Santa Clara University
DEPARTMENT of COMPUTER ENGINEERING

Date: June 9, 2006

I HEREBY RECOMMEND THAT THE THESIS
PREPARED UNDER MY SUPERVISION BY

Kristen Moss & Caroline Ratajski

ENTITLED

Palm Meeting Scheduler

BE ACCEPTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER ENGINEERING

THESIS ADVISOR

DEPARTMENTCHAIR

Palm Meeting Scheduler

Written By:

Kristen Moss
Caroline Ratajski

June 11, 2006
Version 3.1

Table of Contents

Introduction

This document is in intended to describe the problem we are trying to solve and our solution to it. It will also describe the basic requirements for success of our project.

Terminology

Term	Definition
AAM	Agendus Attendees Module (the add-on being created within this project)
AGM	Agendus Mail
AGP	Agendus Palm
Attendee	User interacting with project from the reference point of one who is being invited to and will potentially accept/decline the invitation for a meeting (as opposed to Organizer)
Organizer	User interacting with project from the reference point of one who is organizing and inviting others to the meeting. Controls date, time, location, and purpose of the meeting. Also will determine whether meeting is to be cancelled or to proceed as scheduled. (as opposed to Attendee)
PDA	Personal Data Assistant
SMS	Short Message Service (also known as “text messaging”)

Overview

The Problem

The process of scheduling a meeting between two or more people can become very tedious. Between the use of email, cell phones, PDA’s, and possible schedule conflicts - simple to work around when organizing very few people - organization becomes nearly unmanageable on a large scale. Solutions for this problem exist in such applications as Microsoft Exchange, but this is costly and difficult to maintain, and an impractical tool for smaller companies and startups. Also, many people do not wish to share their entire schedules. It would be nice to be able to share only the time that people have free. And what of those not in the server? To put people into a server requires a great deal of setup that usually must be done by the system administrator. There should be a simpler way to solve such an issue.

The Solution

iambic, Inc. has a program called Agendus for PalmOS, which makes scheduling and organizing information in Palm Handhelds simpler and easier. Our plan is to add functionality to this program in order to allow a dialogue of sorts between meeting coordinators and attendees. This dialogue would be conducted without letting the coordinator see the schedule of each of the attendees. Instead, the attendee would see the request for a meeting and also whether or not he/she already has an appointment scheduled for that time. From there, the attendee could choose to either accept or decline, without having to look up that time and date in his schedule.

Benefits of the Solution

Meeting organizers are able to quickly and easily invite attendees and are notified of their attendance status. The meeting attendees are able to accept or decline meetings while away from their computers, without having the privacy of their schedules compromised. Also, those without Agendus are still able to receive notifications about the meeting, regardless of their own lack of PDA. If they decide to convert to PDA/Agendus, the setup of our system is simpler and cheaper than creating and maintaining a server to hold everyone's schedules. Overall, organization among businesses and people is made much easier by the use of this tool.

Stakeholders

iambic, Inc.

iambic has agreed to give us access to the source code of Agendus and Agendus Mail, and to allow us to enhance and manipulate these products. iambic will be able to use whatever we produce in order to better their existing product. iambic has reviewed this document and agreed with everything written. For more information on iambic, visit http://www.iambic.com (valid as of June 3, 2006).

Kristen Moss and Caroline Ratajski

The above students have taken this project as their senior design project and will see it to its completion as outlined within this document. Failure to do so will place their graduation status in severe jeapordy.

Dr. John Noll

Despite not having any financial or career-oriented investment in this project, Dr. Noll has agreed to oversee it and requires a fully completed project by the end. He is devoting a great deal of time and energy into the success of this project, in the form of advising and meeting with the students (Kristen Moss and Caroline Ratajski). Failure on the part of the students to deliver the project will result in his failing them.

Scenario

Professor Hill wants to call in three of his students for a meeting concerning a project they have been working on. He has been told that the date for their final presentation has been postponed a week due to a problem in the university's scheduling, and he has to let his students know and discuss with them the potential impact this has on their project. He opens up his Personal Data Assistant (PDA) which runs using the Palm Operating System, looks up their information, types an email explaining the meeting, and the date and time he would like to meet them, and sends the email, hoping for the best. These three students, Christina, Brad, and Justin, check their email, open their own schedules, look up their free time, email in response whether or not they can attend, and wait on Professor Hill's confirmation.

Let us try this scenario. When Professor Hill opens his PDA, he accesses Agendus, and with a few taps of the pen he has created a meeting in his schedule. He has selected from his contacts the three students he wishes to have attend, and sends the meeting message. Instead of this message going to their email boxes, it is transferred via SMS and arrives at their PDAs or cellular phones. Agendus will intercept the SMS message and display it in a user-friendly format.

Christina happens to have a PDA with Agendus. The message arrives and she is notified of the potential meeting. Agendus looks at the schedule she keeps in her device and notifies her that she has that time free. She accepts the meeting with the click of a button, and a message is sent to Professor Hill, automatically updating his meeting attendees.

Brad also has a PDA with Agendus as well, but his time isn't free. Agendus will let him know and allow him to either decline the meeting invitation or reschedule his currently scheduled task. Agendus will do this without ever letting Professor Hill know that the reason Brad isn't free is because he has a date with his girlfriend.

Justin does not have a PDA, but he does have a cell phone with SMS functionality, exceedingly common today. He will see this message and be able to contact Professor Hill by other means to either accept or decline the meeting invitation.

Once Professor Hill has coordinated fully with his students, he confirms the meeting. All this has been done without the hassle and delay that the process took in the first scenario.

User Manual

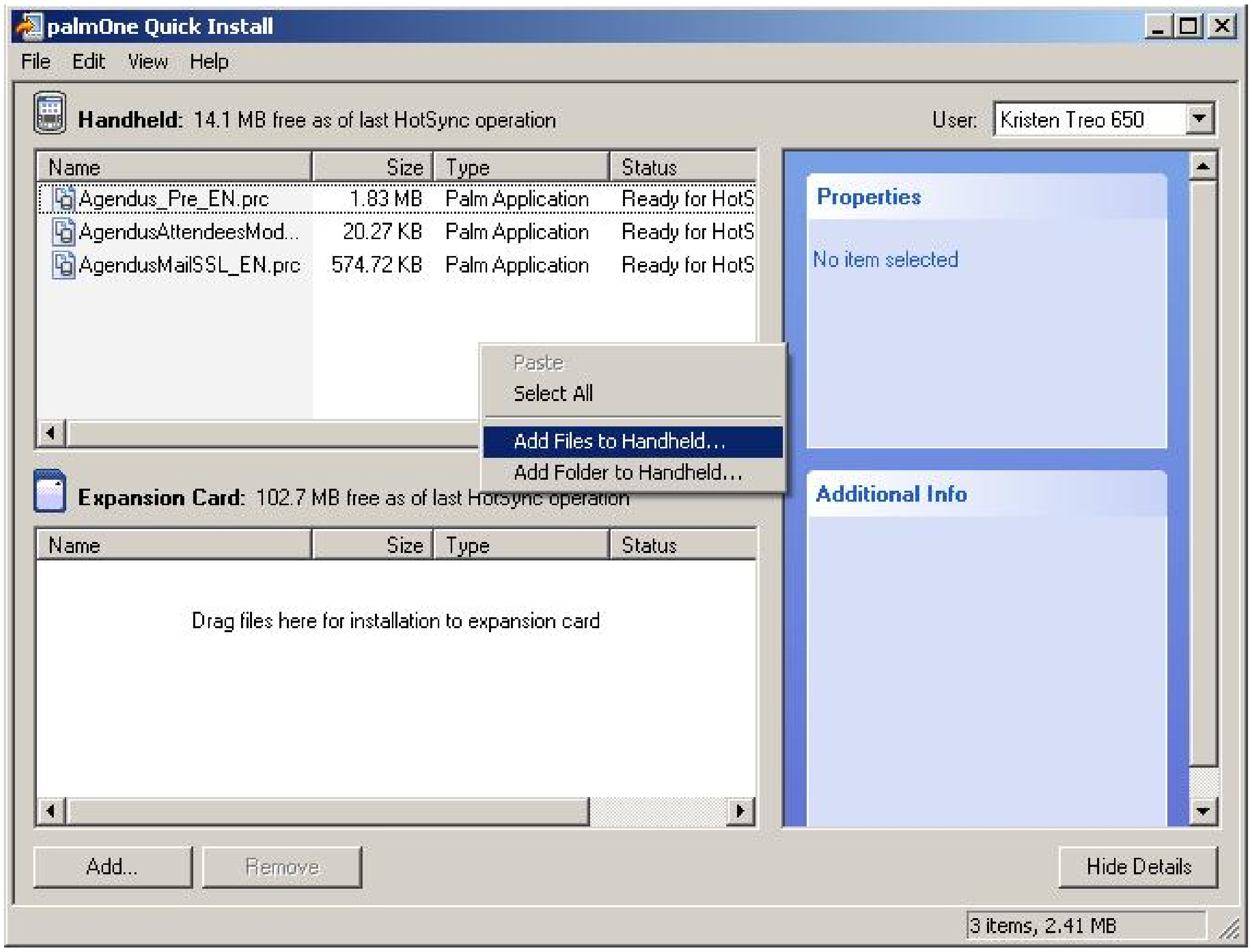
Installation

Requires: SMS-capable handheld device with PalmOS 5.0 or greater installed and functional; PC with Windows operating system installed and functional, 2000 or greater.

