System Design Document

Introduction

Design goals

Usability: The Calendar system should be designed with the goal in mind that it should be fairly easy to use by the Client.

Response time: The Calendar system should be quite responsive to user input, and commonly make actions in a matter of milliseconds.

Extensibility: The Calendar system should support the opportunity to extend the program and make it interact with different calendar programs such as Google Calendar, Exchange and iCal.

Proposed system

Overview

Architectural style
We decided to use the MVC pattern as the basic architectural style for the Calendar system.

The MVC pattern separates data and user interface and has Controller as a intermediated component. The pattern is used when there are multiple ways to view and interact with data.

MVC is a abbreviation for Model View Controller. The software is divided in these components. The components are defined as following:

- **Model** handles data.
- **View** is attached to model and presents the data.
- **Controller** is the link between model and view.

To get an overview of the MVC pattern we outlined the pros and cons.
Pros

- low coupling
- high cohesion
- code maintenance
- code reuse

Cons

- code complexity
- development time

In our Object Oriented Analysis an Analysis Object Model was defined where we identified:

- Entity objects (*model*)
- Boundary objebts (*view*)
- Control objects (*controller*)

The Analysis Object Model is a simplification of MVC pattern. It made it eaiser to set up the MVC pattern and build further on the Object Oriented Design.

Subsystem decomposition

The subsystem CalendarManagement is the system where most of the program is placed. This system is responsible for all the local events happening in the system. This means that it would handle creation of the calendar, events and provide the necessary GUI for the local program.

This subsystem has two connections outside the subsystem itself. The first one goes to "ShareInterface"that has all the necessary functions that the CalendarManagement system could ask for when it is about to share a calendar or an event. The other outgoing connection is to the SyncInterface. This is an interface that holds all the needed method calls to synchronize the calendar. Behind the two interfaces we have the subsystems ShareManagement and SyncManagement.

The Share Management has to do with sending the information to the right people and using the right systems. The reason why we made this an interface is that it has been a commonly thing to share through facebook and other social media platforms. Therefore it would be nice to easily have the opportunity to extend the share option so that it would support e.g. Facebook.

The subsystem SyncManagement is where we keep our calendars up to date, where we got the information, of whom the calendar belongs to, and which users that are invited to see it. The syncManagement keep the client calendars synchronized and handles the updates between the local calendar and the online ones. This is also behind an interface so that we can easily exchange it. That would turn to our advantage so the maintenance and upgrades of the system could be done easy. You simply exchange the subsystem part with a new one. E.g. if we wanted to update our synchronization part with a newer version of a data base program, the syncManagement is exchanged, but the rest could stay intact.