence.

Timeslot stores the preference of one user for a specified time slot. The time range is defined by the *startTime* and *endTime*. The preference for a time slot is expressed as a *float* value and is stored in the *weight*. The data schema knows several types of time slots: *Timetable*, a simple *Timeslot*, a *Preference*, or a *Result*. As these data object contain the same type of data, they are summarized into one data object.

Decision contains the data of one user decision concerning one scheduling process result. A user can accept or reject the final time for a meeting.

Message The communication data is stored into message objects. Due to technical restrictions for mobile devices we can assume that a permanent communication between the involved agents cannot be guaranteed. Therefore, the generated messages are stored until the opposite agent is active and ready to receive the request or response. To prevent the devices from unnecessary messages only the newest messages are transmitted, newer messages replace older ones of the same type.

Error When unexpected events occur, the participating agents send error messages to the coordinating agent. The coordinating agent decides either to handle the unforeseen situation by itself, or inform the initiating user to decide how to proceed. The error messages are stored in error objects.

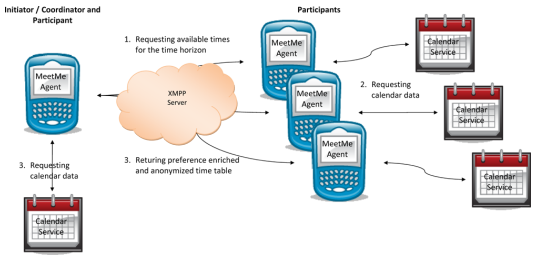

Figure 5.3: The coordinating agent is requesting the timetables

Messages for temporary unavailable agents are stored in a database on the sending agent until the receiver is available. For future development the MeetMe API could also be adopted to implement a permanently available coordination service, which could prevent dead locks between alternately unavailable agents.

5.3 Real World Examples

In this section two real world scheduling example are illustrated, representing the two most tedious scheduling scenarios introduced in chapter 1, *Spare Time Scenario 2* and *Business Scenario 4*. The meetings scheduled in these two scenarios, contain human attendees, some of these attendees are abroad or on the road, and for the meetings there are some resources needed. Furthermore, in the business scenario the persons invited for the meeting are working for different companies. To involve the resources as part of the meeting, it is required to run an MeetMe agent for every resource. Furthermore, it is necessary for every resource to have its own calendar data. In following the examples, it is assumed, that the resources confirm with these prerequisites.

5.3.1 Spare Time Example

David wants to meet his friends within the next days to go into sports. He wants to play basketball with them in a sports hall. As his friends and David are very busy, it is not easy for them to find convenient times for the game. David starts his MeetMe agent, gathers the essential data of the event, defines his friends, himself and the basketball court as participants and starts the negotiation.

Requesting Availability Times

David's agent acts as initiator, coordinator and participant. The agent requests the availability times of all participants for the given time horizon. Therefore, all agents connect their users calendar services and request the timetables for the horizon. Furthermore, the participants agents weight the times with the preferences of their users and send them back to the coordinating agent to match. Figure 5.3 shows the first step in the scheduling process.

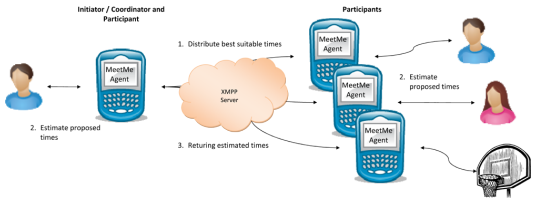

Figure 5.4: The participants are asked to estimate the time proposals

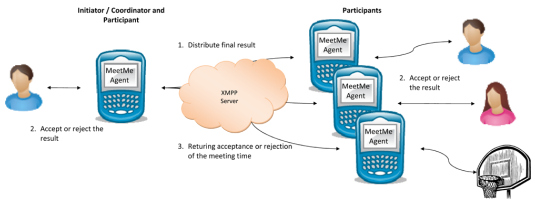

Figure 5.5: The participants are asked to accept or reject the result

Choosing Preferences

After the timetables of the participants are matched, five best fitting times are distributed to all participants. David and his friends are asked to choose the best suitable time for the event and send the preferences back to the coordinating agent. Agents acting on the behalf of resources do not require user interaction. They choose the preferences for the resource in a basic preference algorithm: When the resource is available, the time is marked as "Perfect", otherwise the time is weighted as "No Chance". Figure 5.4 shows the second step in the scheduling process.

