# Wonderland Software Services (WSS) Meeting Scheduler

Version 3.0

May 1, 2022


Created by:

Tayyaba Ali

Bhargvi Handa

# 1. Introduction

## Purpose:

Wonderland Software Services (WSS) wants to design a meeting scheduler that automates and streamlines the process of finding a date, time, and place that meet all the *participants' constraints*. The system-as-is problems are that poor attendance for planned meetings is prominent since meeting dates are chosen poorly. Furthermore, the process of choosing dates can be tedious and often not pleasant for all parties involved, this is especially true with individuals with poor communication or last moment changes in their calendar. The system-to-be would be a scheduler software that is capable of automating the alignment of multiple environmental factors, without much input needed from any party or authorization power. The alternative option would be figuring out the time constraints individually of invited participants through email or accessing their electronic agenda. This is oftentimes not realistic or very time-consuming.

## Overview of Project:

The WSS system, referred to as the *Meeting Scheduling System*, can do the following:

1. Allow the company's *authorized account holder(s)* to register a *user*
2. All users within the company are compiled in the user manager portal (referred to as *systemUsersManager*)
3. A *user*, referred to as an initiator, can create and submit a *request* to the *Scheduling Manager* to hold a *meeting* with other *users* from the company
    a. The meeting request requires the *initiator* to input the following *meetingParameters*: final date/before this date, location (online or building & room), state/province, country, material needed for participants being invited, list of all participants, dictation of *high priority participants*, and additional notes
    b. A *meeting* can only be organized up to 5 weeks from the current date
    c. Once *meetingParameters* are designated, the Scheduling Manager will find the *OptimalMeetingTime*
    d. Based on the *invitedparticipant's* profiles, the *Scheduling Manager* will identify *potentialMeetingConstraints* and then output a list of ideal meeting times
    e. Canceled attendance will trigger an automatic *rescheduling* if it is over 72 hours before the meeting time
4. The meeting scheduler system should have the internal ability to manage several *meeting requests* in parallel and resolve conflicts without *the authorized account holder*'s supervision.
5. Once a new approved/confirmed request enters a user's request manager then it will show up on the user's *calendarView*
6. *PersonalCalenderEvents* added by the user (if they choose to not automatically sync their personal calendar) can also be viewed in the *calendarView*

## World Machine Diagram

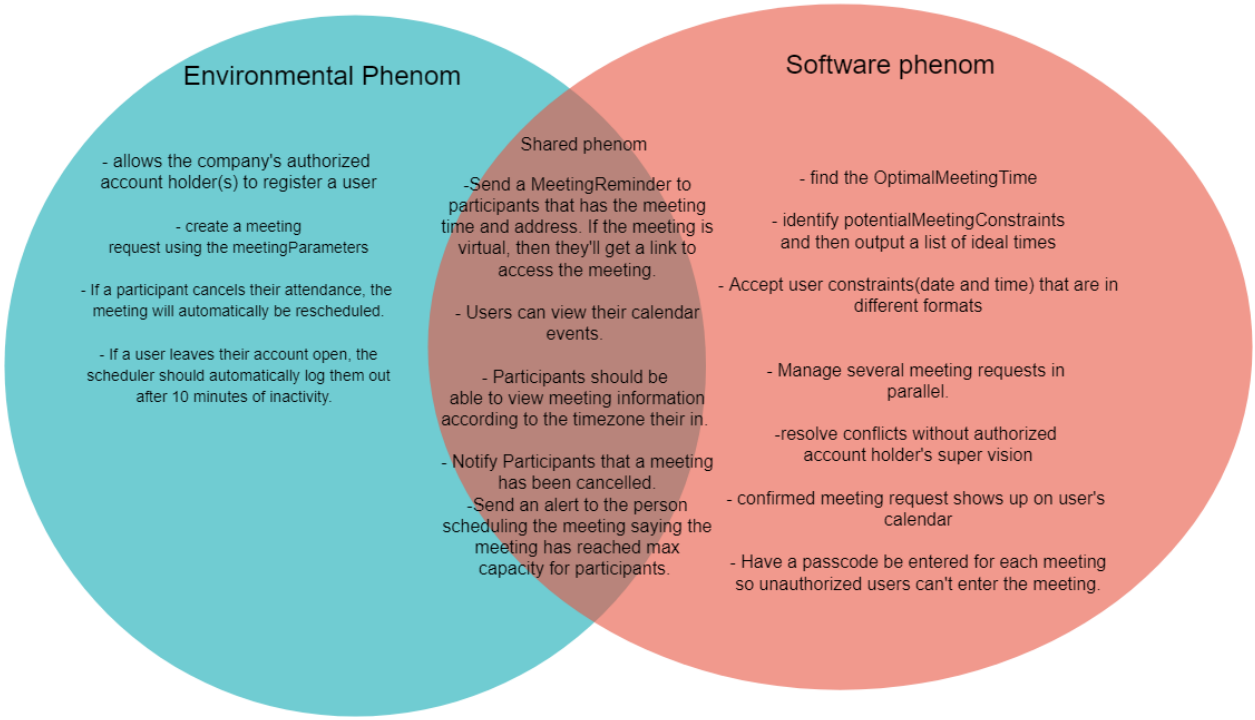



Figure 1.1 provides a high-level conceptual view of the application as a World - Machine Diagram.

**Software Requirements:** The meeting scheduler must be able to find the *OptimalMeetingTime*, identify *potentialMeetingConstraints* and then output a list of ideal times. In order to handle a *participant's* cancellation, the software must be able to automatically reschedule the *meeting*. The meeting Scheduler must be able to manage several *meeting* requests in parallel and resolve conflicts without the *authorized account holder's* supervision. Once the *meeting* is confirmed, the *meeting request* shows up on the *user's* calendar.
**Domain Properties:** *User(s)* can't attend multiple meetings at once.
**Assumptions:** *User(s)* consistently update their *personal constraints* and location preferences and log all other meetings in the scheduler as well, which is internally formatted to time zone changes. Only desired participants are on the meeting *request*.

