Software Engineering Project – DAT255 - lp4, 2013Chalmers University of Technology, 23.05.2013



Fredrik Einarsson - Niklas Johansson - René Niendorf   
Anders Nordin - Sofie Peters

Chalmers On The Go – the Complete Chalmers Experience

This document describes the product vision, examples of user stories, requirements, design decisions, examples of tests carried out, change log, release history and end result of the development of the Android application ChalmersOnTheGo. Note that only examples of user stories, associated acceptance tests and unit tests are described in this document. For details, see document [Test Report](Test%20Report.docx).

Software Development Document

For ChalmersOnTheGo 1.0, Jelly Bean 4.0 and API 16

Software Development Document

Table of Contents

1 Project description 2

2 Product vision 2

2.1 Core features 3

2.2 Non-core features 3

3 Developing 3

3.1 User stories 3

3.2 Product backlog 4

3.2.1 Functional requirements 4

3.2.2 Non-functional requirements 5

3.2.3 Burn down chart 5

3.3 Design decisions 5

3.4 Testing 6

3.4.1 Acceptance tests, examples 6

3.4.2 Unit tests, examples 7

3.5 Change log 8

3.6 Release history 8

4 Resulting product 8

# Project description

The Android application ChalmersOnTheGo was designed, developed and released as the main task of the course Software Engineering Project, DAT255, on the Chalmers University of Technology, April-May, 2013. Android was the operating system of choice on the course. The development team consisted of Fredrik Einarsson, Niklas Johansson, René Niendorf, Anders Nordin and Sofie Peters. The tools used during the development process were the web-based hosting service Github using the Git revision control system, the agile development planning tool Pivotal Tracker and the IDE Eclipse.

# Product vision

The main vision for the product was to create a map application over the Chalmers area, giving the user not only a regular map, but instead “the complete Chalmers experience”. This meant that the user should not only be able to find locations, such as those found with addresses on Google Maps and similar applications. The user should instead be able to search inside Chalmers buildings for Chalmers specific names on locations, such as group rooms, lecture halls or pubs. Moreover, the application should contain some other features than the core map-features, features with connection to the “Chalmers experience”. The application should be useful for the target group of the product: Chalmers students and people new to the Chalmers area. It should furthermore be intuitive and fun to use. It would have the edge over other map applications, since ChalmersOnTheGo would provide ways to find locations inside buildings, as well as specific Chalmers buildings with Chalmers specific names, not needing the exact address. An essential constraint to make the vision realistic, considering the development team’s limited time resources, was that the actual location data would be limited to show the application’s potential, not the complete Chalmers area.

## Core features

* Searching and getting marked Chalmers specific locations
* Searching on Chalmers specific names, not adresses
* Getting the user’s current location

## Non-core features

* Section specific colouring of buildings when searching for pubs
* Logos for pubs and section buildings
* Night and day mode

# Developing

In this chapter user stories, product backlog and requirements presented, as well as design decisions, test examples, change log and release history.

## User stories

In the text below, a cross-section of the user stories used when developing the application are presented. For complete list, see document [Test Report](Test%20Report.docx). The subscript number after most user stories are identical to the subscript number after the associated requirements in section 3.2 Product backlog.

**General**

* As a user, I want to be able to reverse my actions with a back-button2

**Map**

* As a user, I should only be able to see and navigate inside the Chalmers area3

**Navigation**

* As a user having gotten a location marked on the map, I want to be able to get the shortest path from my current location to the wanted location, by clicking the locations information window6

**Searching**

* As a user typing in a search for some item, I want a dropdown menu to appear with word-completed suggestions8
* As a user searching and getting suggestions, I want to be able to click any suggestion and get the location I clicked marked on the map9

**Layer function**

* As a user, I want to be able to have a checkbox-regulated layer function where I can choose between location types13
* As a user, I want to have layers with computer rooms, lecture halls, group rooms and pubs15

**Different application modes**

* As a user, I should be able to switch between night and day mode at any point or time in the application

**Non-navigational features**

* As a user, I want useful and fun features not only concerned with the map and navigation

**Design**

* As a user, I want to be able to reach all the application’s functions from a menu system similar to Google maps17

**Performance**

* As a user, I want the application to perform wanted actions reasonably fast22

## Product backlog

In the sections below, you will find the functional and non-functional requirements used when developing the application. Most of them are extracted from different user stories and have evolved during the development process. There is no internal prioritising in the listing. The subscript number after most requirements are identical to the subscript number after the associated user story in section 3.1 User stories and in the document [Test Report](Test%20Report.docx). Requirements without a subscript number have no associated user story. At the end of the section, you will also find a burn-down chart of the development process, extracted from Pivotal Tracker.