1. Connect your handheld device to your PC using cables provided with the device.

2. Open the “PalmOne Quick Install” application on your PC.

3. Right-click on the white section underneath “Handheld” on the PalmOne Quick Install application and select “Add Files to Handheld”. Locate on the CD and add the following PRCs:
   - AgendusAttendeesModule.prc
   - AgendusMailSSL_EN.prc
   - AgendusPre_EN.prc



palmOne Quick Install

File Edit View Help

Handheld: 14.1 MB free as of last HotSync operation

User: Kristen Treo 650

Name	Size	Type	Status
Agendus_Pre_EN.prc	1.83 MB	Palm Application	Ready for HotS
AgendusAttendeesMod...	20.27 KB	Palm Application	Ready for HotS
AgendusMailSSL_EN.prc	574.72 KB	Palm Application	Ready for HotS

Expansion Card: 102.7 MB free as of last HotSync operation

Name	Size	Type	Status

Drag files here for installation to expansion card

Properties

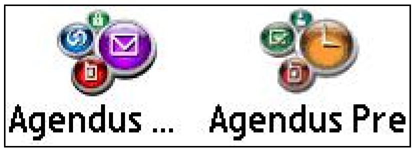
No item selected

Additional Info

3 items, 2.41 MB

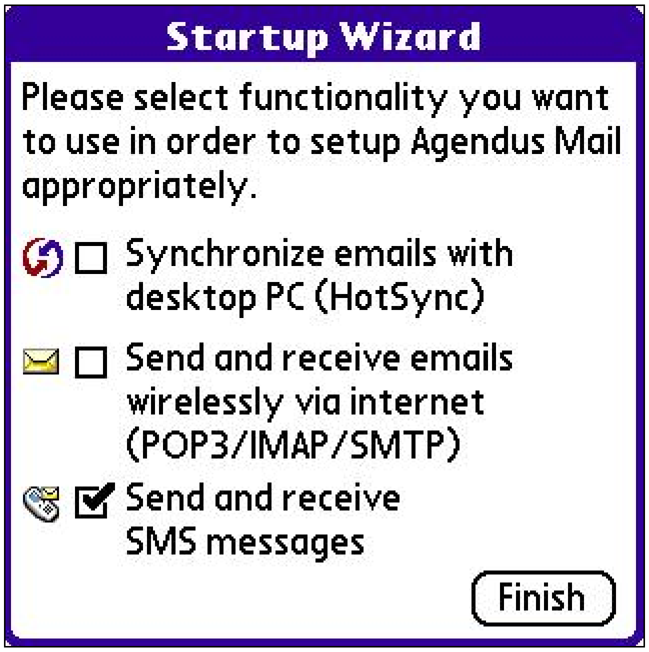
4. Push the Hard Button on the connector to your handheld device.

5. Wait for the HotSync to complete. Once installation is complete, you should see the following two icons:



Note that you should not see any icon for the AgendusAttendeesModule.

6. In order to get the application fully ready to run, you must set up AGM so that it accepts SMS messages. Open AGM and make certain that the checkbox for "Send and receive SMS messages" is selected, as below (if you wish to select the other two, you may; it will not affect this application):



Startup Wizard

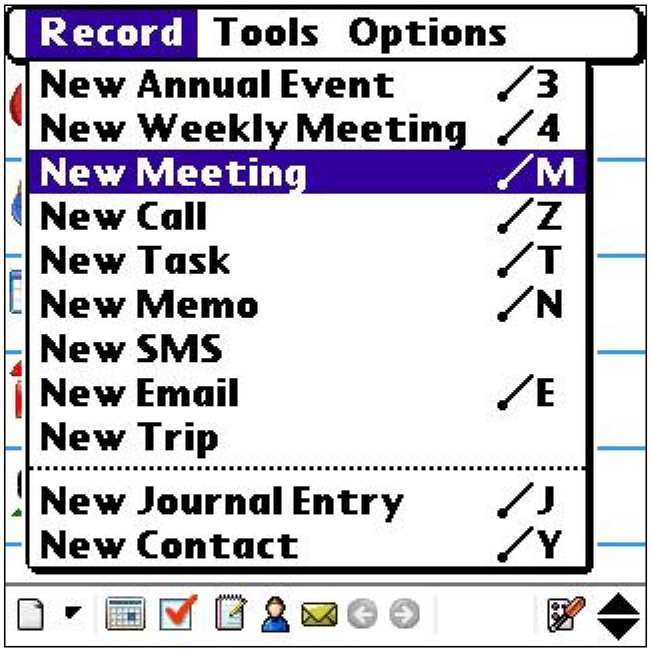
Please select functionality you want to use in order to setup Agendus Mail appropriately.

- ☐ Synchronize emails with desktop PC (HotSync)
- ☐ Send and receive emails wirelessly via internet (POP3/IMAP/SMTP)
- ☑ Send and receive SMS messages

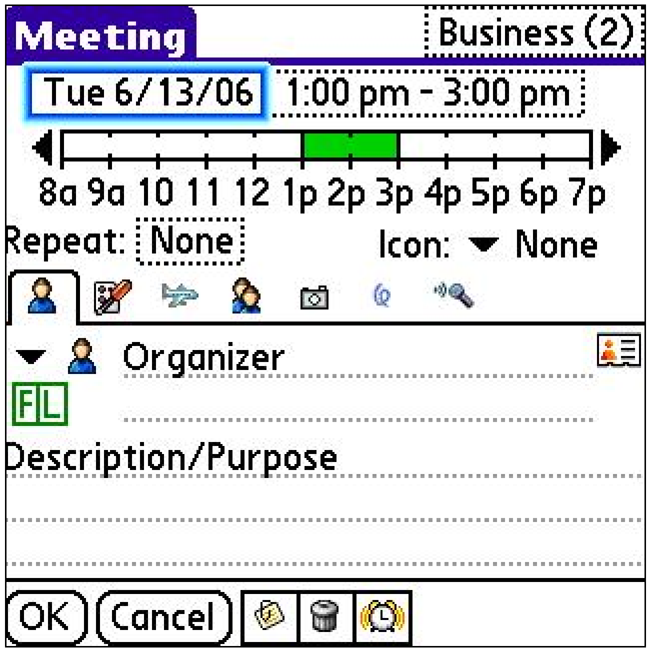
Click "Finish" and you're ready to go!

Create a Meeting and Add Attendees

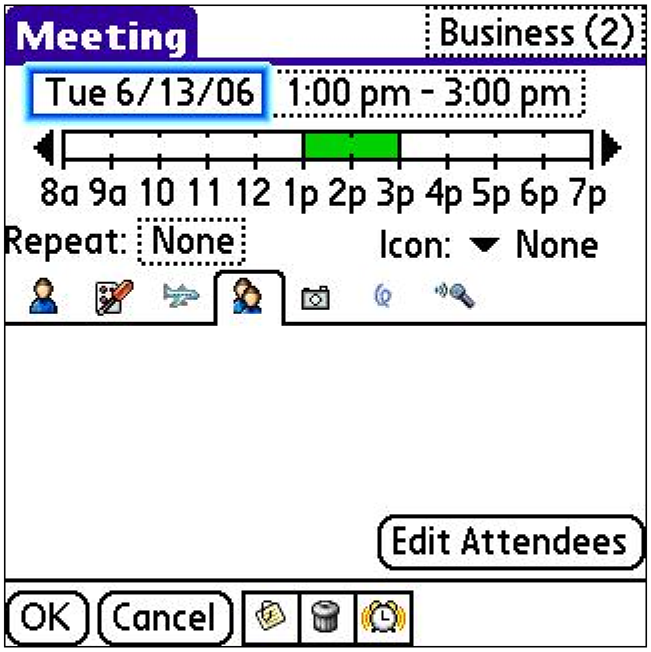
1. To create a new meeting, click the date dialogue at the top and select "New Meeting" (you may also select "New Annual Event" or "New Weekly Meeting").



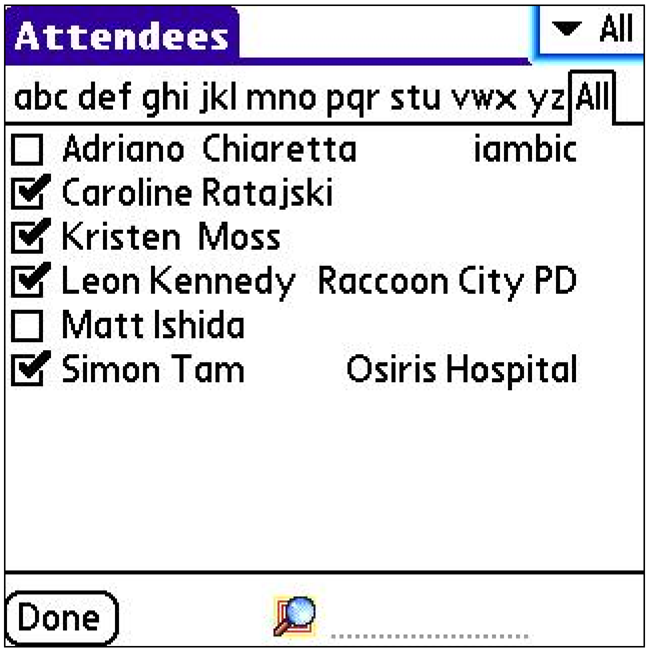
2. Once you have opened a new meeting, fill it with pertinent information, such as the time, date, location, and who will be organizing the event.



3. Now that you have created your meeting, if you wish to add attendees, select the attendees tab as shown below, and click "Edit Attendees."

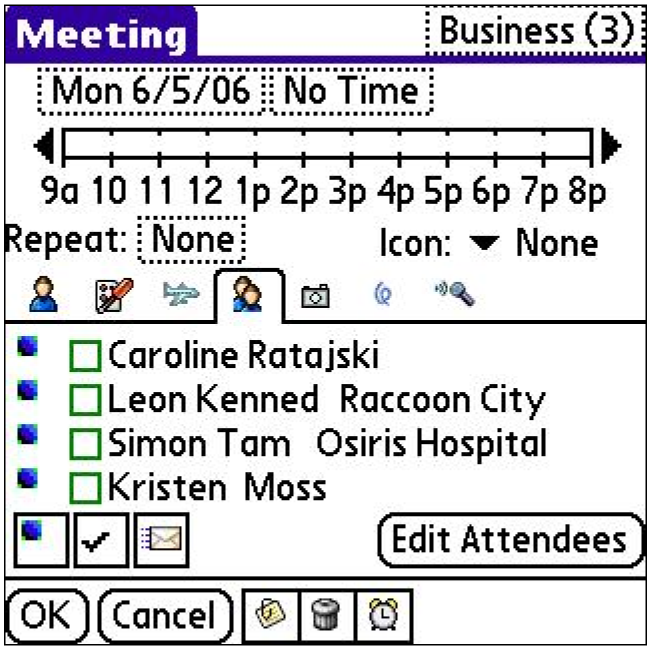


4. Select which contacts you wish to attend this meeting by checking the checkbox next to their name. Once you are satisfied with your attendee list, click "Done."