# 2. Goals and Risk Model

## Goals Model

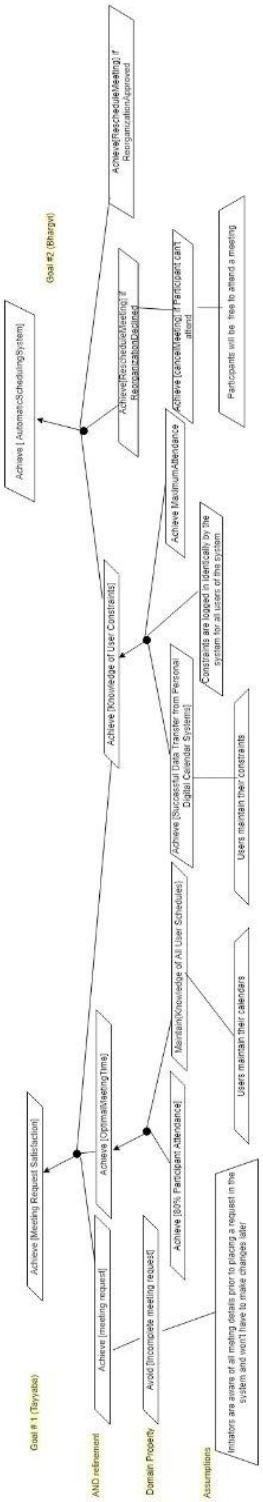
In the goal model below (figure 1.2), there are 2 goals that should be achieved. For Goal 1, The initial problem statement was: "The purpose of the meeting scheduler system
is to support the organization of meetings- that is, to determine, for each meeting *request*, meeting date and location so that most of the intended *participants* will effectively participate." This shows the importance of the meeting *request* going through and its requirements being met by the system and *participants*.

For goal 2, the purpose of the meeting scheduler is to automatically create meetings based on the availability of *participants*. With this top-level goal, the system will be able to accept the *constraints* of *participants* so it can decide what the optimal time, meeting, and location will be for the meeting. Once this goal is achieved, the process of creating meetings will be smoother and not as bothersome to people who want to schedule a meeting.

In order to achieve *MaximumAttendance* in goal 2, Successful Data from Personal Digital Calendar Systems must be achieved. Once there is a successful data transfer from Personal Digital Calendar systems, the scheduler will be able to achieve the *MaximumAttendance* goal. Now, the scheduler will be able to automatically create meetings based on the availability of the *participants*. This overlap is there because, without a successful data transfer, the scheduler will be unable to create the proper constraints which would prevent *MaximumAttendance* from being achieved.

Goal Model (Figure 1.2)

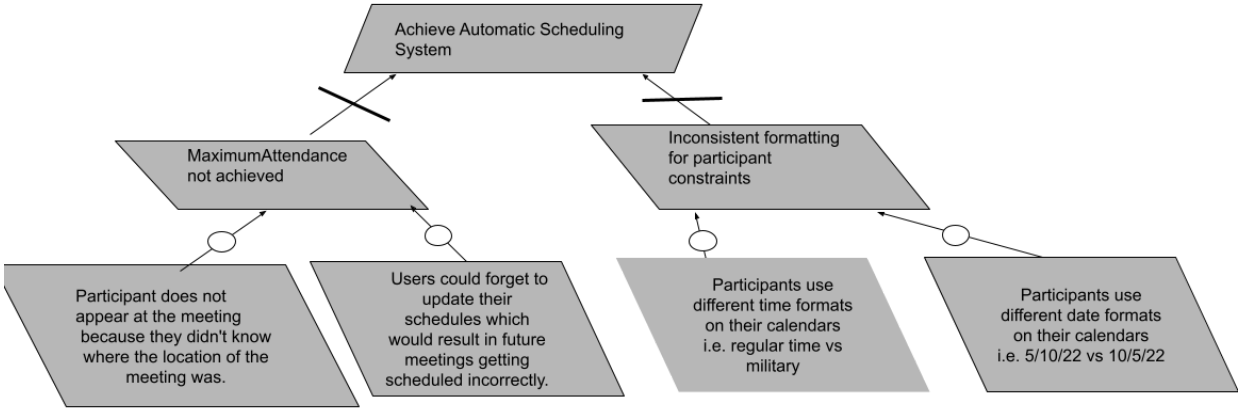
Here is the link to view the full goal model:
https://drive.google.com/file/d/1XRxnbqwldm5hSwCeuRoBfrKB7HNvq0OT/view?usp=sharing



## Risk Model

The first two risks listed in the risk summary chart are shown in Risk model 1. When scheduling meetings, there is a risk that the *meeting* does not have all of its *participants* that were scheduled to attend. A *participant* could not attend a meeting because they did not know where the location of the *meeting* was. Another way that a *meeting* could not reach *MaximumAttendance* is because *users* could forget to update their schedules which would result in future *meetings* getting scheduled incorrectly. Another risk that could happen in scheduling *meetings* is that a *participant's constraints* could be formatted in a way that is inconsistent with the scheduler such as different time and date formats. In order to prevent this risk, a *notification* will appear on a *participant's* email or phone telling them they have an upcoming *meeting* which is sent at the start of the week. This *notification* will have the meeting time, date, location, and address if the meeting is outside in a different space. If the *meeting* is virtual, a *participant* will get a link to access the *meeting*.

Risk Model 1 (Figure 1.3):

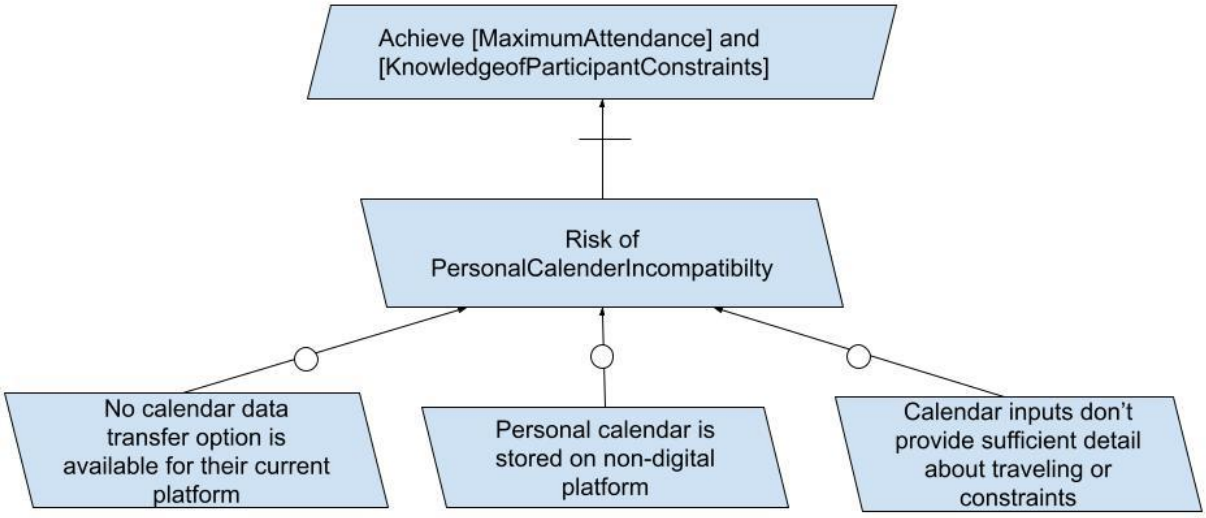



First Chosen Risk: *Maximum Attendance* not achieved and Achieve[knowledge of *participant constraints*]

