# SOFTWARE ENGINEERING: CONCEPTS AND METHODS Assignment 1

# Problem: A Meeting Scheduler System

In this course work, you are the software developers, and your customer is Dr Mehmet B Özcan. A working document of a Meeting Scheduler System is provided, which can be found in Appendix 1. This document is vague because your customer has not made up his mind about some of the features outlined in the document as he lacks technical skills to judge the feasibility of their implementation. Acquisition, specification and validation processes are needed to complete it and lift any shadowy areas. Hence, you are required to develop a rapid software prototype because a prototype focusing on some of these features is likely to help him crystallise his thoughts. Note that due to its sheers complexity, you are expected to focus only on specific functionality of this system. This will be outlined next (see “what should you develop and deliver?”).

# What should you develop and deliver?

You are required to carry out the following tasks:

1. develop a rapid software prototype that addresses **the concurrency aspects of the problem domain and conflicts that arise as a result of location and equipment allocated to meeting rooms**; If you focus on other aspects of the problem, your marks will suffer greatly no matter how good your prototype may be.
2. identify a set of questions that may be pertinent to prototype/prototyping/requirements/requirements validation and that may be asked to users during user validation;
3. identify any issues, including assumptions, that will be used for prototype development and may subsequently be raised during user validation;
4. identify a set of user stories and corresponding acceptance tests that will be used as the basis for requirements validation for the chosen functionality as outlined in (1);
5. produce a set of test scripts (or scenarios) for user stories for validation purposes. These tests (or scripts) should model how the intended software system may be used from the users' perspective and should also model what can go wrong while executing them.

# What should be included in your submission?

1. The final version of your prototyping software. Your prototype should be developed using the Visual Studio 2019 environment. The choice of language is up to you.
2. A document that may contain the following:
3. any questions that may be pertinent to prototype/prototyping/requirements/requirements validation and that may be asked to user during user validation;
4. any issues, including assumptions, that may be raised during user validation;
5. User stories and associated acceptance tests (including your plan of their execution for validation purposes) that may be used during the validation process. Note that the actual validation process will not take place;
6. Test (or scenario) scripts. These should be prepared for each user story.
7. URL link to a YouTube video showcasing your prototype. This video should clearly show how each user story and its corresponding acceptance test(s) are executed. You can, if you so wish, pose the questions you prepared for submission, during the course of your video demonstration. You can even talk about various assumptions you made while implementing your prototype. The length of the video should not be longer than 20 minutes.
8. Your completed peer assessment form.

**Submission details**

You must submit all artefacts outlined above to the Blackboard assignment handler using the following guidelines:

1. your prototype as outlined in (1) above should be compressed into a single 7z file (NO RAR files);
2. your document as outlined in (2) above should be submitted as a MS Word document;
3. user stories and acceptance tests as outlined in (3) above should be submitted as a MS Word document;
4. test (or scenario) scripts as outlined in (4) above should be submitted as a MS Word document;
5. URL address of your YouTube video;
6. Your peer assessment form as outlined in (6) above should be submitted as a MS Word document.

**Important note about submission**

You must carry out installation testing before submitting your work. That is, when your 7z file is expanded it should install and run smoothly without any issues.

**Handing in date**

10th December 2020by 15:00.

**Grading Criteria**

This work will be graded using the criteria given below and grade ranges are taken from the document that you can read in Appendix 2.

|  |  |  |
| --- | --- | --- |
| **Criterion** | **Available Marks** | **Notes** |
| Development of prototype | 55 | * You show very good understanding of the prototype construction processes. * It is clear from your prototype which assumptions that you make about the software requirements are demonstrated. * Your prototype represents requirements accurately, and demonstrates the chosen functionality of the eventual system without problems. * Your prototype is fit for purpose and can be regarded as an effective medium through which the requirements can be negotiated. * You think outside the box, which means that you implement a feature (or an aspect) of the system beyond the information provided in Appendix 1. This may be regarded as a new requirement that has not been thought of before. |
| Documentation | 10 | * You prepare a set of questions that need to be clarified during requirements validation. * You document any assumptions, which you made during the prototype construction process, and which you would wish to clarify with users during requirements validation. |
| User stories and acceptance tests | 15 | * You are able to prepare realistic and numerous user stories and their acceptance tests. * You are able to consider alternate courses of user stories for discussion. * User stories and their acceptance tests have the potential to extend the requirements because they include aspects that were not considered in the initial document (refer to Appendix 1). |
| Test (or scenario)scripts | 20 | * You are able to produce a clear structure/plan in the form of test (or scenario) scripts, which will enable your prototype to be executed without any help from you. * You are able to link test (or scenario) scripts to user stories and acceptance tests, so that their relationship can clearly be seen for validation purposes when your prototype is executed. * Test (or scenario) scripts model how the intended software system may be used from the users' perspective * Test (or scenario) scripts a model what can go wrong while executing them. |