- [ ] Adriano Chiaretta iambic
- [x] Caroline Ratajski
- [x] Kristen Moss
- [x] Leon Kennedy Raccoon City PD
- [ ] Matt Ishida
- [x] Simon Tam Osiris Hospital

5. You should now see your selected attendees in your new meeting's attendee list, as pictured below:


Meeting Business (3)

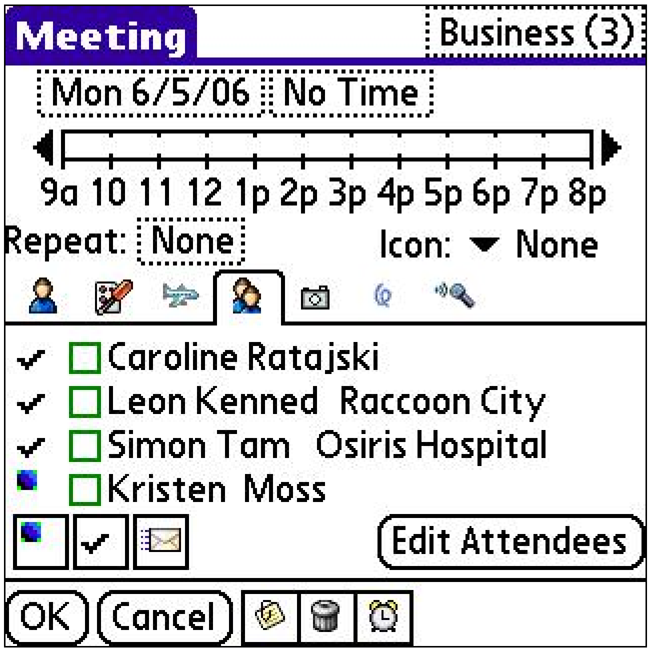
Mon 6/5/06 No Time

9a 10 11 12 1p 2p 3p 4p 5p 6p 7p 8p

Repeat: None Icon: None

- Caroline Ratajski
- Leon Kenned Raccoon City
- Simon Tam Osiris Hospital
- Kristen Moss

6. To send an invite, make certain that you have selected all of the attendees you wish to invite, either by clicking on a singular dot to check them, or by using the “select all” function, which is the checkmark button beneath the attendee list.


Meeting Business (3)

Mon 6/5/06 No Time

9a 10 11 12 1p 2p 3p 4p 5p 6p 7p 8p

Repeat: None Icon: None

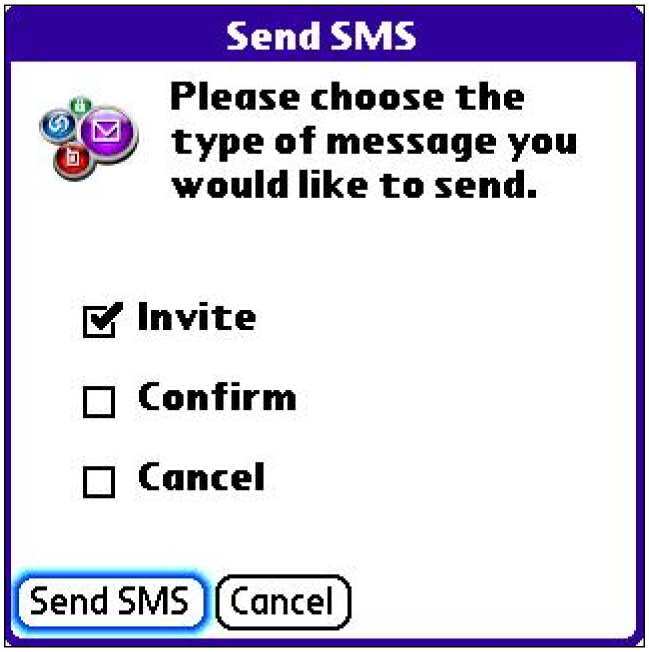
- ✓ Caroline Ratajski
- ✓ Leon Kenned Raccoon City
- ✓ Simon Tam Osiris Hospital
- Kristen Moss

At any time, you can add new attendees by clicking “Edit Attendees” – your previously selected attendees will remain on the list.

Inviting Attendees

1. Once you have selected which attendees you wish to send a message to, click the little envelope icon to bring up the "send message" dialogue.

2. In the following dialogue, select "Invite," "Confirm," or "Cancel," depending on the type of message you wish to send, and then click "Send SMS." This will send an SMS message from your mobile device to the mobile phone numbers associated with each Attendee you have selected.



If you are satisfied with your Attendee List and wish to invite these Attendees, click the "Invite" button.

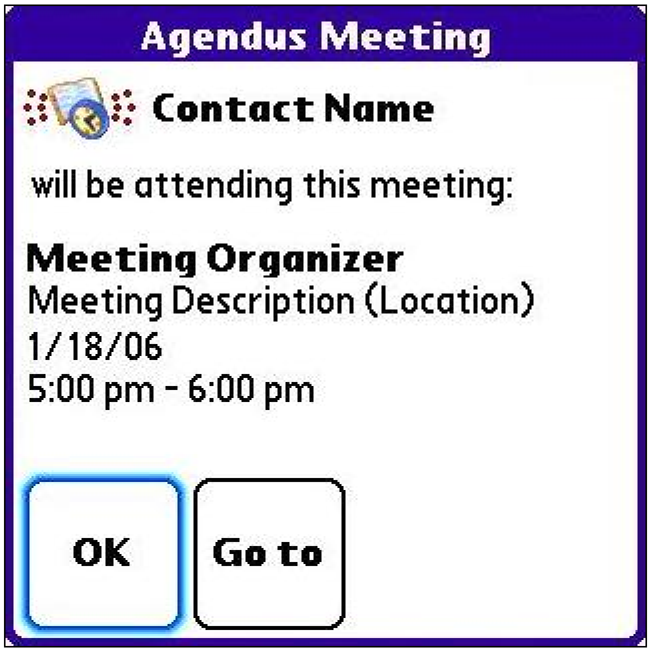
<u>Invite</u>: Sends an invitation message to the Attendee(s), alerting them to a new meeting.

<u>Confirm</u>: Sends a confirmation message to the Attendee(s), letting them know that the meeting has been confirmed. Also updates the status of your own meeting as confirmed.

<u>Cancel</u>: Sends a cancellation message to the Attendee(s), letting them know that the meeting has been cancelled. Also updates the status of your own meeting as cancelled.

And don't worry if you send off the message and realize later that you wish to add more attendees. You can always go back and add more attendees after the message has been sent and select to send a message to them specifically, using the method described previously.

3. And now all you will do is wait for messages to come, alerting you as to who is attending or not. You will periodically receive dialogues such as the one below:


Agendus Meeting

Contact Name

will be attending this meeting:

Meeting Organizer
Meeting Description (Location)
1/18/06
5:00 pm - 6:00 pm

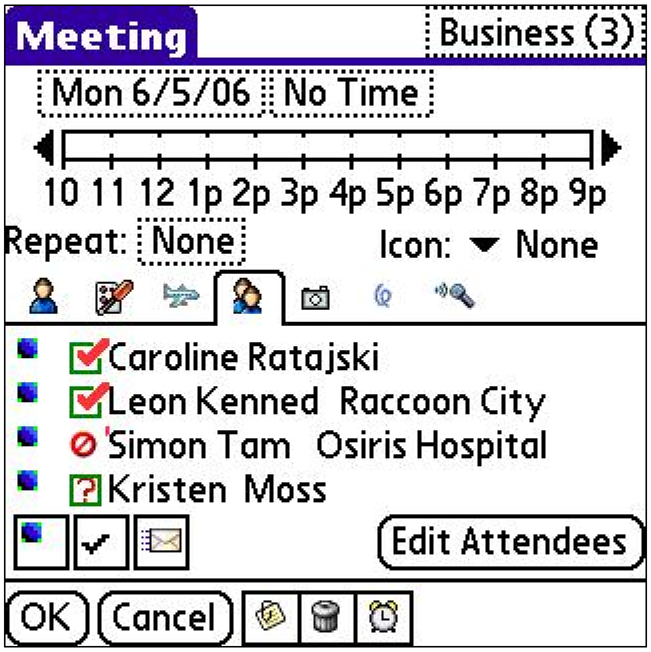
OK: Merely accepts the attendee, updates their status in the Attendee Tab, and returns you to what you were doing before.

Go To: Goes to the meeting, where you will see that the Attendee's status has been updated to reflect whether or not they are attending.

Once you know whether or not you wish to confirm or cancel this meeting, merely return to the dialogue outlined in step three, and select either the "Confirm" or "Cancel" message to be sent.

Attendee Management

As mentioned in the previous section, you will periodically receive messages alerting you as to whether or not your attendees are able to come. Once you start receiving a few responses, your attendee list should start to look like this:



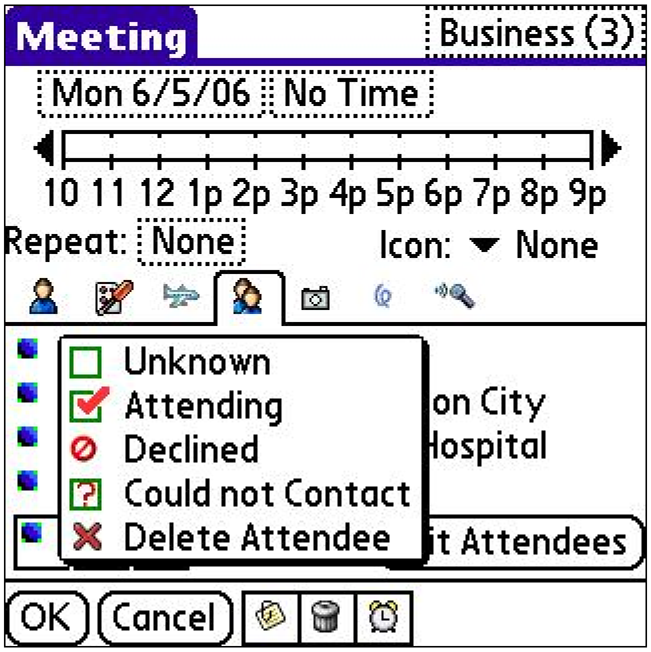
Meeting | Business (3)

Mon 6/5/06 | No Time

Repeat: None   Icon: None

✓ Caroline Ratajski
✓ Leon Kenned Raccoon City
⊘ Simon Tam Osiris Hospital
? Kristen Moss

If you ever wish to modify an attendee's status, merely click on the status icon just to the left of their name and modify it, like in the image below.