Accepting or Rejecting Result

The favored times of the participants are matched again by the coordinating agent and the most suitable time will be chosen as final result. A result exists, if at least the "Mandatory" or "Mandatory Important" marked participants are available at that time. If there are more than one result, the result with the highest rating is selected. In the next step the result is distributed to the participants. David and his friends are asked to accept or reject the result. Agents defined for resources, accept the time, if the resource is available. Otherwise they will reject the result. Figure 5.5 shows the third round of the meeting negotiation.

Adding Event to Calendar

If the result is accepted by everybody, the MeetMe agents of David and his friends add the new event automatically to their calendars. Figure 5.6 shows the final step in the negotiation process of the spare time meeting. Every time an unexpected situation occurs during the meeting negotiation, David receives an error message. In this situation it is his decision how to proceed. When, for example, a participant rejects the final result, David receives an error message, and then he can change

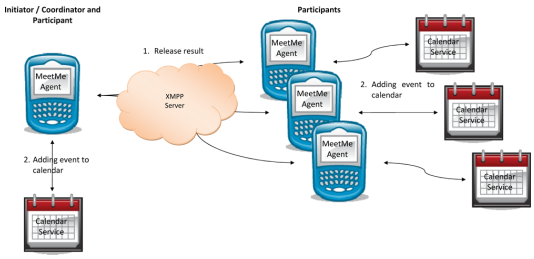

Figure 5.6: The new event is added to the calendars of the participants

the parameters for the meeting, e.g. extend the time horizon for the meeting, and restart the negotiation with the new values.

5.3.2 Business Example

In this example the business woman Jennifer schedules a kick-off meeting for a project. For this meeting she wants to invite some people from her company and some persons from other companies. The meeting should be held in a conference room. In this kick-off meeting Jennifer wants to give a short presentation of the project. Therefore, she needs a beamer to project the presentation onto the wall. To negotiate the meeting time, Jennifer has to proceed similar as David from the spare time example: She starts her MeetMe agent, enters data of the kick-off meeting and defines the required attendees from her company, the ones from the other organizations and herself as participants. To be able to select the people from the other companies for the meeting, it is required to give them access to the same XMPP server, or let the XMPP servers of the different company exchange messages. Various server implementations offer a server spanning message and presence exchange [25]. Next Jennifer chooses the conference room and the beamer as mandatory participants and starts the negotiation. As mentioned before, the resources need to have their own agent and calendar data.

Jennifer's agent acts as initiator, coordinator and participant and requests the availability times of all participants for the given time horizon. The agents from the participants connect their users or resource calendar services and request the timetables for the time horizon. Then the participants agents weight the times with the preferences of their users and send the weighted and anonymized timetable back to the coordinating agent. As coordinator Jennifer's agent matches the received timetables and generates time suggestions for the meeting. These proposals are distributed to all participants. Jennifer and the participants are asked to estimate the times for the event and send the preferences back to the coordinating agent. The agents for the conference room and the beamer do not require user interaction. They choose the preferences as follows: When the resource is not booked, the time is marked as "Perfect", otherwise the time is weighted as "No Chance".

When the coordinating agent has received information from every participant, it starts matching these preferences to generate a resulting time. A result exists, if at least the "Mandatory" or "Mandatory Important" marked participants are available at that time. If there are more than one result, the result with the highest

rating is selected. When two resulting times do have the same weight, the earliest time is selected as final result. Next the final result is distributed to the participants and resources to request their acceptance. Jennifer and the other participants are asked to accept or reject the result. Agents for resources, automatically accept the time, if the resource is available, otherwise they will send a rejection message. If the result is accepted by everybody, the coordinating agent sends a message to inform the participating agents, that the time was accepted and they can add the event to their users calendar.