**APPENDICES**

**APPENDIX 1: THE MEETING SCHEDULER SYSTEM: PRELIMINARY DEFINITION[[1]](#footnote-2)**

**FOREWORD**

This preliminary description is deliberately intended to be sketchy and unprecise. Acquisition, specification and validation processes are needed to complete it and lift many shadow areas.

**SCHEDULING MEETINGS: DOMAIN THEORY**

Meetings are typically arranged in the following way. A **\*meeting initiator\*** asks all potential meeting attendees for the following information based on their personal agenda:

* a set of dates on which they cannot attend the meeting (hereafter referred as **\*exclusion set\***);
* a set of dates on which they would prefer the meeting to take place (hereafter referred as **\*preference set\***)

A **\*meeting date\*** is defined by a pair (calendar date, time period). The exclusion and preference sets are contained in some time interval prescribed by the meeting initiator (hereafter referred as **\*date range\***).

The initiator also asks **\*active\*** participants to provide any special equipment requirements on the meeting location (e.g. overhead-projector, workstation, network connection, telephones, etc.); he/she may also ask **\*important\*** participants to state preferences about the meeting location.

The proposed meeting date should belong to the stated date range and to none of the exclusion sets; furthermore it should ideally belong to as many preference sets as possible. A **\*date conflict\*** occurs when no such date can be found. A conflict is strong when no date can be found within the date range and outside all exclusion sets; it is weak when dates can be found within the date range and outside all exclusion sets, but no date can be found at the intersection of all preference sets. Conflicts can be resolved in several ways:

* the initiator extends the date range;
* some participants remove some dates from their exclusion set;
* some participants withdraw from the meeting;
* some participants add some new dates to their preference set.

A meeting room must be available at the selected meeting date. It should meet the equipment requirements. A new round of negotiations may be required when no such room can be found.

The meeting initiator can be one of the participants or some representative (e.g. a secretary).

**'SKETCHY' SYSTEM REQUIREMENTS**

The purpose of the MEETING SCHEDULER SYSTEM is to support the organisation of meetings- that is, to determine, for each meeting request, a meeting **\*date\*** and **\*location\*** are required so that most of the intended participants will effectively participate. The meeting date and location should thus be as convenient as possible to all participants.

Information about the meeting should also be made available as early as possible to all potential participants. The intended system should considerably reduce the amount of overhead usually incurred in organising meetings where potential attendees are distributed over many different places. On another hand, the system should reflect as closely as possible the way meetings are typically managed (see the domain theory above).

The system should assist users in the following activities.

* Plan meetings under the constraints expressed by participants
* Replan a meeting dynamically to support as much flexibility as possible. On one hand, participants should be allowed to modify their exclusion set, preference set **\*before\*** a meeting date/location is proposed. One the other hand, it should be possible to take some external constraints into account **\*after\*** a date and location have been proposed - e.g., due to the need to accommodate a more important meeting. The original meeting date or location may then need to be changed; sometimes the meeting may even be cancelled. In all cases some bound on replanning should be set up.
* Support conflict resolution according to resolution policies stated by the client.
* Manage all the interactions among participants required during the organisation of the meeting - to communicate requests to get replies even from participants not reacting promptly, to support the negotiation and conflict resolution processes, to make participants aware of what is going on during the planning process, to keep participants informed about schedules and their changes, to make them confident about the reliability of the communications, etc.
* Keep the amount of interaction among participants (e.g., number and length of messages, amount of negotiation required) as small as possible.

The meeting scheduler system must in general handle several meeting requests **\*in parallel\*.** Meeting requests can be competing by overlapping in time or space. Concurrency must thus be managed.

The following aspects should also be taken into account.