Meeting | Business (3)

Mon 6/5/06 | No Time

Repeat: None   Icon: None

Unknown: The Attendee has not responded yet, but was able to be contacted.

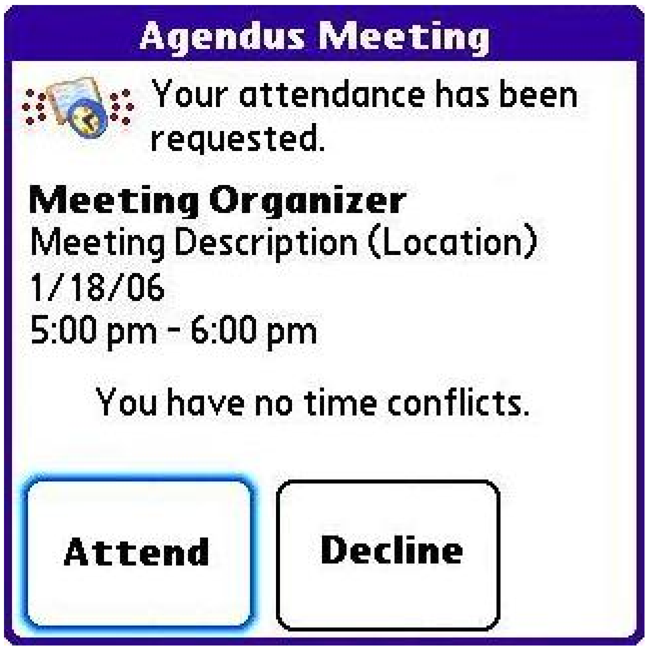
Attending: The Attendee has responded that they will be at the meeting.

Declined: The Attendee has responded that they will *not* be at the meeting.

Could Not Contact: There is no valid Mobile number for the attendee.

How to Respond to a Meeting Message

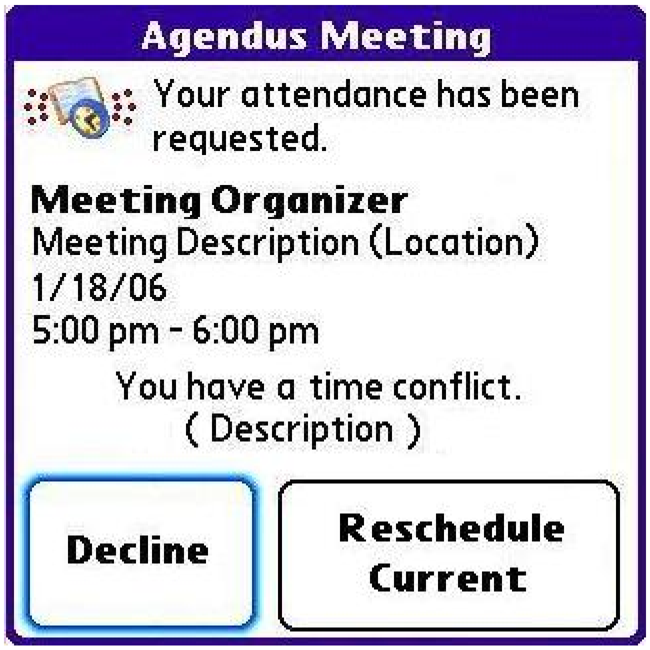
If you receive a message as an attendee, inviting you to a meeting, one of the two following dialogues will come up:


Agendus Meeting

Your attendance has been requested.

Meeting Organizer
Meeting Description (Location)
1/18/06
5:00 pm - 6:00 pm

You have no time conflicts.


Agendus Meeting

Your attendance has been requested.

Meeting Organizer
Meeting Description (Location)
1/18/06
5:00 pm - 6:00 pm

You have a time conflict.
(Description)

<u>Attend</u>: This will add the meeting to your schedule, as well as send a message to the Organizer, alerting them that you are going to attend.

Decline: This will merely send a message back to the Organizer, alerting them that you are not going to attend the meeting.

Reschedule Current: This will open the currently conflicting meeting within your own calendar. Upon rescheduling this meeting, a message will be sent to the organizer, alerting them that you are going to attend.

Use Case

The use case defines the method by which the user (Organizer and Attendee) will utilize the project. It explains the process, not in terms of how each user sees it, but in how the entire operation works. This is due to the fact that separating Organizer process from Attendee process would make the entire operation harder to understand.

Pre-process requirements: Organizer has both AGP and AGM installed on palm device, as well as the AAM, and has SMS service.

1. Organizer opens AGP on palm device and generates a meeting, complete with date, time, location, title/reason for meeting, and list of Attendees.

2. Organizer selects to send invite to all Attendees on attendee list.

3. If the Organizer does not have the Attendee's phone number, the inability to contact the Attendee is reflected in the attendee list, by italicizing the Attendee's name.

4. If the Organizer does have the Attendee's phone number, but that phone number is not linked to a SMS-capable device (for instance, a home phone number) the SMS service will send back a failed message to the Organizer, which will not be handled by AAM but rather by the pre-existing SMS service.

5. If the Organizer does have the Attendee's phone number, and said phone number is linked to a SMS-capable device, but the Attendee lacks AGM and AGP on their device, the Attendee will merely receive the text message as-is. The text message will be in a readable format so that the Attendee can contact the Organizer through alternate means (non-AAM SMS message, email, phone call, etc) and let them know whether they wish to accept or decline the invitation.

6. If the Organizer does have the Attendee's phone number, and said phone number is linked to a SMS-capable device, and the Attendee has AGM and AGP on their device, the program will alert the Attendee as to whether or not they are free during that time period.

7. If the Attendee is free and wishes to attend, the Attendee will click "Accept"

8. If the Attendee is free, but does not wish to attend, the Attendee will click "Decline"

9. If the Attendee is not free, the AAM will ask the Attendee if they would like to reschedule their current obligation.

10. If the Attendee does wish to reschedule their current obligation, they will be taken to the current obligation and be allowed to do so. Then it will ask them to click "Accept" for the meeting.

11. If the Attendee does not wish to reschedule their current obligation, they will click "Decline"

12. The Organizer will receive messages as to whether or not Attendees with AGM and AGP wish to attend the meeting or not. The list will update itself without the Organizer needing to do anything (names will be bolded if the Attendee is planning on attending, strikeout if the Attendee is not going to attend).

13. The Organizer will look over the attendee list, and if they are not satisfied with the list of attendees (not enough people are able to attend, or a critical person for the meeting is missing) they will click "Reschedule Meeting." A dialogue will open, allowing them to reschedule the time and date of the meeting. Upon rescheduling, they will click "Resend Invite" and the process will begin once more from step 3.

14. The Organizer will look over the attendee list, and if they are satisfied with the list of attendees, they will click "Confirm Meeting." The Attendees will see that the meeting has been confirmed.

Success Requirements

This is a brief section of the document outlining the specific functional requirements which define the success of the project. In order for this project to be considered a success, the program must be verified by Adriano Chiaretta (representing iambic) and both team members (Kristen Moss and Caroline Ratajski).

General Requirements

Here is a short list of functionality and features that, if completed, would define the success of the project.

Required

	Title	Deadline
1.	**Message Creation** The SMS messages must be readable by those who do not have Agendus, and only have normal SMS capability.	**December 23, 2006**
2.	**Message Creation** The person scheduling the meeting (coordinator) must be able to send an invitation to all of the Attendees in the form of an SMS (text message) using the existing Agendus and Agendus Mail programs.	**December 23, 2006**
3.	**Message Creation** The SMS messages must be readable by those who do not have Agendus, and only have normal SMS capability.	**December 23, 2006**
4.	**Recognizing/Intercepting SMS** Pre-existing AGM will be modified to recognize and redirect AAM-generated messages that they may be handled correctly, while passing all non-AGM messages	**January 13, 2006**
5.	**Parsing Messages into Useful Information** Messages directed to AAM by AGM will be parsed into useful information, such as organizer, date, time, and location.	**February 3, 2006**
6.	**Check Attendee for Availability** AAM must interface with AGP to utilize pre-existing 'free time checker' to see if Attendee is available for the incoming meeting message.	**February 17, 2005**
7.	**Alert for Attendee: Accept/Decline/Reschedule** If AGP and AGM are present on the Attendee's SMS-receiving device, the program must intercept the SMS and display it using the pre-existing AGP user interface; this interface must allow the user to accept the meeting, decline it, or reschedule any conflicting appointments.	**March 10, 2005**

8. **Auto-Add New Appointment** **March 10, 2005**
   If accepted by the Attendee, the meeting must be automatically inserted into the Attendee's schedule.

9. **Message Creation & Alert for Attendee** **March 10, 2005**
   Once the response to the meeting (accepted or declined) has been recorded, an SMS message must be generated and sent to the coordinator.

10. **Updating Organizer Attendee List** **March 17, 2006**
    The SMS received by the coordinator must be interpreted by Agendus and alter the Attendee's confirmation status accordingly, to reflect whether they have accepted or declined the invitation.

11. **Creating Messages** **April 7, 2005**
    Upon the responses of the Attendees, the coordinator should be able to either confirm that meeting or reschedule it to allow more people to come.

12. **Overall Acceptance** **April 30, 2006**
    Any and all implemented features must be correct as defined by the specification and must be documented well enough to make the project reproducible and expandable.

13. **Overall Acceptance** **April 30, 2006**
    The program must install and run on Palm devices with SMS capability and Palm OS 5.0 or better.

14. **Overall Acceptance** **April 30, 2005**
    The project must be stitched into the existing program.

For Future Consideration

15. **Auto Confirm/Cancel** **Completed**
    After the meeting has been scheduled and confirmed, the Attendees should receive a confirmation message that either flags the meeting as confirmed or automatically reschedules the meeting on their own PDA. Also, upon sending a confirm/cancel message, the organizer's own meeting information should reflect that the meeting has been confirmed or cancelled

16. **Manual Changing of Attendee Status** **Completed**
    The organizer can manually change the status of any attendee at any time. This allows for people who do not have Agendus to respond by other means, as well as permit those who have already responded to change their plans.