**Detailed Name**: *MaximumAttendance* is not achieved in a meeting
**Priority**: Medium
**Category**: System-to user
**Likelihood**: 5
**Definition Explanation of the risk**: When scheduling *meetings*, there is a risk that the *meeting* does not have all of its *participants* that were scheduled to attend. A *participant* could not attend a *meeting* because they did not know where the location of the *meeting* was. Another way that a *meeting* could not reach *MaximumAttendance* is that *users* could forget to update their

schedules which would result in future *meetings* getting scheduled incorrectly. If the *meeting* gets scheduled incorrectly then a *participant* could not attend a *meeting* because their calendar listed that they were busy despite being at the moment the *meeting* was scheduled for.
**Detailed Name**: Participant constraints incompatible with current scheduler system
**Priority**: Medium
**Category**: System-to user
**Likelihood**: 5
**Definition Explanation of the risk**: When scheduling *meetings* with different clients, the clients could come from a place where they format their date and time differently than others. If the scheduler only supports only one time and date format, then it is difficult to schedule *meetings* properly with international clients which is why it is important that the scheduler accepts time and date constraints in different formats.

Risk Model 2 (Figure 1.4):




## Second Chosen Risk: Personal Calendar Incompatibility

**Detailed Name:** Personal calendar host system is incompatible which prevents accurate automatic data transfer
**Priority:** High
**Category:** System-to-system communication
**Likelihood:** Out of a scale of 10, it is a 6 in likelihood (6/10)
**Definition Explanation of the Risk:** The meeting scheduler system can't automatically assume that it will be compatible with every calendar system in the market, as it will require the ability to

customize established data within the *user's* personal calendar system. Some calendar systems don't have a share option, which prevents the scheduler from uploading the information automatically. There is also no personal availability input option in the meeting scheduling system.

**Suggestions for Improvements to the System:** In order to prevent two out of the three risk subcategories, it is important to support the transition of users to prioritize the sole usage of this meeting scheduler. As the primary system used within a company, the scheduling system will be entirely compatible with Google calendar and Microsoft Outlook, the two most common corporate calendar platforms. Since that doesn't cover all the different digital and non-digital platforms that users may utilize, it is important that the option of creating individual events is possible. This personal calendar option will allow users to transition to using our system as their primary calendar platform or at the bare minimum allow them to block out times they may be unavailable or busy. The last subcategory regarding insufficient details is difficult to address purely based on calendar details. To combat this, a notification will be sent out every two weeks regarding traveling and how to properly add those into the scheduling system as a constraint. While this won't prevent all likelihood, it will significantly decrease the chances of occurrence.

# 3. Atomic Requirements

To aid understanding of the functional requirements, the team has divided all requirements into two categories, general functional requirements, and process-specific scheduling requirements. General functional requirements are about the software system and its interaction with other computer systems or the user in a situation outside of setting up a meeting. Process-specific scheduling requirements, as the name states, focus on creating all requirements that require the interaction of users with the system in order to set up a meeting. The following table shows all requirements collectively along with their associated grouping category.

| **Solution Technique Name** | **Grouping Category** | **Description** | **Priority Level** |
|---|---|---|---|
| FR 1.1- US | General | User Settings | High |
| FR 1.2- PCT | General | Personal Calendar Transfer | Medium |
| FR 1.3- PDV | General | Participant Data Viewing | High |
| FR 1.4- UP | General | User Personalization | Low |
| FR 1.5 - RET | General | Required Effort and Troubleshooting | Low |
| FR 1.6 - CVM | General | Cumulative View of all Meetings | High |
| FR 1.7 - AN | General | Agenda Notification | Medium |

| | | | |
|---|---|---|---|
| FR 1.8 - MRS | Process- Specific | Meeting Request Submission | High |
| FR 1.9- RD | Process- Specific | Reorganization Declined | Medium |
| FR 1.10- RA | Process- Specific | Reorganization Approved | Medium |
| FR 1.11- DB | Process- Specific | Double Bookings | High |

- FR 1.1 - User Settings (US)
    - **Specification:** *Users' personal constraints* can be set in the user's *settings*, which are composed of general information (timezone, "out of office hours", and vacation dates) and notification information (such as a phone number, email, etc). This will aid the *Scheduling Manager* in helping initiators find *OptimalMeetingTime* in meeting requests.
    - **Fit criterion:** A test *notification* will be sent to each user when they register their personal contact information within the system's settings.
    - **Category:** Environment Requirement
    - **Assumptions:** All participants update their information consistently or as applicable.
    - **Domain Properties:** The system stores all personal information and notifies the user if the information is incorrect or incomplete.
    - **Priority Level:** High
- FR 1.2 - Personal Calendar Transfer (PCT):
    - **Specification:** The *user* can choose to automatically transfer their *personal calendars* to sync into their user profile or they can manually add personal events manually. All of this will aggregated data can be visible in the user's *calendarVIew*.
    - **Fit criterion:** If the calendar events were able to sync correctly, the scheduler will send a *notification* saying "all events have been added to the scheduler".
    - **Category:** System Requirement
    - **Assumptions:** Users will update their personal calendars regularly and not choose to use a non-compatible format.
    - **Domain Properties:** The scheduler supports data integration from calendars like Microsoft Outlook Calendar and Google Calendar.
    - **Priority Level:** Medium
- FR 1.3 - Participant Data Viewing (PDV):
    - **Specification:** All *data* carried from a user's *personal calendar* can only be viewed by the user at hand. The system, when reading data to find *optimalMeetingTime* for meeting *requests* will process the data in binary (available or busy)
    - **Fit criterion:** The *user profile* is the access point for data from other users and all data is in binary to protect privacy.
    - **Category:** System Requirement
    - **Assumptions:** The scheduler is able to properly convert personal calendar data into binary.