### Functional requirements

**General**

* The application will have an ”exit-button” which exits the application1
* The application will have a “back”-button, reversing performed steps, in accordance with typical Android design2

**Map**

* When clicking a marked location, a popup window will show up, containing the location’s name, floor and a button for navigating to the location5

**Navigation**

* The application will have a ”target”-button, targeting the current location7
* The popup window on a marked location will contain a button which writes out the shortest path (according to Google) from the current location to the wanted location6

**Search**

* The application will have a ”search”-button which triggers a search field8
* The search field will provide word-completed suggestions when writing letters in it8
* The suggestions will concern all the possible alternatives with the chosen letters8
* Specific rooms, buildings or pubs can be searched8
* When clicking a search item, it/they will be marked on the map in the visible view9 & 11
* When a room is clicked, the closest entry to the room will be marked on the map10
* Generic room types can be searched12

**Layers**

* The application will have a ”layers”-button, showing checkbox options for the different possible layers12
* When checking one or more layers, those types of locations will show on the map14
* Layers can be unchecked14
* Possible layers will be group rooms, lecture halls, computer rooms and pubs15

**Database**

* A database containing information about locations will be set up8
* The database will provide the application with information when needed8

**Menu**

* The application will have a ”menu”-button which triggers a menu showing other options than the core features
* The menu will contain the “exit”-button1

**Non-core features**

* Step counter
* Drinking progress bars connected to step counter
* iCal synchronisation

### Non-functional requirements

**Appearance**

* The application icon will have an appealing design16
* The overall appearance shall be appealing21
* When searching, the suggestions will show icons related to the suggested location type, i.e. group room, computer room, lectura hall, pub and section building18
* When checking the ”pub”-layer, the buildings containing pubs will be painted in the respective section colours19
* Change colour scheme when changing between night and day mode

**Design**

* The menu system will be in accordance with typical Google maps design17
* The GUI shall be intuitive

**Support**

* If the GPS is not enabled, a window prompting this will show20

**Reliability**

* The application will not crash when used in the intended way

**Performance**

* Any chosen function should take 1 second or less to compute on a phone not older than 1 year22

**Constraints**

* The map will contain location data only for some parts of the Chalmers area
* The map will have boundaries, making it only possible to navigate inside the Chalmers area3
* The map can not be zoomed out of the Chalmers area3
* The map will have fixed coordinates in the starting view4

### Burn down chart

ADD BURN DOWN CHART HERE

## Design decisions

**External dependencies**

The application ChalmersOnTheGo depends on the Google Maps Android API from the Google Play Services library. Since the application’s core features concerns navigation on a map, the Google Maps dependency is major. The alternative to let the development team write a map was not found to be a realistic alternative, not with respect to time resources, nor to the team’s competence. An alternative to Google Maps discussed initially was the OpenStreetMap (<http://www.openstreetmap.org/>). However, the extensive support for Google Maps in Android software developing, along with it’s well known GUI as well as some previous experience with the API in some team members settled the decision.

**GUI**

To make the application intuitive and easy to orientate in for the user, the GUI was based on typical Android and Google Maps designs. These designs were deemed as both well known and keeping a high intuitive standard. All core features were to be reached from the main view, whereas the non-core features were to be reached via the “menu”-button.

**Double functionality**

A user who searches for and marks “computer rooms” for an example, can receive the same result when marking “computer rooms” as a layer option. However, both features are intuitive: a user should be able to search for anything in the search bar *as well as* checking any preferred layers. The layer function just provides a simplified way to do a specific generic search, a shortcut if you will. The team want the user to chose how to use the application, not the other way around. In addition, the layer function is a common map feature which might be missed if not implemented.

**Considered or removed functionality**

Due to limited time resources, the team’s final release only features data for some buildings, resulting in limited navigational possibilities. The product however provides a complete framework for “the complete Chalmers experience”, with the possibility for future developers to easily add location data, consequently expanding the navigational possibilities. This limitation was planned from an early stage.

Roles were briefly considered initially, such as “nolla”, “IT-student”, “student with 180 p” etcetera, but also initially deemed both difficult to implement and to motivate with user value. The greatest user value would have been the ability for the application to plot different shortcuts through buildings where the current user had access. This was however not deemed as valuable enough for such a heavy implementation.