17. **Automatic Contact Verification/Addition** **Completed**
    When a message comes in, the phone number of the sender gets looked up in your contact database. If the number is found, the display name gets changed to what is in

your database, allowing you to be familiar with the name of the organizer no matter what. If the number is not found, a new record will be added to your contact database with the name and phone number of the organizer.

18. **Seamless Stitching Into Existing Applications** **Completed**
Since the design presentation we have reworked the UI slightly to conform to exactly what iambic desires from one of its own applications.

19. **Select Specific Attendees to Send Message To** **Completed**
Since the design presentation we have reworked the UI slightly to conform to exactly what iambic desires from one of its own applications.

20. **Quick-Deletion of Attendees** **Completed**
Ability to quick-delete attendees from attendee list, as opposed to having to go into directory view, which causes the Organizer to leave the attendee list dialogue and enter a completely different dialogue.

21. **Version/License Checking** **Completed**
At any time, the organizer can choose to send a message to any group of attendees that he/she wishes. This allows the organizer to add attendees later without having to send the message to everyone again, and it also permits the resending of messages to specific people (such as people who haven't responded).

22. **Backwards-Compatibility**
Application should run on older versions of the Palm OS and on older handheld devices.

23. **Key Attendees**
Organizer should be able to deem specific Attendees as 'key attendees' (if they decline the meeting invitation, the coordinator will be prompted to reschedule the meeting).

24. **Priority Matrix**
Organizer should be able to determine the importance of the meeting utilizing a priority/urgency matrix.

25. **Email Messages**
Allow the user to choose between sending messages via email or SMS. This would provide more flexibility to the system and allow us to send more information about the meeting that we are allowed to do in 160 characters of text message.

Technical Details

Programming Language

The bulk of the programming was done in C, with some sections done in C++.

Programming/Debugging Environment

Metrowerks Codewarrior was used for the programming and debugging environment. For more information on Metrowerks Codewarrior, visit http://www.metrowerks.com (valid as of June 3, 2006).

Code Repository

Source Off-Site (SOS) was the code repository used, the central server for which was hosted by iambic. SOS is a free download available at http://www.sourcegear.com/sos/ (valid as of June 3, 2006).

Bug Tracking System

Bugtrack is a web-based system for tracking bugs in a software program. As soon as a developer or tester finds a bug in the program, he/she goes to the website, logs in, and reports it. From there, the project manager can assign a bug to a certain developer. Once the developer fixes that particular bug, he/she sets it as resolved. Then the QA department goes through and double checks the resolutions, and reopens any bugs that did not get fixed.

Operating System

We programmed for the Palm OS, due to the fact that the application we were building and applications we were working within were all designed for Palm OS capable devices. Working with this operating system will be detailed later in the document, in the section labeled "Features of the Palm Operating System."

Testing

The testing that was done by Caroline Ratajski and Kristen Moss was performed on the Treo650 Device, and a Treo650 Simulator which operated on a PC running WindowsXP Professional. iambic's own in-house testing is done on a variety of other handhelds, but for the purpose of the project, only these two testing mechanisms were used.

Release Plans and Statistics

Release Plans	
Our Own Testing	Ongoing, through Release
iambic QA	May 7th, 2006
Private Beta	May 14th, 2006
Public Beta	May 21st, 2006
Release Date	June 13th, 2006

Statistics	
Lines of Code	4692
Files	20
Combined Work Hours	~600
Bugs/Features Reported and Solved	67
Tests Passed	124

Test Plan

This section of the document describes testing which was performed at each stage of the project's development. These sections work in conjunction with the success requirements.

SDP Test Plan

Test Platform: Version 3.5
Test Date: 4/24/06

Note: All tests must be performed on the sim AND on the device

Section 1: Creating Messages

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Create a Meeting	Empty Meeting on 6/14/06 from 1:00-2:00	#$AgendusMeeting$# <blankline> 6/14/2006 13:00-14:00 #061410213#		
Create a Meeting	Full Meeting on 6/14/06 from 1:00-2:00, with description = Description, organizer = Organizer, location = Location	#$AgendusMeeting$# Organizer 6/14/2006 13:00-14:00 Desc:Description Loc:Location #061410213#		
Create a Meeting	No Description	No description field		
Create a Meeting	No Organizer	Blank line where organizer should be		
Create a Meeting	No Location	No location field		
Create a Meeting	No Time	Time field shown as: ??-??		
Create a Meeting	Specified Time	Time field shown in 24hr time		

Create a Meeting	No Attendees	Button should not even appear		
Create a Meeting	No Attendees have Phone #'s	Warning should pop-up, not allowing the message to be sent		
Create a Meeting	Some Attendees have Phone #'s	Send only to these attendees, others denoted as Could Not Contact		
Create a Meeting	All Attendees have Phone #'s	Send to all attendees		
Create a repeating meeting	Send invite	A meeting UID will be generated and added to the notes of the meeting. This should only be done on the current instance of the repeating meeting, and an exception for this meeting will be generated.		
Create a Meeting, send invite.	Assign repeat details to this meeting	The meeting UID should only be kept in the current instance. To do this, the meeting will be assigned the repeat details without the UID, then an exception will be generated that has the UID.		
Create a Meeting	Confirm Message	#$AgendusConfirm$# Organizer 6/14/2006 13:00-14:00 Desc:Description Loc:Location #061410213#		
Create a Meeting, send invite, attendee then sets repeat details for that meeting	Confirm Message	Only confirm the specific instance for which the message refers to.		
Create a Meeting	Cancel Message	#$AgendusCancel$# Organizer 6/14/2006 13:00-14:00 Desc:Description Loc:Location #061410213#		
Create a Meeting, send invite, attendee then sets repeat details for that meeting	Cancel Message	Only cancel/delete the specific instance for which the message refers to.		
Create a Meeting, send invite, attendee presses accept	Response Message	#$AgendusResponse$# YES #061410213#		
Create a Meeting, send invite, attendee presses decline	Response Message	#$AgendusResponse$# NO #061410213#		

Create a repeating meeting, send invite	Response Message	Only change the attendee's status in the specific instance for which the message refers to.		

Section 2: Recognizing/Intercepting Messages

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Send Text Message	Invite	Intercept		
Send Text Message	Response	Intercept		
Send Text Message	Cancel	Intercept		
Send Text Message	Confirm	Intercept		
Send Text Message	Meeting Tag Only	Intercept		
Send Text Message	Partial Meeting Tag	Do not intercept		
Send Text Message	Normal SMS message	Do not intercept		

Section 3: Parsing Messages Into Useful Information

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**

Send Text Message	Invite - all fields filled	All fields parsed correctly		
Send Text Message	Invite - no organizer	Organizer = NULL		
Send Text Message	Invite - no description	Description = NULL		
Send Text Message	Invite - no location	Location = NULL		
Send Text Message	Invite - no time	Time struct filled in with a value of -1		
Send Text Message	Invite - with time	Time struct filled in with appropriate time in 24 hr time		
Send Text Message	Invite - bare bones (min fields)	see above 5, all in conjunction		
Send Text Message	Confirm	Meeting UID parsed correctly, CancelConfirm field is 1 (denotes confirm message)		
Send Text Message	Cancel	Meeting UID parsed correctly, CancelConfirm field is 0 (denotes cancel message)		
Send Text Message	Response - yes	Meeting UID is parsed correctly, Response field is 1 (positive)		
Send Text Message	Response - no	Meeting UID is parsed correctly, Response field is 0 (negative)		
Send Text Message	Tag Only	Message is discarded		

Use Parse Simulator	No Tag present	Shouldn't even be able to get this, but if so, message is discarded		

Section 4: Schedule Checking

Setup/Description	Input	Expected Output	Output	Pass/Fail
Create Meeting: Today, 8:00-9:00	Same Day, 8:00-9:00	Conflict		
Create Repeating Meeting: Today, 8:00-9:00	Same Day, 8:00-9:00	Conflict		
Create Repeating Meeting: Start Yesterday, Repeats on Today, 8:00-9:00	Same Day, 8:00-9:00	Conflict		
Create Meeting: Today, 7:30-8:30	Same Day, 8:00-9:00	Conflict		
Create Meeting: Today, 8:30-9:30	Same Day, 8:00-9:00	Conflict		
Create Meeting: Today, 8:15-8:45	Same Day, 8:00-9:00	Conflict		
Create Meeting: Today, 7:30-9:30	Same Day, 8:00-9:00	Conflict		
Create Meeting: Today, 10:00-11:00	Same Day, 8:00-9:00	Free		
Create Meeting: Today, no time	Same Day, 8:00-9:00	Free		
Create Meeting: Today, no time	Same Day, no time	Free		

Create Meeting: Today, any time	Same Day, no time	Free		
Use AAM to Create a Meeting Invite, send	Send the same invite again	Display as a meeting update		
Create Meeting: in a different timezone such that the adjusted time overlaps with 8:00-9:00	Same Day, 8:00-9:00	Conflict		
Create Meeting: in a different timezone such that the NOT ADJUSTED time overlaps with 8:00-9:00	Same Day, 8:00-9:00	Free		

Section 5: Attendee Alert

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Display	Incoming meeting, attendee is free	Your schedule is clear.		
Display	Incoming meeting, attendee is not free	You have a time conflict. (Display conflicting meeting, tested below)		
Display	Incoming meeting, all fields full	Organizer Description (Location) Date Times		
Display	Incoming meeting, no time	Write "No Time"		
Display	Incoming meeting, no description	Omit the description		

Display	Incoming meeting, no location	Omit the location		
Display	Incoming meeting, no organizer	Omit the organizer		
Display	Incoming meeting, attendee is not free: Other Meeting has no time	Write "No Time" (should never happen though)		
Display	Incoming meeting, description really long	Description is truncated after certain amount and catted with a "…"		
Display	Incoming meeting, attendee is not free: Other meeting has really long description	Description is truncated after certain amount and catted with a "…"		
Display	Incoming meeting, attendee is not free: Other Meeting has a time	Write the times		
Display	Incoming meeting, attendee is not free: Other Meeting has no description	Where the description goes, write the organizer/contact		
Display	Incoming Cancel Message	This meeting has been cancelled: (meeting description as above)		
Display	Incoming Confirm Message	This meeting has been confirmed: (meeting description as above), the string " - Confirmed" should be catted on the end of the description		
Display	Incoming Confirm Message - for something that has already been confirmed	Only one " - Confirmed" should be catted on the end of the description		
Display	Incoming Confirm/Cancel Message - for something that has been deleted	Disregard message, disable alert, go to last Agendus view		
Setup	Invite from someone already in your contact database	Update name of the organizer to what you have stored		