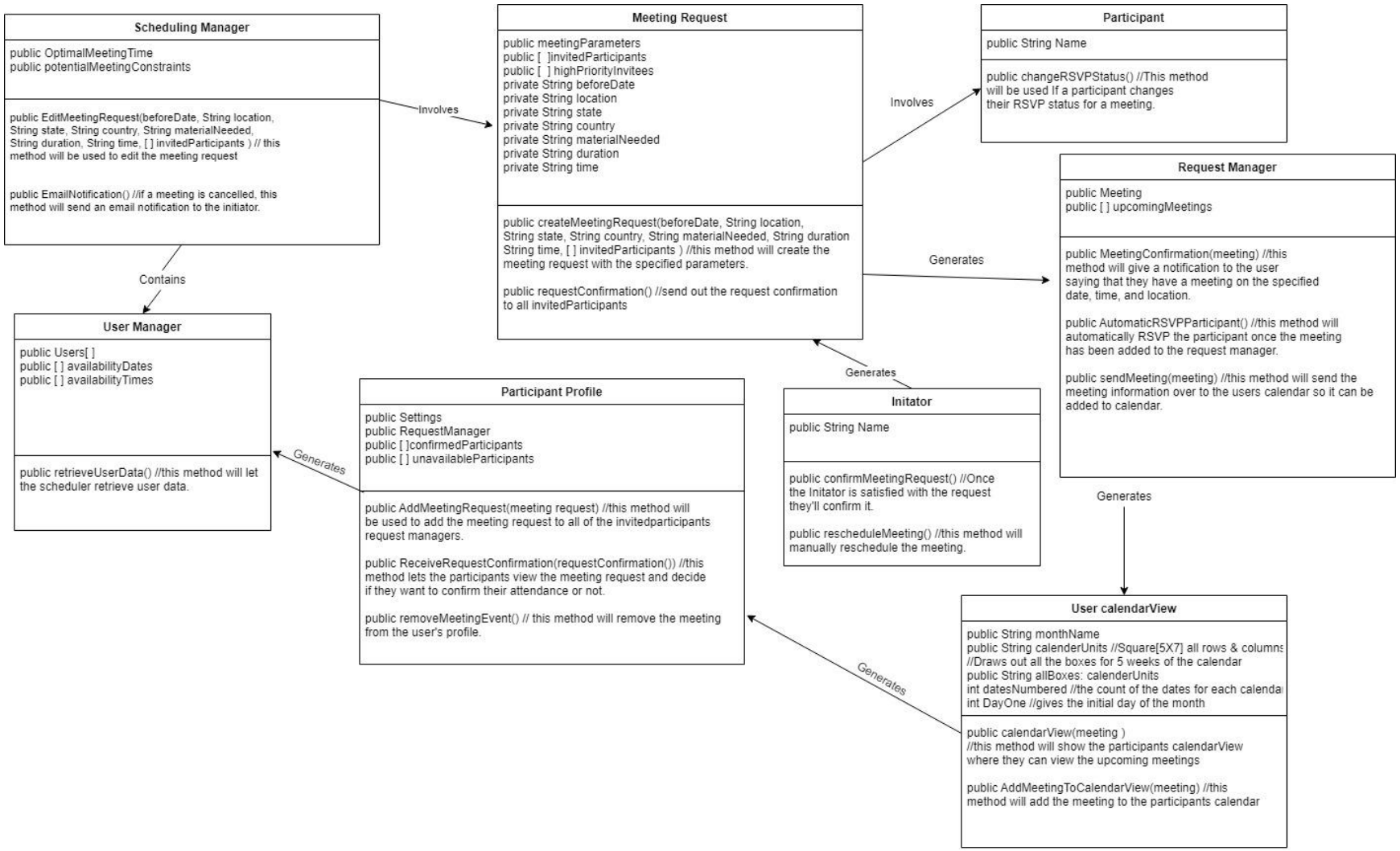
    - **Domain Properties:** Other *users* in the meeting scheduler can not see the specific events that are listed in another *user's* calendar in the system.
    - **Priority Level:** High
- FR 1.4 - User Personalization (UP):
    - **Specification**: The *meeting* information, when being observed by the *participants*, must be automatically changed to their personal *timezone* and language preferences.
    - **Fit criterion:** Create a test *meeting* and have a *participant* see if the *meeting* time and language have changed based on their preference.
    - **Category:** Software Requirement
    - **Assumptions:** The software supports all existing *time zones* and languages.
    - **Domain Properties:** The software supports all existing *time zones* and languages.
    - **Priority Level:** Low
- FR 1.5 - Required Effort and Troubleshooting (RET):
    - **Specification:** Minimal to no effort is required from *authoritative account holders*.
    - Fit criterion: If the *meeting* is successfully scheduled, the *participants* will get a notification saying “meeting schedule is successful”.
    - **Category:** System requirement
    - **Assumptions:** After the *meeting* is successfully scheduled, the *authoritative account holders* should not have to make any further arrangements for the *meeting*.
    - **Domain Properties:** If the *meeting* is successfully scheduled, the *authoritative account holders* should not have to make any further arrangements for the *meeting*.
    - **Priority Level:** Low
- FR 1.6 - Cumulative View of all Meetings (CVM):
    - **Specification:** If a system *user* created a *meeting* in which an individual is listed among the *invitedParticipants*, the *meeting* will be added to the individual’s *Request Manager*. The individual will automatically be RSVP’ed to all meetings that they are available for but the user has the option to cancel a meeting.
    - **Category:** Software Requirement
    - **Assumptions:** The *user* will regularly view the *request manager* and cancel any meetings that they can’t attend.
    - **Domain Properties:** When the *user* is prompted to create a demo *meeting request* when opening their accounts, the meeting will show up in their *request manager*.
    - **Priority Level:** High
- FR 1.7 - Agenda Notifications (AN):
    - **Specification:** The *user* should be notified daily at 5 am, in their local timezone, of their meetings for the current day and the next day (all meetings for the next 48 hours).
    - **Fit criterion:** This is located in *user settings* and required when signing up the user.