When an unexpected situation occurs, Jennifer will receive an error message, identical to the spare time example, it is her decision how to proceed.

6 Evaluation and Further Work

In this chapter the MeetMe prototype illustrated in chapter 5 is theoretically evaluated against the scheduling scenarios from chapter 1.4. The prototype is tested with each of the business and spare time scenarios, considering people working for same companies, people abroad, people working for different companies, friends and resources. Furthermore, in the second part of this chapter field test scenarios are introduced, to illustrate how the prototype can be tested under real world conditions and how the system can be integrated into existing information exchange environments. In the third part of this chapter possible future concepts for scheduling with mobile devices are presented. It will be discussed, how the MeetMe agents can be improved and which extensions in the protocol could be made.

6.1 Evaluation

The MeetMe system was designed to meet each of the scheduling scenarios sketched in chapter 1.4. In this section the prototype introduced in chapter 5 will be evaluated. The agents are fully runnable on mobile devices. They schedule meeting times on the behalf of their human user, or on behalf of resources, e.g., conference rooms, sports grounds, etc. Table 6.1 shows the summary of the meeting scheduling evaluation.

In section 5.3 two generic meeting scheduling examples have been illustrated. One sophisticated kick-off meeting example appearing regularly in business life, where business people from different companies want to meet and additional resources are required. The second example demonstrates a spare time meeting, where friends want to play basketball at a sports ground.

Scenario	Negotiable
Business Scenario 1	yes [1]
Business Scenario 2	yes
Business Scenario 3	yes [2]
Business Scenario 4	yes [3]
Spare Time Scenario 1	yes
Spare Time Scenario 2	yes [3]

[1] Every participant uses a mobile device
[2] The participants use the same XMPP server or the servers are connected
[3] The resources have their own agent and calendar

Table 6.1: Evaluation Summary of the MeetMe Prototype

The real world example from the business life covers the meeting in the *Business Scenario 4*, where a meeting should be organized among persons working for different companies, and additionally considering resources like a conference room. The example has shown, that the MeetMe prototype is able to schedule such a meeting. Therefore it is required to configure the central XMPP servers to let the agents of the participants from the different companies communicate with each other. Either all participating agents receive an account on the same server, or — if multiple servers are involved — the different servers are configured to transfer the messages. The open source XMPP server implementation Igniterealtime, Jive Software [25] is able to exchange messages and presence between server instances. Furthermore, it is possible to exchange messages between Openfire and other non-XMPP servers.

The same requirements are necessary to schedule the meeting in the *Business Scenario 3*. It is the same scheduling situation, without considering the conference room and the beamer. Therefore, it can be assumed that the meeting in *Business Scenario 3* is negotiable with the MeetMe agent prototype.

Designed for mobile devices it is possible to integrate persons outside, on the road, or abroad into the negotiation of a meeting using the MeetMe prototype. This implies, that it is possible to schedule the meeting in *Business Scenario 2*. Prerequisite for invitees to participate the negotiation, is — as for every scenario described — the connection to the centralized XMPP server.

If all of the participating people own a Android-based mobile device — which is assumed —, it is able to schedule the meeting in *Business Scenario 1*. The current implementation of the MeetMe prototype is limited to the Android platform. The MeetMe API would additionally allow desktop implementations, but yet no desktop application exists. The negotiation process would be the same for future desktop versions.

The spare time example in section 5.3 can be applied to the meeting in *Spare Time Scenario 2* in chapter 1. One person initiates a meeting with his friends on the road, to go into sports. Additionally the sports ground — a basketball court in the example — has to be considered into the negotiation. For a successful negotiation with the MeetMe prototype the sports ground has to be represented by its own scheduling agent. Furthermore, the resource needs its own calendar data to extract the availability information of the sports ground.