Setup	Invite from someone not in your contact database	Add the organizer to the database (name and phone #) only if you accept the invitation		
Handle Event	Invite (free): Accept Button	Generate and send accept message, add meeting to schedule, bring up edit dialogue for that meeting		
Handle Event	Invite (free): Decline Button	Generate and send decline message, go to last view		
Handle Event	Invite (not free): Reschedule Button	Generate and send accept message, add meeting to schedule, bring up edit dialogue for the conflicting meeting		
Handle Event	Invite (not free): Decline Button	Generate and send decline message, go to last view		
Handle Event	Confirm: OK button	Go to last view		
Handle Event	Confirm: Go To button	Go to the meeting that has been confirmed		
Handle Event	Cancel: Set as Cancelled Button	Set the meeting as cancelled, go to that meeting's edit dialogue		
Handle Event	Cancel: Delete Button	Delete the meeting, go to last view		

Section 6: Adding to Schedule

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Adding a Meeting	Normal Meeting	Meeting added to calendar		
Adding a Meeting	Meeting already exists (duplicate or update message)	Meeting updated if necessary		

Check Defaults	Category	Meeting Defaults Persist		
Check Defaults	Icon (if selected)	Meeting Defaults Persist		
Check Defaults	Alarm	Meeting Defaults Persist		

Section 7: Manual Changing of Attendee Status

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**
none	Open New Meeting, add attendees	Status is unknown by default		
none	Close and reopen meeting	Status is the same as before		
none	Change attendee status	Icons next to the attendee change to what you selected		
none	Close and reopen meeting	Status is the same, changes have been saved		
none	Tap on attendee	Goes to card view (pre-existing feature)		
none	Return, add more attendees	Former statuses should be preserved		
none	Close and reopen meeting	Status is the same, changes have been saved		

Create a meeting with attendees in a previous version of Agendus (doesn't have storage for the status), install over	Open this meeting	Status is unknown by default		
Create a meeting (with attendees) in latest version of Agendus, install previous version (doesn't have storage for the status)	Open this meeting	Display of attendees should work. Statuses should not cause parse problems in previous versions.		
Add a bunch of attendees to a meeting so that the scroll bar appears on the side	Change the top attendee's status, then scroll down so it is out of view, then scroll up again	This attendee's status should be what you changed it to		
Checking to make sure new status storage doesn't affect other features using the same functions	Open a contact that has a few other contacts linked to him/her	Should still parse/display information about linked contacts as before		

Section 8: Auto-Update Attendee Status

Setup/Description	Input	Expected Output	Output	Pass/Fail
Create a meeting, send an invite	Receive repsonse from someone on your attendee list, positive	Attendee's status is updated to "Attending"		
Create a meeting, send an invite	Receive repsonse from someone on your attendee list, negative	Attendee's status is updated to "Declined"		
Create a meeting, send an invite, delete the meeting	Receive repsonse from someone	Disregard the message and disable the alert, go to last Agendus view		
Create a meeting, send an invite	Receive response from someone not on your list of attendees, but in your contact database	Disregard the message and disable the alert, go to last Agendus view		
Create a meeting, send an invite	Receive response from someone not on your list of attendees or in your contact database	Disregard the message and disable the alert, go to last Agendus view		

Section 9: Organizer Alert

Setup/Description	Input	Expected Output	Output	Pass/Fail

Display	Positive Response	"Name is attending this meeting:"		
Display	Negative Response	"Name is NOT attending this meeting:"		
Display	Meeting Display	(see Attendee Alert, same code so only test it once)		
Handle Event	Ok Button	Go to most recent view		
Handle Event	Go To Button	Go to meeting edit view with attendee tab showing		

Section 10: Selecting Which Attendees to Send To

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Create a Meeting with four attendees	None.	All attendees should be deselected (dot).		
Create a Meeting with four attendees	Press the "Select All" Button (check mark)	All attendees should be selected (check mark)		
Create a Meeting with four attendees	Press the "Deselect All" Button (dot)	All attendees should be deselected (dot).		
Create a Meeting with four attendees	Click on one of the dots next to an attendee	This attendee should now be selected.		
Create a Meeting with eight attendees	Click on one of the dots next to an attendee, then scroll the table so that attendee is not shown, then scroll back	The correct attendee should still be selected.		

Create a Meeting with four attendees	All attendees should be selected (check mark), Send an invitation	The invitation should be sent to all of the attendees with valid mobile phone numbers.		
Create a Meeting with four attendees	All attendees should be deselected (dot), send and invitation	A warning message should appear alerting the user that no attendees have been selected.		
Create a Meeting with four attendees	Check off some attendees, while leaving others delselected, then send an invitation	The invitation should be sent to only the selected attendees with valid mobile phone numbers.		

Section 11: Quick Deleting an Attendee

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Create a Meeting with four attendees	In the status drop down, select remove attendee.	This attendee should be removed from the list.		
Create a Meeting with four attendees	In the status drop down, select remove attendee. Send an invite	That particular attendee should not receive an invitation		
Create a Meeting with four attendees	In the status drop down, select remove attendee. Exit and return to the meeting.	This attendee should still be removed from the list.		

Section 12: Backwards Compatibility and Existence Testing

Setup/Description	**Input**	**Expected Output**	**Output**	**Pass/Fail**
AAM: not installed AGP: latest version AGM: latest version	Try to do as many AAM functions as possible	No functionality available		
AAM: installed AGP: not installed AGM: not installed	Try to do as many AAM functions as possible	Should not affect anything, it should be as if nothing has been installed		
AAM: installed AGP: latest version AGM: not installed	Try to do as many AAM functions as possible	Only AAM feature available: Manual Changing of Attendee's Status		
AAM: installed AGP: not installed AGM: latest version	Try to do as many AAM functions as possible	No functionality available		
AAM: installed AGP: older version AGM: older version	Try to do as many AAM functions as possible	No functionality available		

AAM: installed AGP: older version AGM: latest version	Try to do as many AAM functions as possible	No functionality available		
AAM: installed AGP: latest version AGM: older version	Try to do as many AAM functions as possible	No functionality available		
AAM: installed AGP: full mode AGM: limited mode	Try to send an invitation.	Agendus Mail should pop up and tell you that it is in limited mode.		
AAM: installed AGP: limited mode AGM: full mode	Send a text messag to this system, and click accept (to add an appt to your schedule)	Agendus Palm should pop up and tell you that it is in limited mode.		

Overall (not an automatic field)

Passed:

Failed:

Not Implemented:

Relevant Failed:

Features of the Palm Operating System

The Palm Operating System is an event based operating system. This means that the structure of each program basically consists of an event loop.

```
static void AppEventLoop(void)
{
      UInt16 error;
      EventType event;

      do
      {
            /* change timeout if you need periodic nilEvents */
            EvtGetEvent(&event, evtWaitForever);

            if (! SysHandleEvent(&event))
                  if (! MenuHandleEvent(0, &event, &error))
                        if (! AppHandleEvent(&event))
                              FrmDispatchEvent(&event);

      } while (event.eType != appStopEvent);
}
```

Each view, or form, has its own event handler. These event handlers are essentially giant switch statements for each control (button, textbox, scrollbar, etc) depicting what the behavior of each should be. If the event handler has fully taken care of the event, then it must return true. However, if it has not (such as in the case of directional pad movement), it returns false and gets handled by the basic form controls.

Another feature of Palm is that there is no multitasking, meaning one application runs at a time. This makes communication between applications a little messier, as there is no application stack. This will be discussed further in a later section.

In addition to allowing only one program to run at a time, Palm OS allows only one application to designate itself as the handler for SMS messages. This was a particular trouble for our project because we needed to intercept particular messages and let others be handled normally. It was either hack into the native messaging application (impossible), or create our own SMS application (very difficult and tedious). Luckily, iambic's program Agendus Mail also has SMS capability, and we were able to put our hook inside that program without adding any extra work.

