NTNU

Grumpy Old Git Club
Team Number: 16

TDT4240 Software Architecture

Bomb Hunt
Requirements
Android

*Cabral Cruz, Samuel (*496704*)*
*Claessens, Bart (*486346*)*
*Ihlen, Erling Hærnes (*765137*)*
*Trollebø, Jarle (*766901*)*

Primary Quality Attribute:
Modifiability
Seconday Quality Attribute(s):
Usability, Performance and Interoperability

Presented to
Alf Inge Wang

Department of Computer and Information Science
Norwegian University of Science and Technology
(NTNU), Trondheim
Spring 2018

Contents

List of Figures

List of Tables

Introduction

Requirement Phase

The first phase of this project is the requirements phase. The main objective of this phase is the creation of specification document which will serves as a guideline for the following phases. Different aspects of the project will be analyzed during this phase such as the goals, the scope and the market positioning of the desired product. A list of functional requirements and Quality Attributes (QAs) that should be fulfilled by the application will be established in order to precisely define the agreement between the different stakeholders. Moreover, a list of constraint and issues will be dressed so that their impact will be foreseen and the risk to face them minimized. Once all these information will have been gathered, it will be possible to identify the Architecturally Significant Requirements (ASRs) specific to the application. [Bass, 2013, Chapter 16]

Requirements Document

In this report, we first describe the goals and scope of the project followed by the game concept and the expected market positioning of the final product (Vision Statement). Next, a list of the functional requirements and the quality requirements will be presented in the chapters Functional Requirements and Quality Requirements respectively. The chapter COTS Components and Technical Constraints will discuss about the choices of the Commercial off-the-shell (COTS) components and their impact on the software architecture. The main issues and concerns of the project will also be available in the chapter Issues. Since the following document will be updated with the advancement of the project, a log of changes will be maintained (Changes). Take note that a glossary describing the technical expressions can be found at the end of the report.

Techniques and notations used in this document follow recommendations of Bass [2013] and Larman [2005].

Vision Statement

Project Goals

The goal of this project is to develop a multiplayer game called *Bomb Hunt*. In this game, the users will incarnate a gladiator trying to make its way into a labyrinth to find and defeat its opponents to accumulate points. In addition to the multi player mode, the player will also have the opportunity to play the game on a single player mode where the goal will be to survive as long as possible by resisting to different waves of ennemies spawning. A tutorial mode will also be developed to make the introduction of the game and its different components much easier to the inexperienced users. A more in-depth description of the game concept is available in the section Game Concept.

Project Scope

The *Bomb Hunt* will be developed in the context of the group project for the course TDT4240 Software Architecture. The project will be lead by a team of 4 developers that will work part-time for the next 3 months. The best case scenario would have been to find an artist to optimize the aesthetic part of the application.

The application should minimally works any Android platforms that have the necessary utilities to download and execute an Android Package (APK). Even if this project is mainly dedicated to an educational purpose and bounded to a learning context, the team members would like to officially publish the game and maintain it after the end of the semester.

Game Concept

As mentioned before, the concept of the game we are planning to develop is inspired from the vintage video game series called *Bomberman* first developed in 1983 by Hudson Soft. In those game, the user usually incarnate a robot that try to find its way out of the maze by destroying walls and find the key that will open the door to the next level. Multi player versions of this game has been developed since 1990, but for long time remained a side feature of the original single player game. Wikipedia contributors [2018]

Even if this game has been commercialized under new official console versions for more than 20 years, we still think that this concept can be pushed forward by the addition of new