6.2 Integration and Testing

This section outlines some field test scenarios for further evaluations of the MeetMe system under real world conditions. Furthermore, this section tries to illustrate how the mobile agents can be integrated into current information technology environments to support busy people finding feasible times to meet. This section is divided into two parts: The first part describes how the system can be used to let people organize their meetings in spare time and in the second part possible scenarios for the integration into a business technology environment are lined out.

6.2.1 Spare Time Integration

The flow of information in the MeetMe system is organized by a centralized XMPP server. To get the system started, it is only necessary to connect each of the attendees to this server instance. Participants can set up their own XMPP server instance, or use one of the publicly accessible services. XMPP server implementations are freely available for download and installation. One of the open source XMPP server implementations is Openfire [25]. This server instance is able to exchange messages and presence between several server instances and is also able to exchange messages between Openfire and other non-XMPP servers.

But it is not necessary for the attendees to set up their own server instance. One publicly available XMPP server is provided by Google and is called Google Talk or GTalk [19]. This communication service is fully integrated into the other Google services like the Google Calendar [18] service.

To let the MeetMe agents communicate with each other, it is necessary for every participating person to have an account on the XMPP server. Furthermore, it is required to set up an agent and account for each resource invited to the meeting. Most of the current server implementations offer Internet services to let new user register easily. In the current implementation the agents only accept messages from persons and resources on its user's roster. Therefore, it is required to add each participant on each participants roster. Many XMPP server instances allow to administrate the user's roster through web interfaces, as sub-sites on the user account. The XMPP protocol is capable to add and remove other users from the roster with specific administration messages. The MeetMe agents currently do not include any roster managing functionality. With the evaluated implementation it is not possible to add or remove participants from the user's roster.

The same requirements apply to integrate resources into the meeting negotiation. Each resource has to be represented by an own XMPP account. Furthermore, each participating person and resource does need his or her or its own calendar instance to representing the basic availability data.

Simple meeting scheduling test between two participating agents — representing persons and resources as well — have already been successfully executed. But the scalability of the system has not been tested. In further test scenarios various meetings should be scheduled with an increasing number of participants. As the resources of mobile devices are limited, it seems obvious, that the coordinating agent working to capacity with a high number of participating devices.

6.2.2 Business Integration

For the integration into the technology environment of a company, the same basic prerequisites as for the integration for spare time use are valid. It is required to configure one or more central XMPP servers to let the agents of the participating persons from the same or different companies communicate with each other. Depending on the internal privacy and security terms, it seems unlikely for companies to use publicly available services like GTalk to negotiate their meetings. As mentioned above, XMPP server implementations are freely available and can be comfortably integrated in the current information technology environment of the

company.

Different from the integration for spare time uses, the existing user management from the enterprises can be simply adopted into the server instance. The open source XMPP server implementation Openfire [25] is able to acquire the user management from an Active Directory or an LDAP service[1]. Furthermore Openfire is able to exchange messages and presence between different server instances. It is possible to exchange messages between Openfire and other non-XMPP servers. Therefore, it is not necessary for the participating users to have an account on the same server.

As mentioned above, the current prototype has already been tested within an one-server setup. Next the approach should be tested with agents distributed on multiple XMPP server instances. Depart from the limitation of the coordinating agent, the system seems to be limited also on the server side: Especially in the described business scenarios, the communication between the servers from different companies could be difficult, but more from the security aspect, than from the technical point of view. Security administrators maybe have concerns connecting their internal communication server with one server from another company.

6.3 Future Work

In this section possible future concepts to extend the MeetMe prototype to improve meeting scheduling on mobile devices are presented. The current implementation of the MeetMe scheduling agent is matching the timetables of the participants with a basic transparency time algorithm. This algorithm evaluates the calendar data for the given time horizon: Times where the participant or the resource is booked, receive a negative weight. Times with no other calendar entry are weighted with a positive value. The small time horizon between free and busy times are weighted with a neutral to negative weight, the nearer the time is to the busy time block. Preferences of the user are considered only by general availability times, a user can specify when he or she is generally available during the seven days in the week: e.g. Monday from 8 am to 5pm, Tuesday from 9am to 4pm, etc. The focus of future development should be more on the integration of user preferences to improve the scheduling results. Different approaches discussed in chapter 2 have shown directions how to aggregate and evaluate user preferences for meeting scheduling. However, it should be kept in mind, that mobile devices have limitations in their computing resources. Sophisticated calculations should be avoided, because they could interrupt essential functions like phone connections on cellphones.