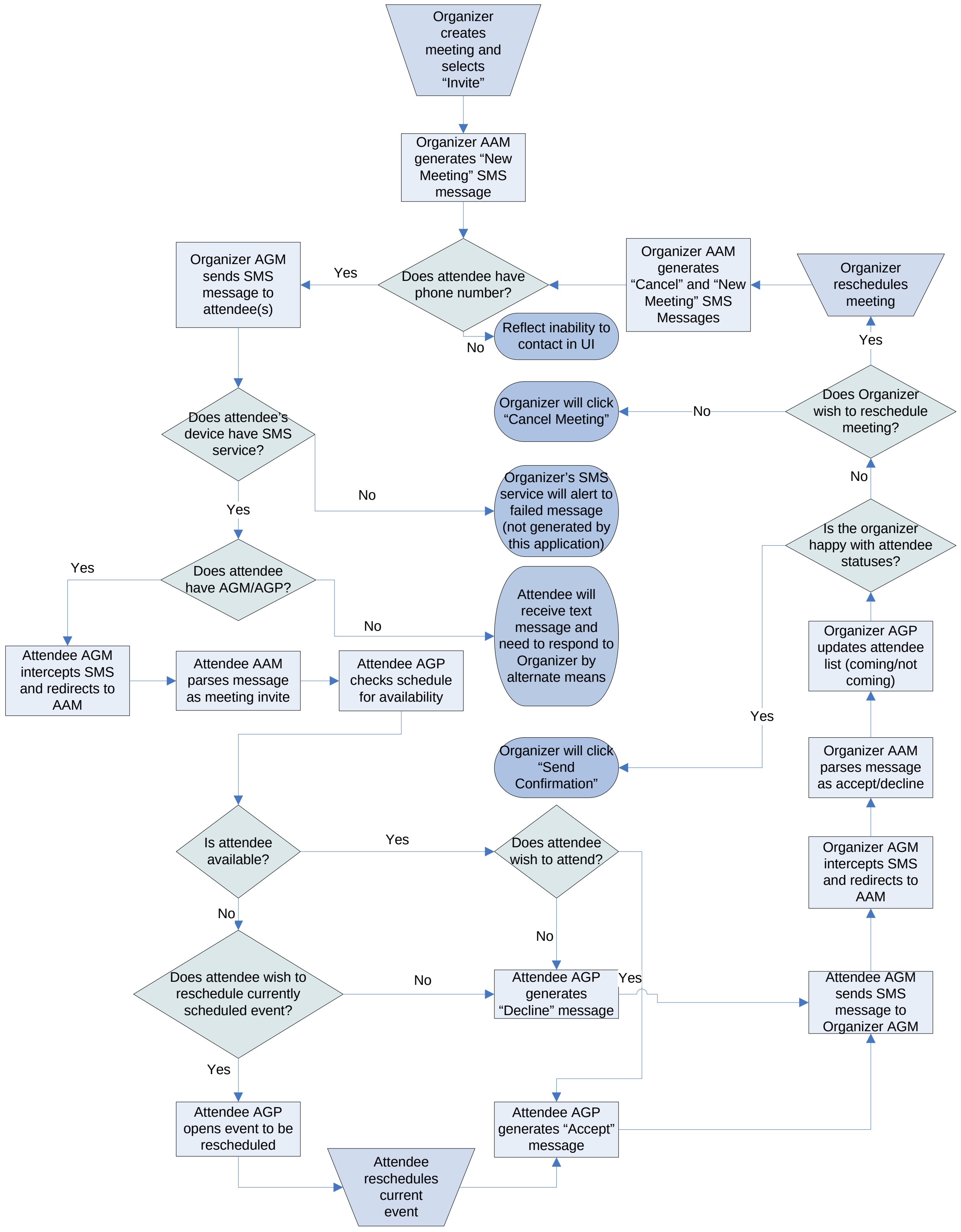
Diagrams

The following diagrams describe the greater overall architecture of the project, as well as a more detailed look at the system level.

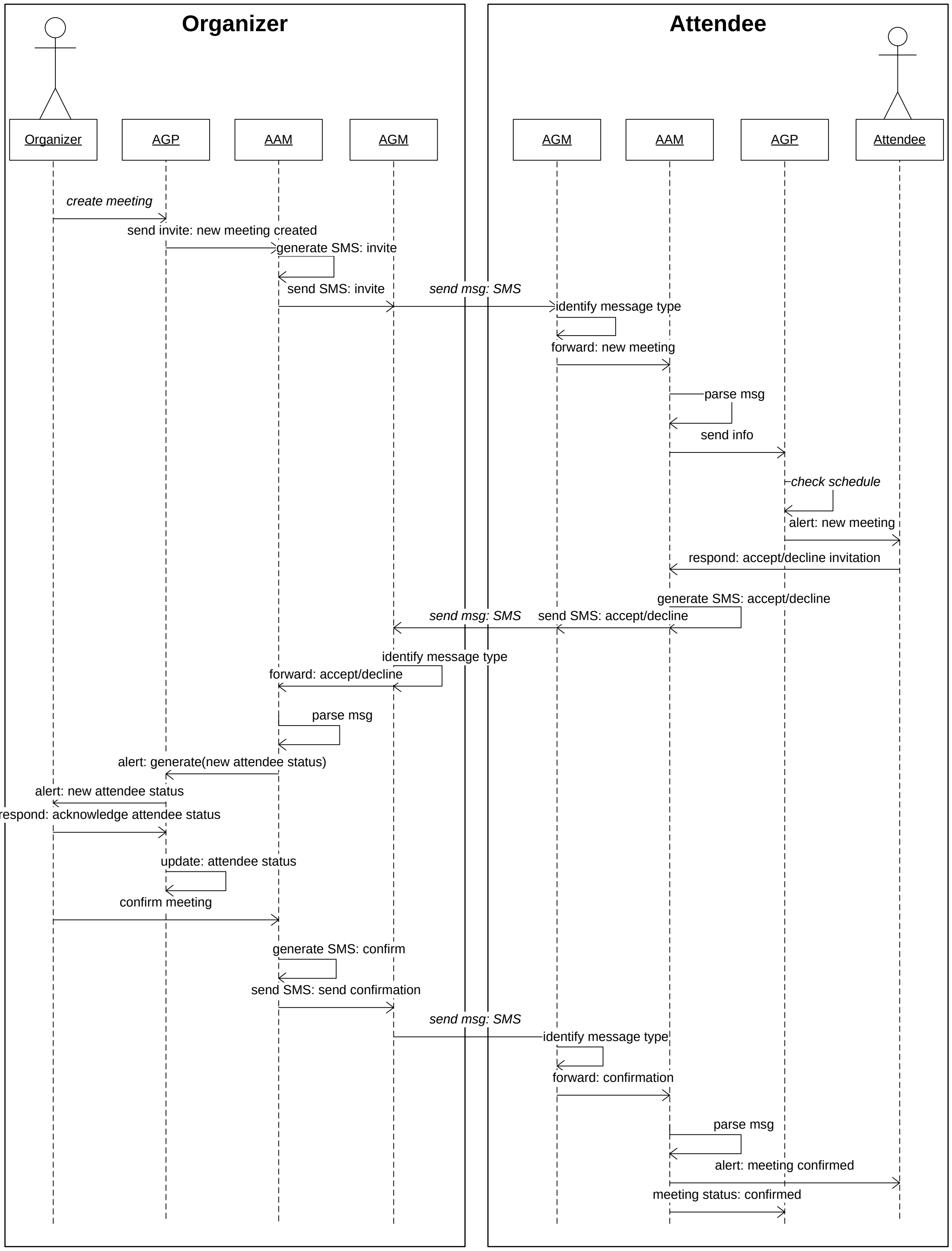
The Flowchart on the following page gives the reader a look at how the overall system operates, with a mapping of all the junctures and decisions that both the Organizer and the Attendee will face when utilizing this project.

Directly after the Flowchart is the System Sequence Diagram. This diagram shows the project’s processes on a more technical level, and makes clearer exactly what functionality is provided by this project. Instead of explaining how the user would interact with the project, this diagram explains how the varying layers of the pre-existing program and the addition this project brings to it would interact.

Flowchart


Organizer creates meeting and selects "Invite"
Organizer AAM generates "New Meeting" SMS message
Does attendee have phone number?
Yes
No
Organizer AGM sends SMS message to attendee(s)
Reflect inability to contact in UI
Organizer AAM generates "Cancel" and "New Meeting" SMS Messages
Organizer reschedules meeting
Yes
Does Organizer wish to reschedule meeting?
No
Organizer will click "Cancel Meeting"
No
Does attendee's device have SMS service?
No
Yes
Organizer's SMS service will alert to failed message (not generated by this application)
Is the organizer happy with attendee statuses?
Does attendee have AGM/AGP?
Yes
No
Attendee will receive text message and need to respond to Organizer by alternate means
Organizer AGP updates attendee list (coming/not coming)
Attendee AGM intercepts SMS and redirects to AAM
Attendee AAM parses message as meeting invite
Attendee AGP checks schedule for availability
Yes
Organizer will click "Send Confirmation"
Organizer AAM parses message as accept/decline
Is attendee available?
Yes
Does attendee wish to attend?
Organizer AGM intercepts SMS and redirects to AAM
No
No
Does attendee wish to reschedule currently scheduled event?
No
Attendee AGP generates "Decline" message
Yes
Attendee AGM sends SMS message to Organizer AGM
Yes
Attendee AGP opens event to be rescheduled
Attendee AGP generates "Accept" message
Attendee reschedules current event

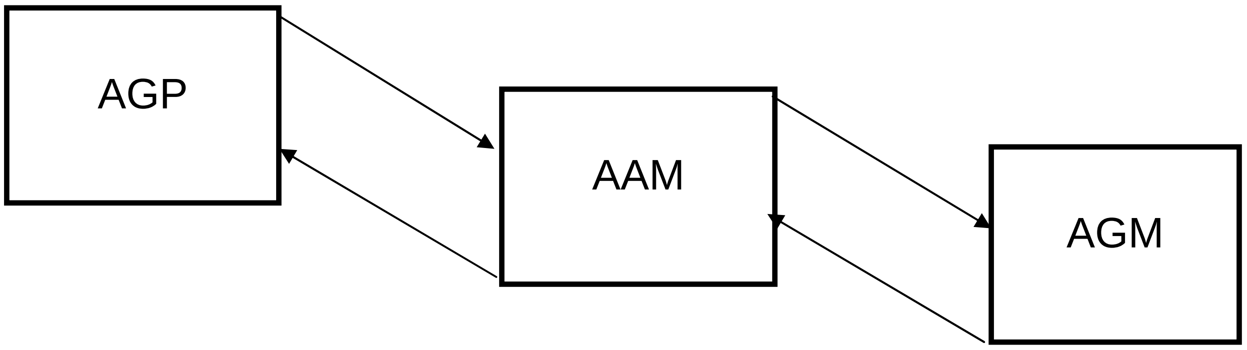
System Sequence Diagram



Technical Challenges

Inter-application Communication

Our application has a whole consists of three applications working together, where no two applications can be running at the same time. AAM acts as a middleman between AGP and AGM. When switching from one application to another, a launch command along with a specific parameter structure is sent to the other application.



The first thing to do when switching between applications is to find out if the other program actually exists on this device. To do this, we search for the program (called a database on this platform) based on its creator ID, which is given to the developer by Palm itself. What made things a little more interesting was that AGP and AAM shared the same creator ID, so when we tried to switch to one of them, we had to add extra checks to ensure that we found the correct one.

Next, the version must be checked. If an older version of any program is installed on the device, then the necessary launch code definitions will not exist in the program, and the application will not know what to do upon receiving them.

After this, the license must be checked. In order for the whole package to work together, a special license must exist inside iambic's license database on this particular device. Not only that, but the license must not be expired. When a user downloads the program for free, he/she is given a short trial period. After that, the program is put into "limited mode," where things like creating new meetings and records are not allowed. If the program is in this limited mode, then our functions would fail.

The next step is to set the memory allocated to the parameter struct to be owned by the operating system. If this is not done, the memory will be released upon exit of the program, leading to strange data on the receiving end. One problem we ran into here was that we had character pointers in our parameter structs. Of course, when we gave the memory to the operating systems, we were only giving the pointers themselves, not what they pointed to. This was giving us very strange and inconsistent results until we found out what the problem was. Once we changed our pointers to character arrays of specific length, these problems disappeared.