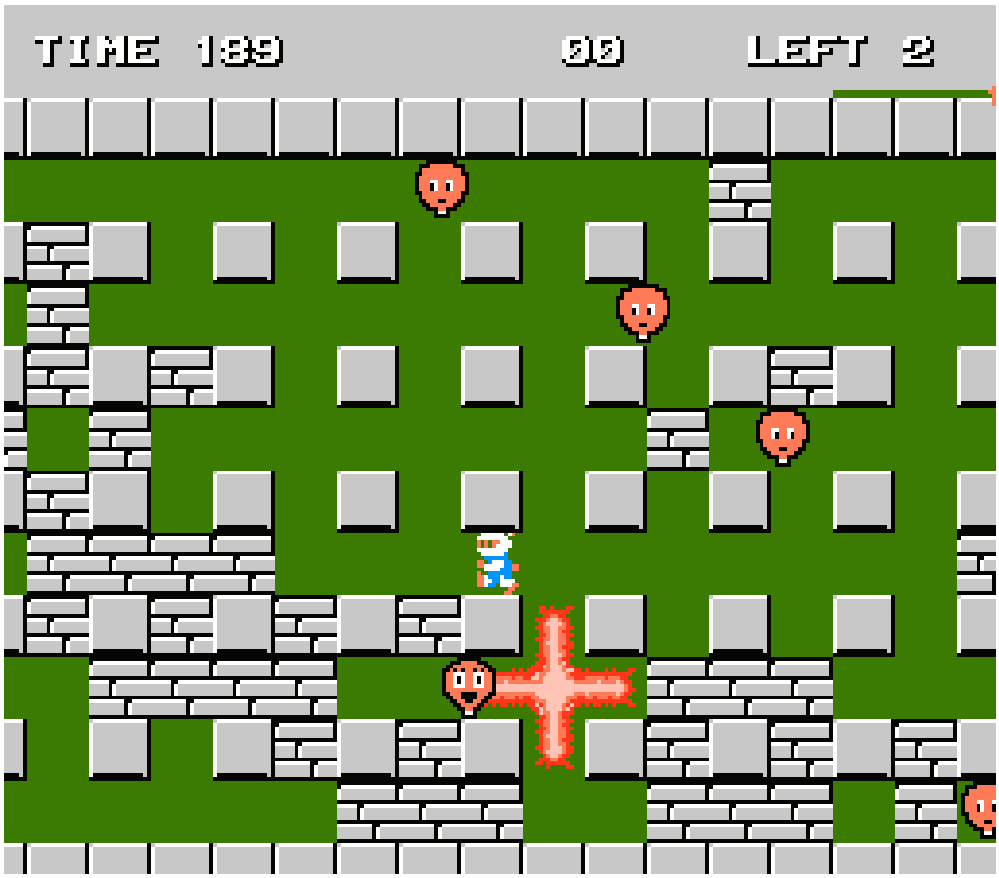

Figure 1: Gameplay of the original Bomberman on NES (1983)

characters, items, maps, abilities, etc. and by creating a mobile version out of it. Despite this long success story, *Bomberman* series did not see any new version released from 2010 to 2016. Its latest version at the moment is called *Super Bomberman R* and is focusing on the mutliplayer gameplay on a really similar way that we are about to do.

In our versions of the game, all the basic elements of the game will be conserved.

- 2D maze where the character can only move up, down, right or left
- Mix of destructible and indestructible walls
- Items dropping off the destroyed walls

The new elements added will mostly be related to the special abilities of the each character. Some examples of those abilities are defusing bombs, building walls, summoning Non-Player Characters (NPCs) and kicking bombs. Great care will be needed to balance the cool down of those special abilities to avoid some characters to be over-powered comparatively to the others. Moreover, we are planning to use bigger maps and smaller camera view that will allow us to use more detailed graphics and to add additional difficulty to the game since a

given user will not necessarily know what is happening on the rest of the map.

We are expecting the gameplay to be fast and to be flexible enough to account for many strategies. Users will be rushed to find the opponents to avoid that they build too much in force. For this part, we are planing to build a skill upgrading system that will introduce some variations depending on the character. Those variations will create some characters that will be stronger in early fights and some others that will only reach their full capacities around the end of the match. Also, the point system will be created in such a way that the only real way to make points will be to tear down the other users whereas breaking walls will only account for small amount of points. These dynamics will necessarily push the opponents to engage the fight faster.

Another interesting aspect of the game will be the accumulation of badges according to the way the users are playing the game. Based on the statistics, the different badges will be unlocked. Down here is a short list of achievements that is currently considered to be included in the game.

Collateral Dommage
Kill 3 adversaries at the same time

Kamikaze
Eliminate everybody on the map (including you) and win the match

Excavator
Destroy 1000 walls

Veteran
Play 100 Matches

Furthermore, all the users of the application will be ranked according to their overall statistics. This overall score will have to take several parameters into consideration to favor active users and distinguish skilled users at the same time.

Due to the simplicity of the gameplay, some efforts will be necessary in order to attract modern users by using smooth graphics and intuitive User Interface (UI). Some mobile versions of this game have already been created. The Figure 2 show one of these that we found particularly nice in terms of the graphical interface. We expect ending up with something that looks somewhat similar to this.