The matching of the timetables and the matching of the preferred times from the participants is one of the tasks, that can actually occupy considerable resources on small mobile devices. While calculating the best times for the meeting, the coordinating agent divides the timetables into time slots of predefined length (normally time slots of 30 minutes). Then it aggregates the normalized preference weights for each time slot for each participant. Finally, subsequent time slots with the same weight are joined together again and the times fit the highest rating are chosen

[1] Integration of LDAP and Active Directory into the Openfire XMPP server: http://www.igniterealtime.org/builds/openfire/docs/latest/documentation/ldap-guide.html

as the suggested times. Obviously the effort for this calculation grows with the number of participants. Therefore, the MeetMe protocol introduced the rule of the coordinator, which can be delegated to a more powerful matching service. This calculating service could run for example on a well equipped server machine.

A server based coordinating service has another benefit: It is permanently available for the participating agents. In a mobile setting, it cannot be guaranteed, that all of the participating mobile devices are online during the whole negotiation process. The agents exchange the messages only when the counterpart is available. In rarely cases this could lead to a deadlock, where no messages are exchanged, because two agents (e.g. the coordinating and a participating agent, or the coordinating and the initiating agent) are never available at the same time. With a centralized 7x24h coordinating agent, this lookout could be avoided.

The current implementation is limited to receive timetables from and save new events to Google Calendar [18]. Other calendar services could be easily integrated into the MeetMe agent as plugins. Interfaces for the calendar data exchange protocol *CalDav* and *iCal* could be the next steps to enlarge the operating area of MeetMe. Furthermore the Android [45] platform provides calendar information which could be integrated as well.

Currently calendar data is integrated into the negotiation of the best meeting time. In the next step more user specific data available on Android should augment the computation of the most suitable time slots for the users. Android includes a localization service providing the current position of the device and its user. In the next step the position data could be integrated into the calculation and the weighting of the users timetable to improve the result. If, by example, a participant is currently in India it would be unlikely for him/her to follow a meeting in New York within the next 6 hours.

7 Summary and Conclusion

This work introduced the design and implementation of a distributed multi-agent meeting scheduling system, called MeetMe. The evaluation of current meeting scheduling systems has shown that none of the approaches are designed to schedule meetings on mobile devices. Unlike to current approaches, MeetMe allows to execute the negotiation of the meeting time fully on mobile devices.

From the results of the evaluation, a new scheduling protocol was specified. This protocol, specially adopted for mobile use, defines three types of actors, involved in a meeting time negotiation process: The initiator releases the negotiation. He or she invites the participants for the meeting and acts as supervisor of the negotiation process. The coordinator organizes the negotiation of the meeting time. He or she requests and matches the timetables and preferences from the participants to extract the most feasible time to meet. Finally, the participant is an invitee for the meeting. Each of the actors can play multiple rules: the initiator can be the coordinator and / or a participant and the coordinator can be the initiator or a participant as well.

The MeetMe prototype — implemented on the base of the theoretical concept — is resting on four layers: The agents communicate over a specially for mobile use adopted Smack XMPP library. The MeetMe API depicts the meeting scheduling protocol. The implementation of the prototype is based on the Android [45] platform for mobile devices. The two upper layers are Android specific: The MeetMe Service is the core of the agent. The user interface is represented by the MeetMe Client. Each of the mobile agents, presented in this work, are able to act on behalf of the three types of actors.

As mobile devices still have limited resources, the elaborated coordinating task can be outsourced to a self-contained coordinating service. The MeetMe API is implemented independent from the underlying platform. A server based coordination service could improve the performance of the negotiation and could be implemented in future works.