Last but not least, simply call the function that switches applications, passing it the specific launch command, the struct, and the necessary information to identify the other program. The operating system will take care of making sure the launch code and struct go where they need to go.

Here is an example of application-switching code:

```
err = DmGetNextDatabaseByTypeCreator(true, &searchState, sysFileTApplication,
                                          iambicMailAppFileCreator, true,
                                          &cardNo, &dbID);

if (!err)
{
      // Find out if the version of AGM is compatible
      err = DmDatabaseInfo(cardNo, dbID, NULL, NULL, &version, NULL,
                              NULL, NULL, NULL, NULL, NULL, NULL, NULL);

      if (!err)
      {
            // Version 5 contains AAM launch code information
            if (version >= 5)
            {
                  // Check product license
                  if (!isLicenseExpired(k_license_mail))
                  {
                        // Everything is OK, send to AGM!
                        if (MemPtrSetOwner(cmdPBP, 0) == errNone)
                        {
                              //switch to AGM to send SMS
                              SysUIAppSwitch(cardNo, dbID,
                                                  sysAppLaunchCmd_AAM_AddSMS,
                                                  cmdPBP);
                        }
                        else
                              MemPtrFree (cmdPBP);
                  }
                  else
                  {
                        // AGM is in limited mode.
                        // Return to AGP, with a message that you need AGM
                        goto GoBack;
                  }
            }
            else
            {
                  // AGM does not have the latest version.
                  // Return to AGP, with a message that you need AGM
                  goto GoBack;

            }
      }
}
```

```
else
{
      // AGM is not installed, but we know AGP is, because that called us.
      // Return to AGP, maybe with a message that you need AGM
GoBack:
      if (DmGetNextDatabaseByTypeCreator(true, &searchState,
                                          sysFileTApplication,
                                          AgendusAppType, true,
                                          &cardNo, &dbID)  != dmErrCantFind)
      {
            SysUIAppSwitch(cardNo, dbID, sysAppLaunchCmd_AAM_NeedAGM, NULL);

      }
}
```

Meeting UID Generation

Within each device, every meeting record is given a unique identification number, or UID. Unfortunately for our purposes, we could not afford to use this number to identify an AAM meeting. A meeting UID was guaranteed to be unique on one particular device only. Since we are spreading the same meeting across multiple devices, we needed to create our own UID.

At first, we created a nine digit UID based off of the date and time of the meeting. We quickly realized that a user may receive or send multiple meetings for the same date and time, so we changed the UID to include a randomly generated element as well.

Repeating Meetings

What happens when an organizer wants to send an invitation to a weekly repeating meeting? Repeating meetings are stored as only one meeting record with details set to show how it repeats. Obviously, the responses from the attendees would be significant only for the current instance of this meeting. So, in order to prevent confusion on both the user's and the application's part, a new meeting is created with the same exact details minus the repeating. In other words, an exact duplicate meeting is created of only that single instance. Then, the original repeating meeting is set to have an exception on that day, meaning that it will not have a repetition where the new single meeting takes place.

Storage of Attendee Status

Both Agendus and the native calendar application already had the ability to add attendees to meetings, but neither of them had ways of tracking an attendee's status. It was very confusing at first because both the native fields *and* the fields within the note were being filled with attendees. Naturally, we decided to modify both. Unfortunately, we did not realize at the time that we weren't allowed to mess with the native database structures.

```
typedef struct
{
      ApptDateTimeType  *when;
      AlarmInfoType           *alarm;
      RepeatInfoType          *repeat;
      ExceptionsListType      *exceptions;

      char *description;
      char *note;
      char *location;

      ApptTimeZoneType timeZone;
      ApptMeetingInfo meetingInfo;
      UInt16 numBlobs;
      BlobType blobs[apptMaxBlobs];

} ApptDBRecordType;
```

Fortunately, Agendus has a much more flexible method of storing extraneous information. It creates a string "header" for the note field of the meeting. Note fields have an exceptionally long length allotment. This allows us to store almost boundless information. For the attendee list, it is stored like this:

```
Note: "…CONLIST:<Contact1>(0), <Contact2>(2), Contact3>(0)\n#AN\nNormal Note
Text!"
```

Notice that the numbers in parentheses next to each contact are integers corresponding to specific statuses.

This information was only written once the user gave the final OK to commit changes to the meeting. The process of temporarily storing and manipulating the data was another headache altogether.

Parsing

A good parser is absolutely necessary for an application to be both strong and stable. The parser needed to be able to interpret data from a string into specific data structures. It also needed to recognize invalid data. For instance, if the meeting is set for 45:00, then the message is rejected as being invalid. Another essential ingredient is security. If not properly formatted, a malicious or unknowing user may send data that is not in exactly the form that the parse expects. Because we are dealing with strings and character pointers, this will often cause the system to crash. We tried our best to make our parser as reliable, stable, and secure as possible.

Interfacing and working with a very large and dynamic codebase

Even as we were developing our product, a new version of Agendus Palm was in development as well. Because of this, the codebase we were working with was both large and dynamic. Sometimes, problems with the code would stem from us and affect normal development, and sometimes it was the reverse. We had to learn to work together to solve some problems and ensure a good product.

Localization

Because our product is being released in eight different languages, we had to make sure that every part of the user interface was completely localizable. AGP and AGM already had the necessary mechanisms to implement this, so we kept all UI in these two programs alone. AAM became a merely a messenger between the two.

Risks

This section delineates certain problems which our project may encounter and have a fair to high probability of impeding the progress of our project.

Critical

Risk: *If our relationship with iambic is somehow damaged or retracted, it would prevent us from accessing their code. Our entire project is defined on the assumption that this code will be available to us, and so this problem would cause our project to fail utterly.*
Resolution: We worked diligently, making certain to produce code that was not only error-free, but was also consistent with the internal coding style of iambic.

Serious

Risk: *If it turns out that the scope of our project is unrealistic, then we may not be able to satisfy all of the success requirements listed in the document, and our project might fail.*
Resolution: We managed to avoid this problem entirely, as can be seen in the section entitled "Success Requirements." Not only were our core requirements satisfied, but we managed a good portion of the requirements deemed "For Future Consideration" (which, at the time of dividing which requirements were mandatory and which were not, it was anticipated that this section would never be reached).

Risk: *One of us is currently suffering from an injury that requires surgery in December; if complications arise and she is unable to work, the completion of the project may be seriously impacted.*
Resolution: Surgery went well, and she was diligent with both her work and recovery. Risk was not a factor.

Risk: *We went a very long period without looking at the risk document, which allowed us to become lax in our productivity.*
Resolution: Print and tape the list of risks to our machines, so we could see them as we worked, alongside with the list of requirements and deadlines.

Risk: *Unforeseen technological failures, such as email unable to be sent and received, could seriously hamper our ability to communicate and stall the project and perhaps put it at a critical status.*
Resolution: Several different methods of communication, such as various email addresses and phone numbers, were exchanged in order to ensure a potential for constant communication from the side of the programmers to the side of the advisor. Regardless of this, communication was still an issue.

Risk: *Unforseen problems can occur due to interfacing with a large, pre-existing codebase.*
Resolution: A willingness to ask questions of those who know the codebase very intimately, as well as the ability to revert to a previous version of said code if considerable changes were made that did not coincide with how the codebase should work alleviated this problem.

Concerning

Risk: *If we have issues with obtaining and installing tools, iambic has offered to allow us the use of the environments they have already set up in the office. It would mean, however, that every time we want to work on the project, we would have to go to iambic's office, which may have certain schedule constraints associated with it. This may require us to work harder to guarantee the success of the project.*
Resolution: This turned out to be less of a risk and more of a boon to the project's success. By going to iambic every day to work, we were ensured a work environment with no distractions, as well as ready access to those more knowledgeable about iambic's codebase, such as Adriano.

Risk: *Because one of us has never worked with the Palm Operating System before, there may be a significant learning curve to overcome in order to get her up to speed. This will cause extra work for both team members. However, this is in no way an excuse for the failure of the project.*
Resolution: Said team member worked very hard to understand the codebase, and asked questions constantly, to ensure the success of the project.

What We Learned (from a Software Engineering Perspective)

Do Not Consider Mediocrity an Option

Because we knew from the beginning that our product was going to be released to the public, we never had the option to program anything less than near perfection. When tens of thousands of users get a hold of the product, what seems like a small problem to the developer is multiplied exponentially over the user-base. In addition, users like to complain about problems they encounter, which deters other users from purchasing the product. If you do not allow yourself to program anything that isn't extremely stable, then you will never find yourself in these situations.

Never Back Down From a Challenge

There is always a solution to your problem. Have faith in that, and commit yourself to the task of finding out what it is. Force yourself to think about what would be good for the users, and not what would be easy for the developers.

Keep the Repository Gold

Without this, you can go absolutely nowhere. It is especially important when working on a project with people in different locations or even time zones. We were working with developers in the Ukraine and in New York. If we had done something to break the code and then left for the day, the people in the Ukraine would not be able to work until we came back. For the sake of moving forward *at all*, the repository must always be kept golden.

Testing is Incredibly Important

Testing in small increments is incredibly helpful because you are able to pinpoint problems quickly and efficiently *before* it comes time for the debug phase (integration testing) of the project. Unfortunately, we were not able to use an automated test framework, and so we had to do all the testing by hand at every iteration. On one hand, it was nice because we could perform one test at any time. However, you have to be very careful with that. A fix for one bug may have caused something else to go wrong. After you fix one thing, you should always go back and test everything to ensure the proper function of the application as a whole.

Find a Workplace with no Distractions

For us, this was one of the most important ingredients to our success. Because we were working with iambic, we had to go to the office in order to do our work. There, we had programming environments set up for us. Since we were in an office environment, every time we went to work, we really worked. There was no TV, video games, or conversations to distract us.

Set Aside Weekly Times to Work

This was the other major factor in our achievement. We set aside a specific schedule of 9-10 hours a week where we would work on our project. We never broke this schedule. We solidified it in our heads and made sure to tell our boss when to expect us. This way, if we didn't show up it wouldn't just be our loss, but his as well.