NTNU – Trondheim
Norwegian University of
Science and Technology

TDT4240 Sofware Architecture

Requirements Documentation

Group 10 - A7:

Mats Byrkjeland
Andreas Drivenes
Anders Wold Eldhuset
Stein-Otto Svorstøl
Torstein Sørnes
Håkon Meyer Tørnquist

COTS: Android SDK

Game title: Sea Battle

Primary quality attribute:
Modifiability

Secondary quality attribute:
Availability

April 26, 2015

Contents

1 Introduction

This document contains the requirements for the Android game *Sea Battle*, an online two-player version of the famous *Battleship* game, where one player targets coordinates on the enemy's map, and bombs that coordinate. The enemy's map is initially hidden from the player, but as it is bombed, it will be revealed. A ship has been sunk when all its cells have been hit. The game is over when all the enemy's ships are sunk. You can see the cover of the traditional board game in figure 1.

A *game map* is a 10 x 10 grid. It contains the ships described in table 1, with given dimensions:

Ship name	Dimension
Aircraft carrier	5x1
Battleship	4x1
Submarine	3x1
Destroyer	3x1
Patrol boat	2x1

Table 1: The ships and their size in the game of Battleship

Figure 1: The cover of the traditional boardgame Battleship

Note that ships can be oriented either vertically or horizontally, but not diagonally.

We chose Battleship, because it is simple, and should not be too hard to implement. This will give us more time to focus on the architecture and network part of the game. We plan to make a *matchmaking* system that will allow a user to either create a *game room* and wait for another player to join, or join an existing game room. When there are two players in the same game room, the game will start.

We will begin the document by describing the functional requirements of our game, before we move on to the quality requirements, which will be described in scenarios. After that there will be a section on our COTS of choice, before we list references, issues and changes.

2 Functional Requirements

First some definitions:

User: The person that is operating the Android mobile application on "this" end.

Game: A round or instance of the game

Player: One who takes part in a game

With that in mind, let's present the functional requirements based on the game idea. They are given a priority, high, medium or low, based on how important they are to get a working game, and for our main quality attributes.

FR 1 The playing board is shown as a grid, a 10x10 matrix where one can place ships, see item **FR 2**. *High priority*

FR 2 A ship in the game is an entity with a name and a size. The ships are described in table 1 *Medium priority*

FR 3 The user should be presented with a menu consisting of a "New game"-button and a "Help"-button. *Low priority*

- **FR 3.1** The "New game"-button starts a new game, see item **FR 4**.
- **FR 3.2** The "Help"-button shows information about how the game works, see item **FR 10**.

FR 4 The user should be able to start a new game *High priority*

- **FR 4.1** The user should get a clear visual feedback that the game is being started, and that the system is looking for another player. This can be done with e.g. a text label telling the user that the system is looking for an opponent
- **FR 4.2** The user should be notified when another player also is ready to start a new game
- **FR 4.3** When two players are ready to play, the system wil generate two boards with ships on them, see item **FR 5**
- **FR 4.4** When both players are ready, the game is considered "started"

FR 5 The system should generate a grid for each player, populated with ships. The grids should have the same amout of squares filled. *High priority*

FR 6 The user should see his own grid with ship placements, as well as the opponent's grid without ship placements (although with succesful hits shown) *Medium priority*

FR 7 The user should be able to make a move in a started game *High priority*

- **FR 7.1** The user selects a cell (coordinates) by pressing it on the screen
- **FR 7.2** The user completes the move by pressing a “Fire”-button.
- **FR 7.3** The system gives the user visual feedback on the move.
  - **FR 7.3.1** If there was a ship placed on the coordinates the user gave, it is considered a hit.
  - **FR 7.3.2** If there are no ships on the coordinates the user gave, it is considered a miss.

FR 8 The user should be able to make a move when the second player has made a move. *High priority*

- **FR 8.1** The user is notified that the other player has made a move.
- **FR 8.2** The user can see the move made by the other player
- **FR 8.3** The user can make their own move, see item **FR 7**

FR 9 The user should be able to press a button to flick between his own grid, and the oppnonents grid. See item **FR 1** for the grid requirements.

- **FR 9.1** The opponent grid should only show the moves he or she has done, and if the move was a hit or a miss. See item **FR 7** for requirements regarding moves.