With MeetMe the negotiation of a meeting is a four-step-process: In the first step the weighted timetables of the participants are matched to extract the best fitting times. Next these time proposals are estimated by the participants and matched again to achieve the final time. The participants then have the possibility to accept or reject the resulting time. If all mandatory participants accept, the time is automatically added to the invitees calendar, otherwise the negotiation has to start over again with modified parameters.

The prototype performed well in basic two-agents test settings. Extended tests with a higher number of agents involved have not been executed so far. But this work introduces enhanced test scenarios, for complex business situations and spare time meetings as well, to test the prototype in future works.

Bibliography

[1] Beard, D., Palaniappan, M., Humm, A., Banks, D., Nair, A. and Shan, Y.-P. (1990). A visual calendar for scheduling group meetings. In: *CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pp. 279–290. ACM, New York, NY, USA. ISBN 0-89791-402-3.

[2] Bellifemine, F., Caire, G., Poggii, A. and Rimassa, G. (2003 September). Jade - a white paper. http://exp.telecomitalialab.com.
Available at: http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf

[3] Berry, P., Gervasio, M., Uribe, T., Myers, K. and Nitz, K. (2004 March). A personalized calendar assistant. In: *AAAI Spring Symposium Series*. AAAI, Stanford University.

[4] Berry, P., Peintner, B., Conley, K., Gervasio, M., Uribe, T. and Yorke-Smith, N. (2006). Deploying a personalized time management agent. In: *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 1564–1571. ACM Press, New York, NY, USA. ISBN 1-59593-303-4.

[5] Berry, P.M., Albright, C., Bowring, E., Conley, K., Nitz, K., Pearce, J.P., Peintner, B., Saadati, S., Tambe, M., Uribe, T. and Yorke-Smith, N. (2006). Conflict negotiation among personal calendar agents. In: *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 1467–1468. ACM, New York, NY, USA. ISBN 1-59593-303-4.

[6] Brzozowski, M., Carattini, K., Klemmer, S.R., Mihelich, P., Hu, J. and Ng, A.Y. (2006). grouptime: preference based group scheduling. In: *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 1047–1056. ACM, New York, NY, USA. ISBN 1-59593-372-7.

[7] Calefato, F., Lanubile, F. and Scalas, M. (2007). Porting a distributed meeting system to the eclipse communication framework. In: *eclipse '07: Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange*, pp. 46–49. ACM, New York, NY, USA. ISBN 978-1-60558-015-9.

[8] Chun, A., Wai, H. and Wong, R. (October 2003). Optimizing agent-based meeting scheduling through preference estimation. *Engineering Applications of Artificial Intelligence*, vol. 16, pp. 727–743(17).
Available at: http://www.ingentaconnect.com/content/els/09521976/2003/00000016/00000007/art00116

[9] Chun, H.W. and Wong, Y.M. (2003). N*: an agent-based negotiation algorithm for dynamic scheduling and rescheduling. *Adv. Eng. Informatics*, vol. 17, no. 2003, pp. 1–22.

[10] Crawford, E. and Veloso, M. (2004 October). Opportunities for learning in multi-agent meeting scheduling. In: *Proceedings of the Fall AAAI Symposium on Artificial Multiagent Learning*. Washington, DC.
Available at: http://www.cs.cmu.edu/~mmv/papers/04aaais-liz.pdf

[11] Crawford, E. and Veloso, M. (2005). Learning dynamic preferences in multi-agent meeting scheduling. In: *IAT '05: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pp. 487–490. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-2416-8.

[12] Demirel, I.O. and Erdogan, N. (2007). Meeting scheduling with multi agent systems: design and implementation. In: *SEPADS'07: Proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, pp. 92–97. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA. ISBN 978-960-8457-59-1.

[13] Eugene C. Freuder, Marius Minca, R.J.W. (2001). Privacy/efficiency trade-offs in distributed meeting scheduling by constraint-based agents. *IJCAI-2001 Workshop on Distributed Constraint Reasoning*, vol. 1, pp. 63–71.