Figure 2: Bomberman mobile in action by Fejer

Market Positioning

Business Opportunities

Nowadays, mobile applications are not seen as something really extraordinary anymore. Their easy accessibility by the end-users joint to the rapidity of their development and deployment have lead to the creation of countless number of them, especially games. It is harder and harder to find new game concept that will be completely different as something that already exists somewhere else. Nevertheless, some games are still able to stand out themselves in this ocean of pastime.

Most of these outstanding games have somewhat the following set of features:

- Easy to learn and use
- Nice graphics and animations
- Can be played for 1 minute or hours
- Have some real challenges involved

- Constantly requests the attention of the user (fast gameplay)

With our concept, we think that most of these elements have been reunited, but having those is not necessarily a direct way to success and the only way to know it will be when the game will have been published.

On the other hand, creating mobile games is still considered as an open market comparatively to console games development where only few giant companies have sufficient money and staff to try their chance.

Overall, the team does not expect to get money out of this project, but, since they are obliged to develop this game, they will make their best to have a nice final product that **may** lead to some returns through advertisements.

Product Position Declaration

Bomb Hunt is developed for a very diverse clientele of any age and culture wishing to have some good moments during their spare times independently of their location. The main aspect of the game that we think will make it different from the other games available or variants of the *Bomberman* concept is the real-time interaction between the different users. Instead of trying to create any kind of sophisticated Artificial Intelligence (AI), users will have to compete against each others. Another trilling aspect is the tracking of the user statistics that will allow them to unlock achievements badges and the ranking of the users. We will also develop the game in such a way that the addition of new features such as characters, items and maps will be easy to perform. This modifiability will allow us to keep the active users interested in the game and to have new things to exploit and discover along their play time. We must however keep in mind that the vast majority of our customers will probably be aged from 6 to 18 years old. Hence, the animations should not be to crude while at the same time remains funny. Finally, the game will provide a platform allowing the users to defeat their friends in a bombing skirmish, thus main purpose of this application is recreative.

Functional Requirements

In this chapter, an exhaustive list of all the functional requirements will be established. For each requirement, an unique ID, a description and a prioritization measure will be identified. The priority measure will be based on the Must Have, Should Have, Could Have, Won't have (MoSCoW) technique Brennan [2009] [1]. The unique ID will only be used for easy referencing and task splitting accros the development team.

First things first, the Figure 3 presents the use case diagram that provides the main functionalities that the game should have. This diagram is at the center of the different functional requirements that will be enumerated below. From the figure, we can identify 5 categories of use cases [2]:

- Deployment (D)
- Gameplay (P)
- Statistics Tracking (T)
- Game Setting (S)
- Licensing and Attribution (L)

Those categories will be used to raffines and group together related functional requirements.

UID	Description	Priority
FR.D.1.1	User should be able to install the application	M
FR.D.1.2	User should be able to uninstall the application	M
FR.D.2.1	Installation should be performed via `Google Play Store`	M
FR.D.2.2	Uninstallation should be available via `Google Play Store`	M
FR.D.3.1	Installation of the application should not require the intervention of the user	M
FR.D.3.2	Uninstallation of the application should not require the intervention of the user	M

Table 1: Functional requirements - Deployment

[1] Nomenclature used: M - Must Have, S - Should Have, C - Could Have, and W - Won't have
[2] Those categories are not explicitly illustrated on the Figure 3.

UID	Description	Priority
FR.P.1.1	The user should be able to control game with touch gestures	M
FR.P.1.2	The user should be able to control its character with virtual joystick and virtual buttons	M
FR.P.2.1	The user should have an in-game menu	S
FR.P.2.2	The user should be able to quit to main menu while in-game	S
FR.P.2.3	The user should be able to quit the game from the main menu	C
FR.P.2.4	The game should not pause when the in-game menu is chosen while playing in multiplayer mode	H
FR.P.2.5	The game should pause when the in-game menu is chosen while playing in single player mode	S
FR.P.3.1	The user should be able to choose different game modes from the menu	S
FR.P.3.2	The user should be able to select different characters	S
FR.P.3.3	The user should be able to select different map	C
FR.P.4.1	The user should be able to customize the appearance of the characters	W
FR.P.4.2	The user should be able to create custom maps	W
FR.P.5.1	The user should be able to chat with its opponents	W
FR.P.6.1	The user should be able to play the game in multiplayer mode	M
FR.P.6.2	The user should be able to play the game in multiplayer mode	S
FR.P.7.1	If the player gets hurt by another player or enemy the player should lose health	M
FR.P.7.2	If destroyable map elements get hurt by the player, the element should accumulate damages	M
FR.P.7.3	When walls get destroyed, different powerups should randomly spawn	S
FR.P.7.4	If the player collides with a powerup, the player should get that powerup	H
FR.P.8.1	The user should be correctly assigned to a match when waiting in the lobby	H
FR.P.8.2	The user should not wait to long before getting assigned to a match	H
FR.P.8.3	The user should be able to invite its friends	S
FR.P.9.1	The user should be able to ask for a rematch at the end of a match	S
FR.P.9.2	If a rematch is asked by the players, a single match should be created for all the players having voted for the rematch	S
FR.P.9.3	When a match as found all the players, a countdown should be initiated before the launch of the match	S
FR.P.9.4	Two incomplete matches should be combined to create a single and complete one	S