    - **Category:** Software Requirement
    - **Assumptions:** The user will update their contact information as needed.
    - **Domain Properties:** A test message is emailed or texted to the user to ensure there are no delivery concerns. Initial test samples will be done using a sample user and their *personalCalender*.
    - **Priority Level:** High
- FR 1.8 - Meeting Request Submission (MRS):
    - **Specification:** Any system *user*, referred to as an *initiator*, can create a meeting *request* with any *users* that exist in the system. Meetings are limited to being up to five weeks away from the date of the meeting *request* creation.
    - **Category:** Environment Requirement
    - **Assumptions:** The *initiator* will only invite *participants* that are required.
    - **Domain Properties:** All *users* will be prompted to create a demo meeting when their accounts are created to aid them in the process.
    - **Priority Level:** High
- FR 1.9 - Reorganization Declined (RD):
    - **Specification:** If the *participant is* unable to attend and declines the *meeting* more than 72-hours before the *meeting* starts, the *meeting* will not automatically get reorganized without the *initiator's* approval.
    - **Fit criterion:** To ensure that *participants* are aware of the cancellation, all associated parties to the *meeting* will be notified within 60 minutes of the *meeting* being rejected/canceled by a *participant* through email or text. The *initiator* will also get the option to reschedule the *meeting* easily through an email option, with all data already previously entered.
    - **Category:** System Requirement
    - **Assumptions:** The *participant* will notify the *initiator* or system they are unable to attend a *meeting* as soon as they find out.
    - **Domain Properties:** If a *participant* can't attend a *meeting*, they must notify the *initiator*.
    - **Priority Level:** Medium
- FR 1.10 - Reorganization Approved (RA):
    - **Specification:** If a *participant is* unable to attend and declines a *meeting* 72 hours in advance of the meeting time, the *meeting* will automatically get reorganized by the system to the next available time without the *initiator's* approval.
    - **Fit criterion:** All associated parties to the *meeting* will be notified within 60 minutes of the *meeting* being rejected/canceled by a *participant* through email or text.
    - **Category:** System Requirements
    - **Assumptions:** The *participant* will notify the *initiator* or system they are unable to attend a *meeting* as soon as they find out.
    - **Domain Properties:** If a *participant* can't attend a *meeting*, they must notify the *initiator*.
    - **Priority Level:** Medium

- FR 1.11 - Double Bookings (DB):
    - **Specification**: No *user* within the system can be double-booked for any *meeting* at the same time.
    - **Fit criterion:** The software will check all the *participant's* calendars to see if they have 2 *meetings* set in a one-time slot. If the software finds two *meetings*, then it will send a notification to the *participant* asking them when they would like to reschedule the second *meeting*.
    - **Category:** Software requirement
    - **Assumptions:** The *participant* is not attending a second *meeting* at the same time as another *meeting*. The participant only has one calendar that is accessible to the service.
    - **Domain Properties:** The second *meeting* is rescheduled to be at a later time.
    - **Priority Level:** High

# 4. Concept / Class Model

This concept model shows the relationships between all of the static components that are a part of the meeting scheduling system. It shows the flow of how the components work together to create meetings and reschedule meetings.