FR 10 The user should be presented with information about how the game works, before he can start a new game. *Low priority.*

FR 11 The user should be able to play a local game against the computer. *Low priority.*

3 Quality Requirements

3.1 Introduction

In this section we’ll define the quality requirements for the system: Which one will be our primary focus, and which one will be our secondary focus? An ideal world, all QA attributes would be heavily weighted in the development of a computer system, but as one has limited resources, one also has to limit oneself.

But what is a quality attribute? It is a measurable or testable property of a system [**Side 63**, 2]. For instance a system can have high usability. One must have clear understanding of how the system can meet a requirement of high usability, before the system is developed.

3.2 Quality requirement scenarios

In this section we go through each of the quality requirments, starting with our primary and secondary quality requirement attributes.

Modifiability

Scenario ID	MOD-1a
Source	User, course staff or developer
Stimulus	Wishes a change in text on buttons, or placement of them in menus
Environment	Evolution phase
Artifact	Code, changed documentation.
Response	Change made and application tested
Response measure	Within an hour

Scenario ID	MOD-1b
Source	User, course staff or developer
Stimulus	Wishes a change of background image or logo or graphic for a ship.
Environment	Evolution phase
Artifact	Code, changed documentation.
Response	Change made and application tested
Response measure	Within an hour

Scenario ID	MOD-2
Source	Developer
Stimulus	Whishes to have graphics for ships, instead of an S indicating that a cell has a ship
Environment	Evolution
Artifact	Code on server and client application, changed APIs and documentaiton.
Response	Change made to client and server application, system and unit tests executed.
Response measure	Within two weeks

Scenario ID	MOD-3
Source	Developer, user or course staff
Stimulus	Wants to be able to have more than one game going at a time, and have an overview ongoing games in the client application.
Environment	Evolution phase
Artifact	Code on the client application, changed documentation.
Response	Feature implemented, system and units tested
Response measure	Within three weeks

Scenario ID	MOD-4
Source	Developer, user or course staff
Stimulus	Wants to be able to communicate with opponent by text (chat)
Environment	Evolution phase
Artifact	Code on the client application, code on server application, new API for chat, changed documentation.
Response	Feature implemented, system and units tested
Response measure	Within four weeks

Scenario ID	MOD-5
Source	Developer, user or course staff
Stimulus	A wish for an indicator that shows how many ships are left on the opponent board
Environment	Evolution phase
Artifact	Code on the client application, code on server application, changed documentation.
Response	Feature implemented, system and units tested
Response measure	Within three days

Scenario ID	MOD-6
Source	Developer, user or course staff
Stimulus	A wish for being able to place own ships
Environment	Evolution phase
Artifact	Code on the client application, code on server application, changed API, changed documentation.
Response	Feature implemented, system and units tested
Response measure	Within two months

Availability

Scenario ID	AVB-1
Source	User
Stimulus	Unresponsive server
Environment	Normal operation
Artifact	Process
Response	Notify operations team. Continue to operate.
Response measure	Maximum five minutes of downtime, given that

Security

This QA attribute is not a priority for us, but we have some requirements for the number of players in the game.

Scenario ID	SEC-1
Source	User
Stimulus	Tries to break into an excisting game
Environment	Normal operation
Artifact	Process
Response	Join a new game, not throwing any players out of existing games
Response measure	Not throw players out of excisting games 98% of the time.

Performance

Scenario ID	PERF-1
Source	User
Stimulus	Pressing the screen in the mobile application
Environment	Normal operation
Artifact	Mobile application
Response	Visual feedback that the app registred what you did
Response measure	Feedback within 2 seconds

Scenario ID	PERF-2
Source	User
Stimulus	Making a move in the game
Environment	Normal operation
Artifact	Process
Response	Give move to other player
Response measure	Maximum 1 second from move is made and sent by player 1, to player 2 is informed about the move.

Usability

Scenario ID	USR-1
Source	User
Stimulus	Opens the application
Environment	Normal operation
Artifact	Information
Response	An information screen on how to operate the application
Response measure	The user should be able to understand and operate all the necessary functions to play the game within 5 minutes.

4 COTS - Components And Technical Constraints

We will develop for the Android platform. It is an open-source operating system for mobile devices. Apps are written primarily in Java.