A night mode/day mode functionality with mode associated activities and colour schemes was a planned feature. It was found to be superfluous however since the layer functionality could just as easily entail the intended night mode functions as the mode itself. In this way, the user would experience a more compact and intuitive GUI.

A user was initially meant to be able to search and plot the path between two different locations, with one search field for each location.

**Classes**

GRASP

Föreläsning 4, design choices:

Associtation, ex. ”A is a part of B – wing of airplane”, ”A is owned by B – plane of airline” etc.

Inheritance, composition and aggregation

**Packages**

GRASP

## Testing

In this section you will find examples of performed acceptance and unit tests, along with their associated user stories. For a complete test report, see document [Test Report.docx](Test%20Report.docx).

### Acceptance tests, examples

EXAMPLES

Acceptance tests have been run for the different user stories.

Skriv vilka ser stories de hör till

See more in document “Tests”

Acceptance testing – tests a user story (from a “custome” perspective) – the customer can accept or reject

### Unit tests, examples

* **Insertion and getting in table 1 (coordinates and buildings table), test suite**
  + insertIntoTable1 and getClosestEntry were tested together. Connected to user story “As a user, I want to be able to search for a room, mark it and get the closest entry to the room marked on the map”:
    - A pair of coordinates (Double) and a building name (String) were inserted into table 1 via insertIntoTable1.
    - The coordinates (Double) were used to create an object (LatLng) containing latitude and longitude.
    - The object (LatLng) and the building name (String) served as input in getClosestEntry.
    - The result of getClosestEntry (LatLng) and the object (LatLng) containing the coordinates were compared and found to be equal.
    - Calculating the closest entry
  + insertIntoTable2, insertIntoTable3, insertIntoTable4 and getAllRoomsInBuilding were tested together. Connected to user story “As a user typing in a search for some item, I want a dropdown menu to appear with word-completed suggestions”:
    - A room type (String) was inserted into table 2 via insertIntoTable2.
    - A false room type (String) was inserted the same way.
    - A building name (String) was inserted into table 4 via insertIntoTable4
    - Room name1 (String), a coordinate pair1 (Double), the true room type (String), the building name (String) and floor1 (String) were inserted into table 3 via insertIntoTable3.
    - Room name2 (String), coordinate pair1 (Double), the true room type (String), the building name (String) and floor2 (String) were inserted into table 3 via insertIntoTable3.
    - A false room name (String), coordinate pair1 (Double), the false room type (String), the building name (String) and floor1 (String) were inserted into table 3 via insertIntoTable3.
    - The building name (String) served as input in getAllRoomsInBuilding.
    - The result of getAllRoomsInBuilding (ArrayList<String>) was tested using methods size and contains, and found to be satisfactory.
    - Getting suggestions
  + insertIntoTable3 and getRoomCoordinates were tested together. Connected to user story “As a user, I want to be able to search for a building and get all the rooms in the building marked on the map”:
    - A pair of coordinates (Double) were used to create an object (LatLng) containing latitude and longitude.
    - A room type (String) was inserted into table 2 via insertIntoTable2.
    - A building name (String) was inserted into table 4 via insertIntoTable4.
    - The room name (String), the coordinates (Double, the room type (String), the building name (String) and a floor (String) were inserted into table 3 via insertIntoTable3.
    - The room name (String) served as input in getRoomCoordinates.
    - The result of getRoomCoordinates (LatLng) and the object (LatLng) containing the coordinates were compared and found to be equal.
    - Getting all rooms in a specific building

## Change log

* ”Log or record of changes made to a project, such as a [website](http://en.wikipedia.org/wiki/Website) or software project, usually including such records as bug fixes, new features, etc. Some [open source](http://en.wikipedia.org/wiki/Open_source) projects include a changelog as one of the top level files in their distribution.” (Wikipedia)

## Release history

Nice with a **release history** – brief

Inte obligatoriskt

“Före och efter typ”

# Resulting product

The product lives up to expectations. It contains both core features and non-core features, all of which contributes to “the complete Chalmers experience” as the development team interpreted the slogan.

**Final core features**

* Searching and getting marked Chalmers specific locations
* Searching on Chalmers specific names, not adresses
* Getting search suggestions when searching
* Getting the user’s current location
* Check and uncheck layers for specific location types

**Final non-core features**

* Stepcounter
* Drinking progress bar
* iCal synchronisation
* Section specific colouring of buildings when searching for pubs
* Logos for pubs and section buildings