Table 2: Functional requirements - Gameplay

UID	Description	Priority
FR.T.1.1	The user should be to link its Google Play account	S
FR.T.2.1	The user should be able to consult its ranking on a leaderboard	H
FR.T.2.2	The leaderboard should show a view specific to the user friends	C
FR.T.3.1	The statistics should be accumulated and stored in a dedicated database	H
FR.T.4.1	The statistics should enable the user to unlock achievements	H
FR.T.4.2	The user should be able to know what he has to do in order to unlock the achievements	S
FR.T.4.3	The user should be able to consult the achievements he has earned so far	H
FR.T.4.4	The accumulation of achievements should have an impact on the leaderboard score	S
FR.T.5.1	The user should be aware of how to enhance its score	C
FR.T.5.2	The score should favored active users	S
FR.T.5.3	The score should be able to make the difference between skilled and unskilled users	H

Table 3: Functional requirements - Statistics Tracking

UID	Description	Priority
FR.S.1.1	The user should be able to turn off sound FX	H
FR.S.1.2	The user should be able to turn off music	H
FR.S.2.1	The settings should be saved in configuration files in order to avoid the user to constantly modify them	H

Table 4: Functional requirements - Game Setting

UID	Description	Priority
FR.L.1.1	The user should be aware of the license	M
FR.L.2.1	The user should be able to consult the attributions of the external features	M
FR.L.3.1	The user should have access to these credits from the main menu	M