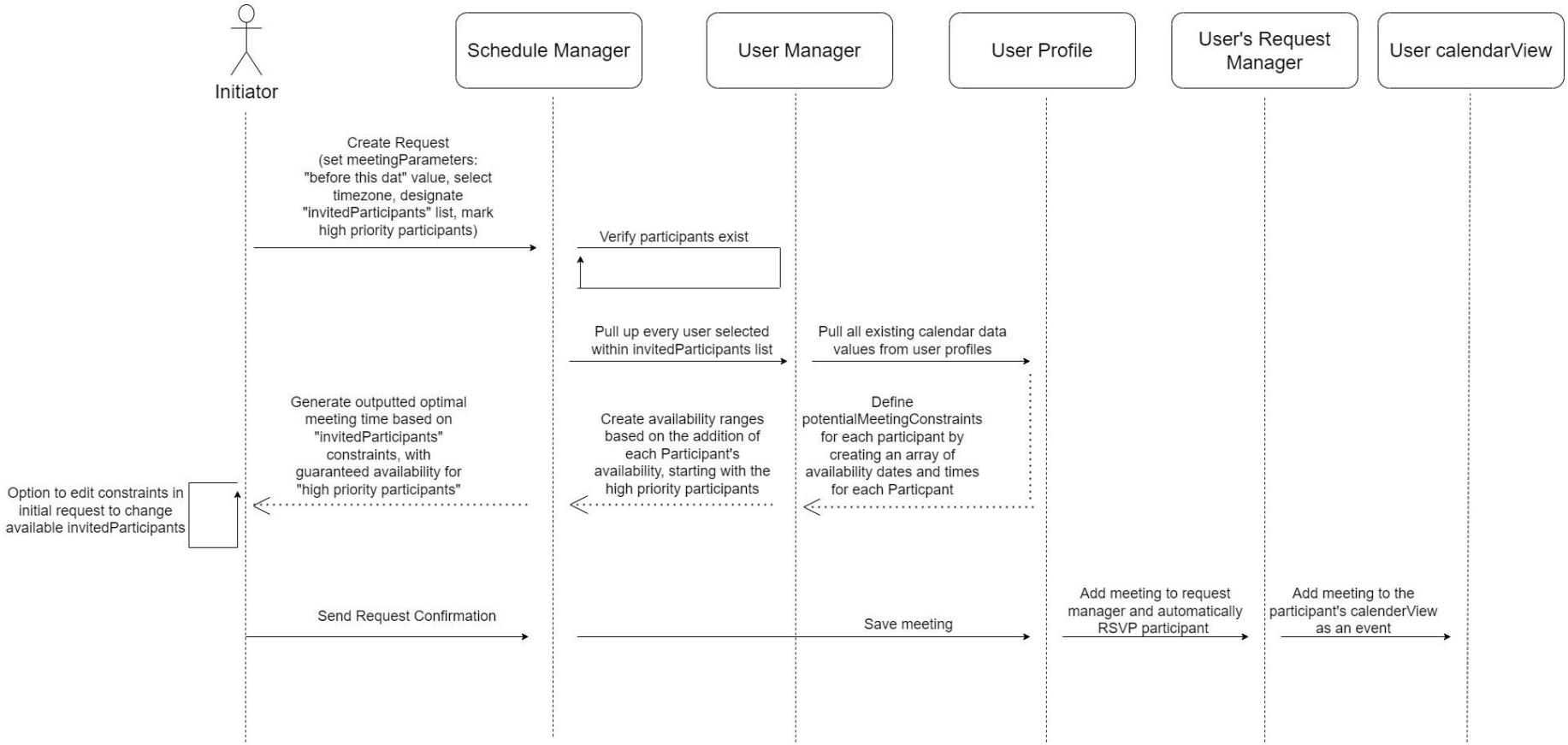

Scheduling Manager
public OptimalMeetingTime
public potentialMeetingConstraints
public EditMeetingRequest(beforeDate, String location, String state, String country, String materialNeeded, String duration, String time, [ ] invitedParticipants ) // this method will be used to edit the meeting request
public EmailNotification() //if a meeting is cancelled, this method will send an email notification to the initiator.
Involves
Meeting Request
public meetingParameters
public [ ]invitedParticipants
public [ ] highPriorityInvitees
private String beforeDate
private String location
private String state
private String country
private String materialNeeded
private String duration
private String time
public createMeetingRequest(beforeDate, String location, String state, String country, String materialNeeded, String duration String time, [ ] invitedParticipants ) //this method will create the meeting request with the specified parameters.
public requestConfirmation() //send out the request confirmation to all invitedParticipants
Involves
Participant
public String Name
public changeRSVPStatus() //This method will be used If a participant changes their RSVP status for a meeting.
Contains
User Manager
public Users[ ]
public [ ] availabilityDates
public [ ] availabilityTimes
public retrieveUserData() //this method will let the scheduler retrieve user data.
Generates
Participant Profile
public Settings
public RequestManager
public [ ]confirmedParticipants
public [ ] unavailableParticipants
public AddMeetingRequest(meeting request) //this method will be used to add the meeting request to all of the invitedparticipants request managers.
public ReceiveRequestConfirmation(requestConfirmation()) //this method lets the participants view the meeting request and decide if they want to confirm their attendance or not.
public removeMeetingEvent() // this method will remove the meeting from the user's profile.
Generates
Initator
public String Name
public confirmMeetingRequest() //Once the Initator is satisfied with the request they'll confirm it.
public rescheduleMeeting() //this method will manually reschedule the meeting.
Generates
Request Manager
public Meeting
public [ ] upcomingMeetings
public MeetingConfirmation(meeting) //this method will give a notification to the user saying that they have a meeting on the specified date, time, and location.
public AutomaticRSVPParticipant() //this method will automatically RSVP the participant once the meeting has been added to the request manager.
public sendMeeting(meeting) //this method will send the meeting information over to the users calendar so it can be added to calendar.
Generates
User calendarView
public String monthName
public String calenderUnits //Square[5X7] all rows & columns
//Draws out all the boxes for 5 weeks of the calendar
public String allBoxes: calenderUnits
int datesNumbered //the count of the dates for each calenda
int DayOne //gives the initial day of the month
public calendarView(meeting )
//this method will show the participants calendarView where they can view the upcoming meetings
public AddMeetingToCalendarView(meeting) //this method will add the meeting to the participants calendar
Generates

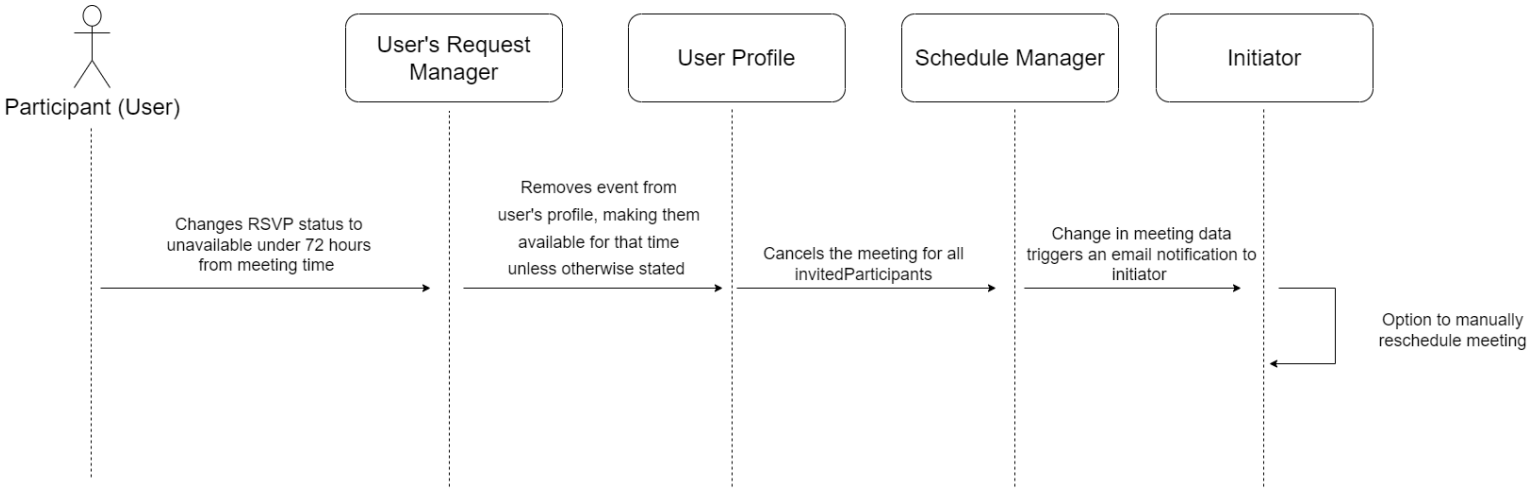
(Figure 1.5)

# 5. Sequence Diagrams / Behavioral Models

The first behavior diagram (Figure 1.6) shown below demonstrates the process of scheduling a *meeting*. This process starts when the initiator creates a meeting *request*, where they will outline the *meetingParameters*. One of the requirements of completing *meetingParameters* is determining the *invitedParticipants* list and selecting any *highPriorityParticipants* from the list. Once completed, the *Meeting Scheduling System* will verify that the selected *participants* exist within the system. All *users* are stored in the *User Manager*, which will pull all calendar data from the individual user profiles. The calendar data will help create *potentialMeetingContraints* for each *user* which will be compiled to create availability ranges by the *Meeting System Manager.* All of this information, when compiled for all users, will provide the *initiator* with *optimalMeetingTime*(s). If the meeting is too far away, or there is no availability found in the next five weeks, then the *initiator* has the option to change the *meetingParameters*. If the *optimalMeetingTime* is finalized then the *initiator* can confirm the *request*. This will save the meeting in the *Request Manager* and *calendarView* of all the *invitedParticipants*, which will also send all *participants* a *notification*.

The second behavior diagram (Figure 1.7) demonstrates the process of a meeting rescheduling under 72-hours from the *meeting* time. While the *Schedule Manager* will automatically reschedule a meeting using the originally provided *meetingParameters* (not shown in this diagram), when there are less than 72 hours left prior to the meeting, the event is canceled for all *invitedParticipants*. The *initiator* is then emailed about this cancelation and has the option to reschedule the meeting.


Initiator
Schedule Manager
User Manager
User Profile
User's Request Manager
User calendarView
Create Request (set meetingParameters: "before this dat" value, select timezone, designate "invitedParticipants" list, mark high priority participants)
Verify participants exist
Pull up every user selected within invitedParticipants list
Pull all existing calendar data values from user profiles
Generate outputted optimal meeting time based on "invitedParticipants" constraints, with guaranteed availability for "high priority participants"
Create availability ranges based on the addition of each Participant's availability, starting with the high priority participants
Define potentialMeetingConstraints for each participant by creating an array of availability dates and times for each Particpant
Option to edit constraints in initial request to change available invitedParticipants
Send Request Confirmation
Save meeting
Add meeting to request manager and automatically RSVP participant
Add meeting to the participant's calenderView as an event




# 6. Conclusion:

If all of the above requirements are implemented in the meeting scheduler software, the process of setting up *meetings* will be easier for the company members. The software to be will be able to find an optimal meeting time, place, and necessary equipment for all of the *participants* that want to have a *meeting*.