We have chosen libGDX [3], an open-source and cross-platform game development framework written in Java to power our multiplayer game. The advantage of this approach is that we can run and debug our game natively on our computers instead of emulating/running on an Android device. We also found that it has an extensive wiki-page[4], and that the project is still under development, which means it will be supported for a while. We need to make a lot of HTTP requests, and do JSON reading and building on the client LibGDX has APIs for both of these things, and they are as mentioned, well documented on their wiki. [7]. [8]

libGDX has APIs for setting up multiple views (called Screen) that can be handled from a main game class (Game). The different APIs are also described in their wiki. [9][5]

While we could have targeted multiple other platforms as well (HTML/WebGL, Windows Phone, iOS), we will only test our game on Android.

4.1 Screen resolutions

The latest mobile devices have very large resolutions. In the latest version of the Android SDK, the emulated device is Nexus 5 with a resolution of 1920x1080p. We will only support and test for the resolutions 1920x1080p (Nexus 5 etc.) and 1280x800p (original Nexus 7 etc.), but the game will most likely work on other devices as well. Other than this we can make use of all the different APIs of the libGDX library, described on their wiki. [5]

4.2 Input

The game will use touch as the primary input method, and all the buttons and actions must work well with the resolutions specified above. For testing purposes, a mouse will suffice.

The libGDX has extensive APIs for handling all necessary inputs, including touch and text. This is described in their wiki. [6]

4.3 Server

The server will provide a REST API written in Javascript provided by Express.js, a web application framework running on the Node.js open-source run-time environment. The data exchange format will be JSON, and this must be handled by the client and reflected in the architecture. The server will be running on the cloud platform as a service (PaaS) Heroku[1], which let's us deploy the server application easily with one command. The database management system will be PostgreSQL[10], as it is supported by Heroku, and as the team has some experience with it. This is the only requirement given by the Heroku platform. The choice to use NodeJS was done in advance, and this works well with the platform.

5 Issues

One issue we faced writing this report was regarding the qualty requirement scenarios. We found that exactly what was expected in this regard was not clear, and that the example report this not give a good lead on how to go through with it.

To meet this issue we contacted the course staff through the forum Piazza which was opened for the course.

6 Changes

In table 6 you can see what changes we've carried out with this document from first draft to the final delivery.

Date	**Change**
02.03.2015	Delivery of the first draft of the report.
20.03.2015	Reworked quality requirement scenarios and prioritsed requirmenet due to feedback from courses staff. Went through the functional requirements to ensure that they did not conflict with each other, so that we would not encounter conflicting requirements in development. Also updated COTS so it refers to the APIs we plan to use.
22.04.2015	Changes due to feedback from course staff: Added a picture to the introduction and tried to better explain how Battleship works. Made all subrequirements have correct labels. Delivery of final draft.

Table 2: Changes made to this document.

References

[1] *Heroku – Cloud platform as a service (PaaS)*. Apr. 22, 2015. URL: http://heroku.com/.

[2] R. Kazman L. Bass P. Clements. *Software Architecture in Practice*. Addison Wesley, 2013.

[3] *libGDX – Java game framework*. Feb. 23, 2015. URL: http://libgdx.badlogicgames.com/.

[4] *libGDX Wiki*. Mar. 20, 2015. URL: https://github.com/libgdx/libgdx/wiki.

[5] *libGDX Wiki about graphics*. Mar. 20, 2015. URL: https://github.com/libgdx/libgdx/wiki/Graphics.

[6] *libGDX Wiki about Input-handling*. Mar. 20, 2015. URL: https://github.com/libgdx/libgdx/wiki/Input-handling.

[7] *libGDX Wiki about JSON-reading and -writing*. Mar. 20, 2015. URL: https://github.com/libgdx/libgdx/wiki/Reading-\%26-writing-JSON.

[8] *libGDX Wiki about Networking*. Mar. 20, 2015. URL: https://github.com/libgdx/libgdx/wiki/Networking.

[9] *libGDX Wiki about the Game class*. Mar. 20, 2015. URL: https://github.com/libgdx/libgdx/wiki/Extending-the-simple-game.

[10] *PostgreSQL*. Apr. 22, 2015. URL: http://www.postgresql.org.