Norwegian University of Science and Technology

IT2901 - Project 2

Don't Panic!

A Serious Electronic Board Game

Authors:
Jens Even Blomsøy - jenseven@stud.ntnu.no
Stian Aurheim - aurheim@stud.ntnu.no
Jørgen Foss Eri - jorgeer@stud.ntnu.no
Sindre Svendsrud - indrsv@stud.ntnu.no
Adrian Arne Skogvold - adriansk@stud.ntnu.no
Jim Frode Hoff - jimfrode@stud.ntnu.no

Customer:
Ines Di Loreto - inesd@idi.ntnu.no

Supervisor:
Mohsen Anvaari - mohsena@idi.ntnu.no

May 6, 2013

Contents

Chapter 1

Introduction

1.1 The course

The main goals in this course are to experience and learn how to work on a development project as a team. In addition, the team has to answer to a customer, as software development companies often do, which stands out from other projects in the past. This is an advanced course and it is expected that knowledge obtained from previous courses is used, especially the development courses such as Informatics Project I and the collaborative System Development project.

The group has an appointed guidance counselor as well as a customer. The counselor will be available for answering questions regarding the project management in general, and push the group to reflect on its decisions and review the work done. Status reports will be delivered regularly, so the counselor can stay up-to-date with the work in the group.

During the course, several project reports are scheduled for delivery; the preliminary project report, the mid-term project report and the final project report. Working on and delivering these reports will help in the planning and development of the project, and feedback will be given from the counselor. The grading of the project will take the final project report into consideration, as well as the final product.

1.2 The team

The team consists of six students of Informatics at NTNU:
Stian Aurheim

- Third year bachelor in Informatics
- Main experience in Java. Some experience in Python, PHP, HTML, JavaScript

Jens Even Berg Blomsøy

- Third year bachelor in Informatics
- Programming languages worked with: Java, Python.
- Main knowledge in System Development, system architecture and system documentation.

Jørgen Foss Eri

- Third year bachelor in Informatics
- Experience with Java, Python, JavaScript/HTML5/CSS3 and general web development

Jim Frode Hoff

- Third year bachelor in Informatics
- Programming languages worked with: Java, Python, PHP, JavaScript and general web development

Adrian Arne Skogvold

- Third year bachelor in Informatics
- Programming languages worked with: Java, C#, Oz, Actionscript

Sindre Svendsrud

- Third year bachelor in Informatics
- Experience with Java, Python and C++

1.3 Problem description

The customer has developed a paper prototype [Figure A.1] of a board game called Don't Panic. The game is designed to help crisis workers make the right decisions during a crisis in a city. It is a turn based, collaborative multiplayer strategy game where the players have to take actions to stop the inhabitants from panicking. After the game is finished, an expert will review the actions with the players to evaluate whether their choices were sound.
Our customer wants an electronic version of the board game. The electronic board game should maintain the social aspect (both physical and verbal) of a regular board game. In addition, a replay function will be added, to make it easy for the expert to review the game with the different players. The physical version of the board game takes a lot of work setting up and maintaining, as it is time consuming to move the pieces and update the panic levels. The electronic version will automate all of this.