* The system should accommodate decentralised requests; any authorised user should be able to request a meeting independently of his whereabouts.
* Physical constraints may not be broken - e.g., a person may not be at two different places at the same time, a meeting room may not be allocated.
* The system should provide an appropriate level of performance, for example:
* the elapsed time between the submission of a meeting request and the determination of the corresponding meeting date/location should be as small as possible;
* the elapsed time between the determination of a meeting date/location and the communication of this information to all participants concerned should be as small as possible;
* a lower bound should be fixed between the time at which the meeting date is determined and the time at which the meeting is actually taking place.
* Privacy rules should be enforced; a non-privileged participant should not be aware of constraints stated by other participants.
* The system should be usable by non-experts.
* The system should be customisable to professional as well as private meetings. These two modes of use are characterised by different restrictions on the time periods that may be allocated (e.g., meetings during office hours, private activities during leisure time).
* The system should be flexible enough to accommodate evolving data - e.g., the sets of concerned participants may be varying, the address at which a participant can be reached may be varying, etc.
* The system should be easily extendable to accommodate the following typical variations:
* handling of explicit status and priorities among participants;
* handling of explicit priorities among dates in preference sets;
* handling of explicit dependencies between meeting date and meeting location;
* participation through delegation - a participant may ask another person to represent him/her at the meeting;
* partial attendance - a participant can only attend part of the meeting;
* variations in date formats, address formats, interface language, etc.

**EXTENDING THE REQUIREMENTS**

This extension aims at suggesting a useful extension to the Meeting Scheduler System. The objective is to incorporate knowledge about participant status and about various kinds of priorities among participants and meetings.

**Finding Best Meetings and Resolving Conflicts**

Clients and analysts came to the conclusion that knowledge about participant status and about priorities about users and meetings should help in resolving conflicts by determining a "best" way to resolve a conflict. Even when there is no conflict, the participant status may be useful in determining a "best" meeting date and location.

**Status and Priorities**

The following notations should be incorporated in the proposed extension. They capture the hierarchical importance of participants, the importance for a participant to attend a particular meeting relatively to other participants or to other meetings, and the ease with which a participant can make a particular date interval free. These various notations will be used in the conflict resolution process.

**Participant Status**

The participant **\*status\*** captures the hierarchical importance of a participant with respect to others independently of any specific meetings he is expected to participate in. The participant **\*status\*** might be used, e.g., to determine a "best" compromise on date and location whenever several ones are possible.

The participant **\*status\*** is typically determined by some super user. For instance, in the context of scheduling Faculty meetings the Department Head would be a higher **\*status\*** than normal professors. The latter would have a higher **\*status\*** than student representatives.

**Participant Importance**

The participant **\*importance\*** captures the importance for a specific person to attend a particular meeting **\*relatively\*** to other participants. Participant **\*importances\*** are typically determined by the meeting initiator. For instance, the meeting chairman and secretary must be present; they have the highest participant **\*importance\*.** In a project meeting where specific tasks are discussed, the task leaders would have a higher participant **\*importance\*** than normal project members and a lower importance than the meeting chair, the task speakers or the project reviewers.

**Meeting Significance**

The meeting \***significance\*** represents the importance for a specific person to attend a particular meeting **\*relatively to other meetings or meeting requests\***. Meeting **\*significances\*** are typically determined by the participants concerned. For instance, participants to a specific task in a research project would assign a greater significance to a project meeting where their tasks will be discussed. This must be kept confidential.

**Participant Flexibility**

The participant **\*flexibility\*** is intended to indicate how easily a user can make a particular date interval free to allow meetings to be scheduled within that interval. Dates in exclusion sets and/or preference sets can thus be weighted accordingly.

The participant **\*flexibility\*** is typically determined by the participants concerned. For instance, professors cannot move lecture periods easily, their participant **\*flexibility\*** for the corresponding date intervals should be low. A date interval which is not in the exclusion set of a participant should have a high **\*flexibility\*** for that participant. This information must be kept confidential.

**Using Knowledge about Status and Priorities**

The following tactics illustrate some typical uses of the various kinds of priorities suggested above.

1. Best meeting dates and locations should be determined by considering participants with higher participant \*status\* first.
2. If no date can be found to organise a meeting, the Meeting Scheduler System could propose a person having low participant **\*importance\*** to withdraw from the meeting.
3. If no date can be found to organise a meeting, the Meeting Scheduler System could propose a participant to cancel (or to withdraw from) another meeting having a lower meeting \*significance\*.
4. A meeting date within some exclusion set (or outside some preference set) could be considered if the corresponding participant has a high **\*flexibility\*** for it.

1. Axel van Lamsweerde, Robert Darimont and Philippe Masonet

   UCL - Unite d'Informatique

   B-1348 Louvain-la-Neuve Belgium [↑](#footnote-ref-2)