Table 5: Functional requirements - Licensing and Attribution

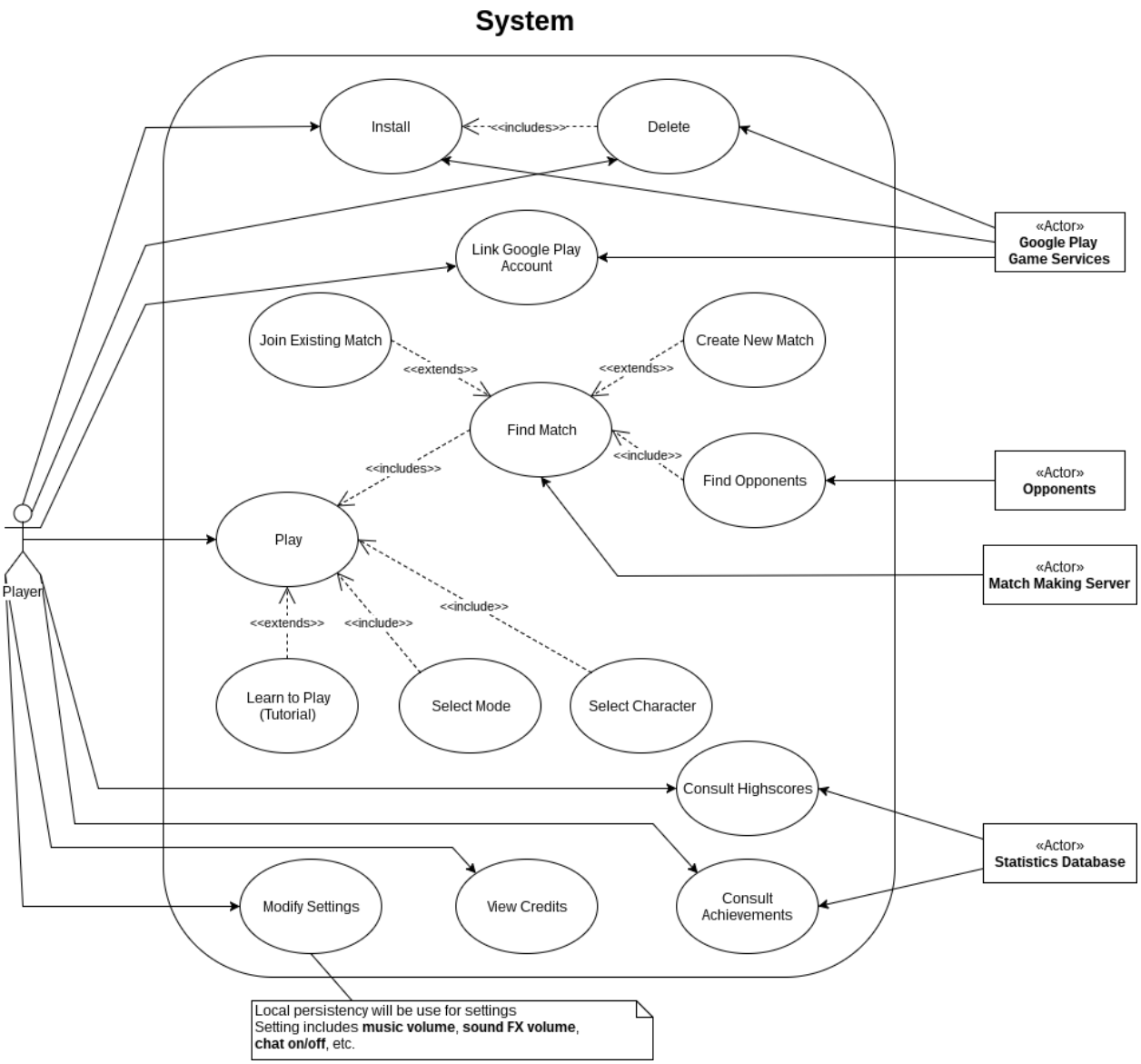

Figure 3: Use Case Diagram of the *Bomb Hunt* game

Quality Requirements

Quality attributes usually have an important impact on the overall architecture because their realization often imply the modification of many part of the application whereas functional requirements are usually more located. Moreover, quality attributes are often more difficult to assert due to their subjective meaning. Per example, some people may have different conceptions of what is defined as a secure application. The purpose of this chapter is to determine some thresholds that will uncover this subjectivity.

To specify the different quality attributes, a scenario approach will be used as the one presented in [Bass, 2013, Chapter 4, p.68-69]. According to this method, a quality attribute can be made unambiguous and testable by specifying the following 6 characteristics:

- Stimulus
- Stimulus source
- Response
- Response measure
- Environment
- Artifact

Modifiability

ID	**M1**
Source	Developer
Stimulus	Wishes to modify existing graphics
Artifacts	Graphical assets
Environment	Design Time
Response	Graphic updated
Response measure	The graphics are updated and seamlessly integrated into the game in less than half an hour.

Table 6: Quality Requirement Scenario - Modifying existing graphics

ID	M2
Source	Developer
Stimulus	Wishes to modify existing audio components
Artifacts	Audio assets
Environment	Design Time
Response	Audio components updated
Response measure	The audio components is updated and seamlessly integrated into the game in less than half an hour.

Table 7: Quality Requirement Scenario - Modifying existing audio components

ID	M3
Source	Developer
Stimulus	Wishes to modify the logic of an existing component
Artifacts	Code
Environment	Design Time
Response	Changes performed, functional tests passed and integration tests passed
Response measure	Change made in less than half an hour without affecting the component's interface or other parts of the software.

Table 8: Quality Requirement Scenario - Modifying logic of an existing component

ID	M4
Source	Developer
Stimulus	Wishes to add a new powerup item
Artifacts	Code
Environment	Design Time
Response	Addition performed, functional tests passed and integration tests passed
Response measure	Change made in less than two hours without affecting the item's interface or other parts of the software.

Table 9: Quality Requirement Scenario - Adding new powerup item

ID	M5
Source	Developer
Stimulus	Wishes to add a new character
Artifacts	Code and assets
Environment	Design Time
Response	Addition performed, functional tests passed and integration tests passed. The skills of the new character have to be balanced to match the average capacities of the existing characters.
Response measure	Change made in less than three hours without affecting the character's interface or other parts of the software.