When it comes to meeting attendance, there is the possibility that one or two invited *participants* are unable to attend a *meeting* which causes a problem for the other *participants* who were ready for the meeting. The software resolves that issue by letting the *initiator* give their approval to reorganize the meeting. If the *participant* declines an invitation to attend a meeting in advance of the meeting time, the *meeting* will automatically get reorganized to the next available time that fits all of the participants' *constraints*.

To aid user experience, the software converts the scheduled time for the meeting into the time zone that the user specified in their *settings*. Converting the meeting time into the time zone helps *participants* not get confused about meeting times if their *meeting* is scheduled at a different time than their time zone.

Even though all of the requirements will be able to help schedule meetings more efficiently, there needs to be attention to the security aspect of the software. There could be times when a *user* forgets to log out of their account so the software should be able to log them out after ten minutes of inactivity. In order to prevent unauthorized people from entering an online *meeting*, there should be a passcode that a *participant* would have to type in so they can access the *meeting*. If these next steps are implemented, it would help keep *users'* information safe on the scheduler and it would help keep *meetings* protected from unauthorized people.

# 7. Definitions and Glossary:

These terms are intended to have precise meanings in the document and other application deliverables. They appear in *italics* throughout the document.

1. Approval: This is the finalization agreement from the participants, where they are agreeing that the constraints meet their needs.
2. Authorization Account Holders: These are the individuals that own administration power to override requests and address technical issues.
3. CalenderView: This is the calendar where the *users* can view their upcoming *meetings* and any personal events they have.
4. Confirmed Group: These are participants that have confirmed the meeting request stating that they have no issues with the constraints.
2. Constraints: These are important factors that can change the meeting's finalization within the system. These include the date, duration, location, room, and equipment access.
3. Data: The content of private or work calenders that is being transferred into the software system in order to project availability options to the initiator.
4. Exclusion Group: This is the group of participants that do not have any availability in the range or end up canceling the meeting that can't be rescheduled.
5. highPriorityParticipants: These are the participants that need to be in the meeting at all costs.
6. Initiator: This is the person that is in charge of sending the request and who is trying to organize the meeting.
7. invitedParticipants: This is the list of participants that need to be in the meeting.
8. KnowledgeofParticipantContraints: This is the scheduling system being aware of all constraints for all users that may relate to time, location, equipment, resources, accommodations, traveling requirements, etc., prior to the request of a meeting being sent out. Since meetings can be scheduled up to one month in advance, this means all constraints for the month need to be established by the system.
9. MaximumAttendance: this implies that a meeting has all of its participants present or at least 80% of its participants.
10. Meeting: This is the event that needs to occur between associated parties (initiator and participants).
11. Notification: This is the notification that will appear on a participant's email or phone telling them they have an upcoming meeting which is sent at the start of the week. This notification will have the meeting time, date, location, and address if the meeting is outside in a different space. If the meeting is virtual, a participant will get a link to access the meeting.
12. Optimal Group Availability: This is the output the WSS meeting scheduler would output upon all constraints and participant lists being imputed by the initiator.
13. OptimalMeetingTime: This is the meeting time that a meeting can be scheduled which should work for all of the scheduled participants. This must be within 5 weeks of the current date.

14. Participants: These are the individuals that need to be in the meeting and are sent the request by the initiator.
15. Personal Constraints: These are the constraints that control when a participant can show up for a meeting like travel/vacation, working hours, preferred location, etc.
16. Personal Calendar: This is the calendar that is on the user's profile. It has all of the user's personal events and their upcoming meetings.
17. Range: This is the initial set of dates that the initiator will put into the system for it to output acceptable time availabilities. Ranges can only look at two-week ranges from the present day. This is the range that the system will look at when automatically presenting initiators with meeting time options. If a wider range is needed, it must be set manually.
18. Request: This is the meeting creation documentation sent by the initiator to the participants.
19. Scheduling Manager: This will hold the meeting requests and find the optimal time to create the meeting.
20. Settings: This is where a *Participant* can go to change their constraint preferences, add personal calendars, and adjust any personal contact information.
21. Time Zone: A range of longitudes where a common standard time is used, set using the Participant's computer time clock settings.
22. User: This is the general term for everyone one who uses the software. This is the basic term before they become an initiator or participant for certain cases.
23. User Manager: This is where all users and their profiles are stored.

# 8. Appendix:

## Risk Analysis

Even though the meeting scheduler will help people book *meetings* more efficiently, there are still some potential risks that can happen for the *users* of the scheduler. In this chart, there are three potential risks that can happen. The first potential risk is that *MaximumAttendance* is not achieved for a meeting. In the subsequent boxes below Risk #1 are two reasons why *MaximumAttendance* is not achieved. The second risk is that a participant's constraints have an inconsistent format that does not work with the scheduler. In the boxes below are two reasons why this potential risk can happen.

The third risk is that a user's personal calendar is compatible with the meeting scheduler system. In the boxes below are two reasons why a user's personal calendar could be incompatible with the meeting scheduler system. These three risks were used to help create the risk models.

Risk Summary

| Risk #1:<br>*MaximumAttendance* not achieved | Risk #2:<br>Inconsistent formatting for participant constraints | Risk #3:<br>Risk of PersonalCalendarIncompatability |
|---|---|---|
| Description: When scheduling meetings, there is a risk that the meeting does not have all of its participants that were scheduled to attend. A participant could not attend a meeting because they did not know where the location of the meeting was. Another way that a meeting could not reach *MaximumAttendanc*e is that users could forget to update their schedules which would result in future meetings getting scheduled incorrectly. If the meeting gets scheduled incorrectly then a participant could not attend a meeting because their calendar listed that they were busy despite being at | Description:<br>Since meetings are going to be made with international people, they could use different date formats and time formats on their calendars as opposed to others which could cause problems when scheduling a meeting. | Description: The meeting scheduler system can't automatically assume that it will be compatible with every calendar system in the market, as it will require the ability to customize established data within the user's personal calendar system. Some calendar systems don't have a share option, which prevents the scheduler from uploading the information automatically. There is also no personal availability input option in to the meeting scheduling system. |

| | | |
|---|---|---|
| the moment the meeting was scheduled for. | | |
| Why does this risk happen? Participants did not appear at the meeting because they didn't know where the location of the meeting was. | Why does this risk happen? Participants use different time formats on their calendars i.e. regular time vs military which makes it difficult for the scheduler to determine if their free at the time the meeting is going to be held. | Why does this risk happen? No calendar data transfer option is available for their current platform. |
| Why does this risk happen? Another reason why this risk could happen is that users could forget to update their schedules which would result in future meetings getting scheduled incorrectly. | Why does this risk happen? Participants use a different date format on their calendars than what the scheduler is equipped for which makes it difficult for the scheduler to determine if they are free on the day the meeting wants to be held. | Why does this risk happen? A personal calendar is stored on a non-digital platform. |

**Cost / Value Trade-Off Analysis**

This is a graph of the cost/value trade-off. The requirements that were deemed high priority were put in the high priority section. The requirements that were deemed medium priority were put in the medium priority section. The requirements that were deemed low priority were put in the low priority section. The requirements in the high priority section are the most useful to the users because they deal with the *initiator* booking a *meeting* for the *participants*.