[14] Faulring, A. and Myers, B.A. (2005). Enabling rich human-agent interaction for a calendar scheduling agent. In: *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pp. 1367–1370. ACM, New York, NY, USA. ISBN 1-59593-002-7.

[15] Franzin, M.S., Freuder, E.C., Rossi, F. and Wallace, R. (2002 August). Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality. In: *Proc. AAAI 2002 workshop on Preference in AI and CP*, p. 25–32. Canada.

[16] Garrido, L. and Sycara, K. (1996). Multi-agent meeting scheduling: preliminary results. In: *1996 International Conference on Multi-Agent Systems (ICMAS '96)*, pp. 95 – 102.

[17] Gervasio, M.T., Moffitt, M.D., Pollack, M.E., Taylor, J.M. and Uribe, T.E. (2005). Active preference learning for personalized calendar scheduling assistance. In: *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pp. 90–97. ACM Press, New York, NY, USA. ISBN 1-58113-894-6.

[18] Google Inc. (2009). Google calendar. Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA.
Available at: http://www.google.com/calendar

[19] Google Inc. (2009). Google talk. Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA.
Available at: http://www.google.com/talk/

[20] Haralick, R. and Elliot, G. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, vol. 14, no. 3, pp. 263–313.

[21] Hassine, A.B. and Ho, T.B. (2007). An agent-based approach to solve dynamic meeting scheduling problems with preferences. *Eng. Appl. Artif. Intell.*, vol. 20, no. 6, pp. 857–873. ISSN 0952-1976.

[22] Haynes, T., Sen, S., Arora, N. and Nadella, R. (1997). An automated meeting scheduling system that utilizes user preferences. In: *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pp. 308–315. ACM, New York, NY, USA. ISBN 0-89791-877-0.

[23] IBM Corporation (2005). Lotus notes. IBM Corporation, Armonk, NY, USA. Available at: www.ibm.com/software/lotus

[24] Igniterealtime, Jive Software (2008). Open source xmpp (jabber) client library for instant messaging and presence. Ignite Realtime, Jive Software, 915 SW Stark St., Suite 400, Portland, OR 97205, USA.
Available at: http://www.igniterealtime.org/projects/smack/index.jsp

[25] Igniterealtime, Jive Software (2009). Openfire a real time collaboration (rtc) server. Ignite Realtime, Jive Software, 915 SW Stark St., Suite 400, Portland, OR 97205, USA.
Available at: http://www.igniterealtime.org/projects/openfire/index.jsp

[26] Internet Engineering Task Force (2004 October). Extensible messaging and presence protocol (xmpp): Core.
Available at: http://www.ietf.org/rfc/rfc3920.txt

[27] Kincaid, C.M., Dupont, P.B. and Kaye, A.R. (1985). Electronic calendars in the office: an assessment of user needs and current technology. *ACM Trans. Inf. Syst.*, vol. 3, no. 1, pp. 89–102. ISSN 1046-8188.

[28] Kozierok, R. and Maes, P. (1993). A learning interface agent for scheduling meetings. In: *IUI '93: Proceedings of the 1st international conference on Intelligent user interfaces*, pp. 81–88. ACM, New York, NY, USA. ISBN 0-89791-556-9.

[29] Lima, E.F.A., Machado, P.D.L., Sampaio, F.R. and Figueiredo, J.C.A. (2004). An approach to modelling and applying mobile agent design patterns. *SIGSOFT Softw. Eng. Notes*, vol. 29, no. 3, pp. 1–8. ISSN 0163-5948.

[30] Lin, S.-j. and Hsu, J.Y.-j. (2000 November). Learning user's scheduling criteria in a personal calendar agent. In: *Proceedings of TAAI2000*. Taipei,.