Table 10: Quality Requirement Scenario - Adding new character

ID	M6
Source	Developer
Stimulus	Wishes to add a new map
Artifacts	Code and assets
Environment	Design Time
Response	Addition performed, functional tests passed and integration tests passed.
Response measure	Change made in less than seven hours without affecting the current map creation process. The time account for the creation of the map.

Table 11: Quality Requirement Scenario - Adding new map

ID	M7
Source	Developer
Stimulus	Wishes to add a new achievement
Artifacts	Code and assets
Environment	Design Time
Response	Addition performed, functional tests passed and integration tests passed. The new achievement attribution process receive all the needed information from the database connection.
Response measure	Change made in less than three hours without affecting the achievement's interface or other parts of the software. This consider that all the information need was already available for the attribution.

Table 12: Quality Requirement Scenario - Adding new achievement

ID	**M8**
Source	Developer
Stimulus	Wishes to modify the scoring algorithm
Artifacts	Code
Environment	Design Time
Response	Addition performed, functional tests passed and integration tests passed. Users reranked and leaderboard updated.
Response measure	Change made in less than five hours without affecting the structure of the database or other parts of the software. The time ignore the time needed to reprocess the user ranking.

Table 13: Quality Requirement Scenario - Modifying score calculation algorithm

ID	**M9**
Source	Developer
Stimulus	Wishes to modify the database structure
Artifacts	Code and databases
Environment	Design Time
Response	Addition performed, functional tests passed and integration tests passed. All the requests made to the database processed correctly.
Response measure	Change made in less than seven hours without affecting the parts of the software that depends on the database.

Table 14: Quality Requirement Scenario - Modifying database structure

Usability

ID	U1
Source	User
Stimulus	Wants to play for the first time
Artifacts	System
Environment	Runtime
Response	User is introduced to controls and mechanics within the game through a short and simple textual tutorial.
Response measure	Finished the tutorial within 3 minutes and has a basic understanding of the game components and its controls.

Table 15: Quality Requirement Scenario - First-time play

ID	U2
Source	User
Stimulus	Wants to join an online multiplayer match for the first time
Artifacts	System
Environment	Runtime
Response	User enters in communication with the server, reaches the lobby and gets assigned to a match.
Response measure	The user does not have to modify any internet communication settings to get a connection to the server as long as a valid internet connection is available. The user do not remain unassigned for an excessive amount of time (less than 2 minutes). In case of failure to join a match, the user receives a message mentioning the reason why no match as been found. User should be able to start playing within 3 minutes on average.

Table 16: Quality Requirement Scenario - Introduction to multiplayer mode

ID	U3
Source	User
Stimulus	Wants to modify a setting of the game
Artifacts	System
Environment	Runtime
Response	User finds the setting and successfully modifies it. New settings are saved for later next booting of the application.
Response measure	Setting is found without any navigation-errors in 15 seconds or less.

Table 17: Quality Requirement Scenario - Modify game through settings

Performance

ID	P1
Source	Player-client
Stimulus	Server sends data of other players
Artifacts	Player-client
Environment	Normal Operation
Response	Data is used to update the relevant player locally.
Response measure	Latency below 30 ms for the update of local state.

Table 18: Quality Requirement Scenario - Client receives data from server

ID	P2
Source	Player-client
Stimulus	Periodic update
Artifacts	Player-client
Environment	Normal Operation
Response	All local data is used to construct the update message.
Response measure	Throughput of 30 successful updates per second for steady frame rate.

Table 19: Quality Requirement Scenario - Periodic client update

Interoperability

ID	I1
Source	Player-client
Stimulus	A request to set up a new multiplayer room
Artifacts	Player-client and the Google Play services servers
Environment	Runtime
Response	The request is accepted and a room specified by parameters is set up
Response measure	100% of the sent requests that are in the correct format and order are accepted

Table 20: Quality Requirement Scenario - Player wants to initiate a multiplayer room

COTS Components and Technical Constraints

In this chapter, we will describe the constraints imposed by our choice of COTS. Among the different COTS that will be used for the development of this application, we can find `LibGDX`, `Artemis-odb`, `Android`, and `Google Play Game Services`. For each of these, the rationale behind their selection, the specific interface that we have to implement and an analysis of the constraints they add to the project will be provided.

LibGDX

During early assignments in this course, we got to know the way `LibGDX` [Zechner] can have a stack of screens, and only render and display the top one. This will constrain how we develop the way we change screens, and how we should handle dumping all entities once we drop a screen. This is important to get right as the performance might vary if this is done poorly.