The User Settings requirement has a cost of 10 on the cost percentage scale going from 5 to 50 because it is not difficult to make the scheduler have enough storage to be able to hold every *user's* constraint data. The value is 50 because the *user's* constraints will decide when the *meeting* is going to be held. The Cumulative View of all Meetings has a cost of 20 because the software needs to let the participants view their upcoming *meetings* The value is 50 because this lets the *participants* know when their *meetings* are which would help prevent them from missing them.

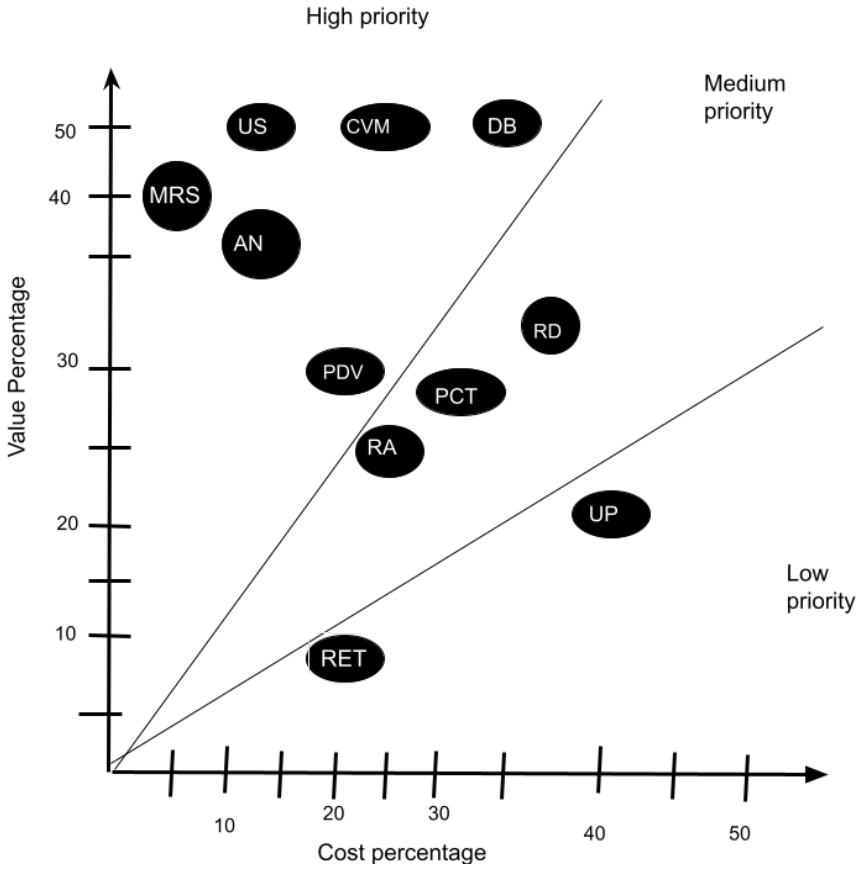
The Double Bookings requirement has a cost of 30 because the software would have to check the *participants'* calendars to see if there are two meetings at a one-time slot. Then, the software would have to book the second *meeting* to be at a later time which makes it difficult to develop. The value is 50 because this requirement solves the issue of double bookings. The Participant Data Viewing requirement has a cost of 20 because it would have to convert the event titles in the calendar into binary which would be difficult and time-consuming to do. The

value is 30 because this requirement is only helpful in two aspects. It protects the data from other *users* and it's helpful for the *initiator* to set the meetings.

The Participant Calendar Transfer requirement has a cost of 30 because the software must be able to accept data from personal phone calendars as well as business calendars like Microsoft Calendar. The value is 30 because this would save the *users* time from having to type in their availability into the software. The Reorganization Declined requirement has a cost of 36 because the software would have to give the *authoritative account holders* access to the system. The value is 30 because this requirement is only helpful when a *meeting* needs to be rescheduled. The Reorganization Approved requirement has a cost of 25 on the cost percentage scale going from 5 to 50 because it would not be difficult for the software to reorganize the *meeting* to the next available time slot. The value is 25 because this requirement is only helpful if the *participant* knows ahead of time that they won't be able to attend a *meeting*.

The Meeting Request Submission has a cost of 5 because it would be difficult to make the scheduler create the *meeting request*. This has a value of 40 because this is the first important step in creating a meeting. It has to be able to accept all of the *meetingParameters* in order to create the *meeting*. The Agenda Notification requirement has a cost of 10 because it would not be difficult to create a notification that would alert a *participant* of their upcoming *meetings*. The value is 35 because this *notification* helps prevent *participants* from missing *meetings*.

The User Personalization requirement has a cost of 40 because the software needs to know every existing timezone and language. The value is 20 because this requirement is helpful once the *meeting* is set. It helps a *participant to* not get confused as to when a *meeting* is happening if they are in a different time zone. The Required Effort and Troubleshooting requirement has a cost of 20 because this feature would not be difficult to develop. The *authoritative account holders* would just need access to the system. The value is 10 because the effort would only be needed during rescheduling meetings.




FR 1.1 - User Settings (US)
FR 1.2 - Personal Calendar Transfer (PCT)
FR 1.3 - Participant Data viewing (PDV)
FR 1.4 - User Personalization (UP
FR 1.5 - Required Effort and Troubleshooting (RET)
FR 1.6 - Cumulative View of all Meetings (CVM)
FR 1.7 - Agenda Notifications (AN)
FR 1.8 - Meeting Request Submission(MRS)
FR 1.8. - Reorganization Declined (RD)
FR 1.9 - Reorganization Approved (RA)
FR 1.10. - Double Bookings  (DB)