[31] Microsoft Corporation (2007). Microsoft exchange server. Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-7329, USA.
Available at: http://www.microsoft.com/exchange/2007/default.mspx

[32] Microsoft Corporation (2007). Microsoft outlook mobile. Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-7329, USA.
Available at: http://www.microsoft.com/windowsmobile/en-us/downloads/microsoft/office-outlook-mobile.mspx

[33] Microsoft Corporation (2007). Microsoft windows mobile. Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-7329, USA.
Available at: http://www.microsoft.com/windowsmobile/en-us/default.mspx

[34] Microsoft Corporation (2007). Outlook. Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-7329, USA.
Available at: http://office.microsoft.com

[35] Microsoft Corporation (2008). Messaging application programming interface (mapi). Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-7329, USA.
Available at: http://msdn.microsoft.com/en-us/library/ms529058.aspx

[36] Mitchell, T.M., Caruana, R., Freitag, D., McDermott, J. and Zabowski, D. (1994). Experience with a learning personal assistant. *Commun. ACM*, vol. 37, no. 7, pp. 80–91. ISSN 0001-0782.

[37] Modi, P.J., Veloso, M., Smith, S. and Oh, J. (2005). Cmradar: A personal assistant agent for calendar management. In: *Agent-Oriented Information Systems II, Lecture Notes in Computer Science*, vol. 3508/2005, pp. 169–181. Springer-Verlag.

[38] Morley, D. and Myers, K. (2004). The spark agent framework. In: *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 714–721. IEEE Computer Society, Washington, DC, USA. ISBN 1-58113-864-4.

[39] Mynatt, E. and Tullio, J. (2001). Inferring calendar event attendance. In: *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pp. 121–128. ACM Press, New York, NY, USA. ISBN 1-58113-325-1.

[40] Palm, Inc. (). Palm operating system. Palm, Inc., 950 W. Maude Ave., Sunnyvale, CA 94085, USA.
Available at: http://www.palm.com/us/

[41] Sen, S. and Durfee, E. (1994 Mar). On the design of an adaptive meeting scheduler. *Artificial Intelligence for Applications, 1994., Proceedings of the Tenth Conference on*, pp. 40–46.

[42] Sen, S. and Durfee, E.H. (1991). A formal study of distributed meeting scheduling: preliminary results. *SIGOIS Bull.*, vol. 12, no. 2-3, pp. 55–68. ISSN 0894-0819.

[43] Sen, S., Haynes, T. and Arora, N. (1997). Satisfying user preferences while negotiating meetings. *Int. J. Hum.-Comput. Stud.*, vol. 47, no. 3, pp. 407–427. ISSN 1071-5819.

[44] SQLite (2009). Embedded sql database engine. Hwaci, 6200 Maple Cove Lane, Charlotte, NC 28269, USA.
Available at: http://www.sqlite.org/

[45] The Open Handset Alliance (2007 November). Android. Google Inc., 1600 Amphitheatre Parkway, Mountain View, California, 94043 USA.
Available at: http://www.openhandsetalliance.com/android_overview.html

[46] Tsuruta, T. and Shintani, T. (2000). Scheduling meetings using distributed valued constraint satisfaction algorithm. In: *In Proceedings 14th ECAI*, pp. 383–387.

[47] Tullio, J., Goecks, J., Mynatt, E.D. and Nguyen, D.H. (2002). Augmenting shared personal calendars. In: *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pp. 11–20. ACM, New York, NY, USA. ISBN 1-58113-488-6.

[48] Tungle Corporation (2008 July). Tungle. Tungle Corporation, 145 rue St-Pierre, Suite 108, Montreal, Quebec, Canada.
Available at: http://www.tungle.com

[49] Wainer, J., Paulo Roberto Ferreira, J. and Constantino, E.R. (2007). Scheduling meetings through multi-agent negotiations. *Decis. Support Syst.*, vol. 44, no. 1, pp. 285–297. ISSN 0167-9236.

[50] Yokoo, M. and Hirayama, K. (1998). Distributed constraint satisfaction algorithm for complex local problems. In: *ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems*, p. 372. IEEE Computer Society, Washington, DC, USA. ISBN 0-8186-8500-X.