Artemis-odb

Since we decided to develop the core part of our game using Entity-Component-System (ECS) we looked at different frameworks that already implements such system. First we looked at `Ashley` [ash] which is a sub-library of LibGDX, it is a very lightweight library that only contains the most basic aspects of ecs. Secondly we had a look at a more preferment [3] `Artemis-odb` [Papari]. This library is slightly more heavy weight than Ashley is, but in the end a lot easier to work with. It doesn't only implement the basics of ecs, but also implements a huge number of helper classes that for instance simplifies entity creation[4] and modification[5]. ECS requires that the logic is separated from the data using systems and components. Each system performs a set of actions on a subset of all entities fulfilling a set of require components for the action to be executed. Entities are therefor just bags that holds a set of components and nothing more. In practise this means any entity can be completely change behaviour by changing the set of components it has.

[3] https://github.com/junkdog/entity-system-benchmarks
[4] https://github.com/junkdog/artemis-odb/wiki/Archetype
[5] https://github.com/junkdog/artemis-odb/wiki/Entity-Transmuter

Android

One of the mayor challenges for developing a game for Android, is that the version of Android and Java might vary a lot based on what device is playing the game. We imagine we will probably just restrict the game to not work on older devices, to save ourselves some headache with trying to understand why something might not work on older devices. Another thing that will constraint how we make our game, is the difference in resolutions and hardware performance on each device. This means we will have to come up with a way to scale our UI and on-screen elements, instead of just rendering based on a set resolution. This also means we will have to follow strict rules on how to clear unused elements in the game world, so the memory of the device doesn't get filled up.

Google Play Game Services or other solutions

Since we want to have some service that can set up Peer-to-peer (P2P) connections for us, we realize our game architecture needs to include some form of manager or handler that will communicate with this service. Assuming we use some COTS like Google Play Game Services (GPGS) [Google], we will most likely have to follow some very strict rules that allow P2P setups via this service. These sort of interactions will happen when a user sets up a room, when a user want to join that room and when users are restricted to join certain rooms.

Issues

Before starting this project, we had a few ideas of what issues might arise during this project.

Graphics

We have no graphical experience within the group, and as we look at other mobile games, it becomes apparent that a simple graphical look is something we wish to achieve during this project. This is to make the game more appealing to the users.

peer-to-peer connection

Since we rely on peer-to-peer to transfer the data between devices, we might encounter problems with devices that are unstable. How we will we handle bad devices potentially crashing game sessions and ruining the experience for other players.

Changes

This section records the changes brought to the current document since its creation.

Date	Version	Description
2018/02/21	**1.0**	Initial draft

Glossary

ASR Requirement that will have a profound effect on the architecture—that is, the architecture might well be dramatically different in the absence of such a requirement.
[4, p.291].

Cool-Down The minimum length of time that the player needs to wait after using an ability or item before it can be used again.
[2, cool-down].

NPC Taken from the world of pen-and-ink role-playing games, an NPC is a character encountered in an RKJ who is not controlled by the user.
[3, p.38].

Bibliography

ashley wiki. URL https://github.com/libgdx/ashley/wiki. [Online; accessed 25-February-2018].

Dictionary definitions. URL http://www.yourdictionary.com/.

Lexicon A to Z: NPC (Nonplayer Character). *The Next Generation Magazine*, (15), Mar 1996.

Len Bass. *Software architecture in practice*. Addison-Wesley, Upper Saddle River, NJ, 2013. ISBN 978-0-321-81573-6.

Kevin Brennan. *A guide to the Business analysis body of knowledge (BABOK guide*. International Institute of Business Analysis, Toronto, 2009. ISBN 978-0-9811292-1-1.

Kenneth Fejer. Pixel art. URL http://www.kennethfejer.com/pixelart.html.

Google. Google play game services. URL https://developers.google.com/games/services/. [Online; accessed 19-February-2018].

Craig Larman. *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development*. Prentice Hall PTR, Upper Saddle River, N.J, 2005. ISBN 0131489062.

Adrian Papari. artemis-odb wiki. URL https://github.com/junkdog/artemis-odb/wiki. [Online; accessed 23-February-2018].

Wikipedia contributors. Bomberman (1983 video game) — wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=Bomberman_(1983_video_game). [Online; accessed 25-February-2018].

Mario Zechner. Libgdx -goals and features. URL https://libgdx.badlogicgames.com/features.html. [Online; accessed 25-February-2018].