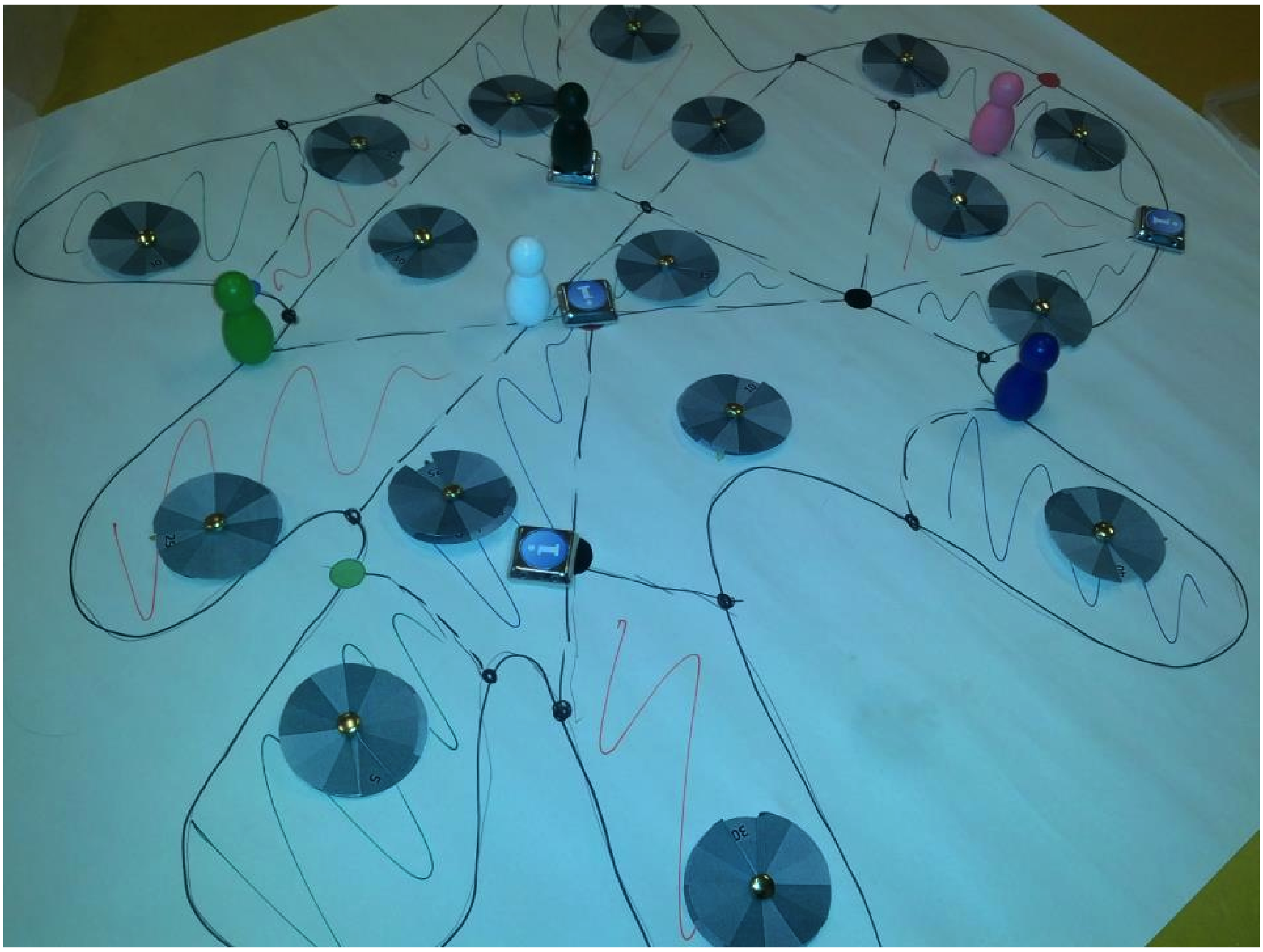


Figure 1.1: Picture of the paper prototype

1.4 Constraints

Only a few of us have some past experience with HTML5 and JavaScript
The game is to be completed within one semester (21 January - 27 May)

1.5 Customer and supervisor

The customer for this project is Ines Di Loreto, a researcher in the Department of Computer & Information Science at NTNU. The supervisor assigned for this project is Mohsen Anvaari, a PhD candidate in the same department.

Chapter 2

Project Management

2.1 Process Model

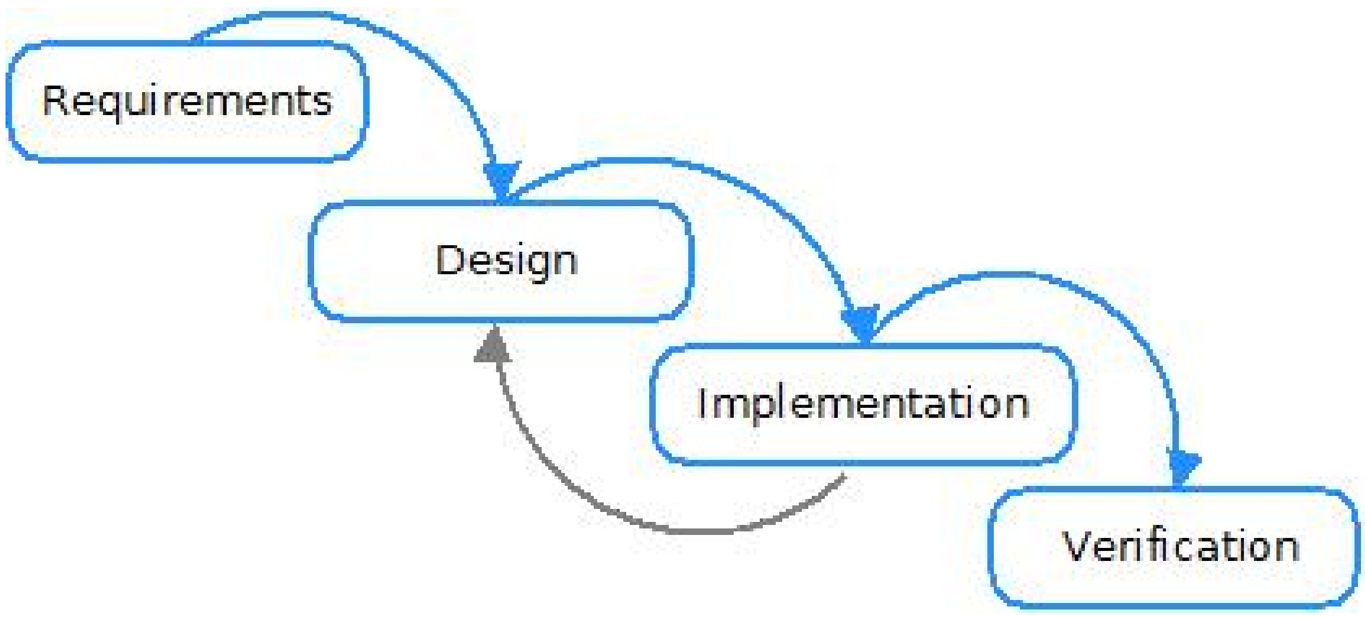
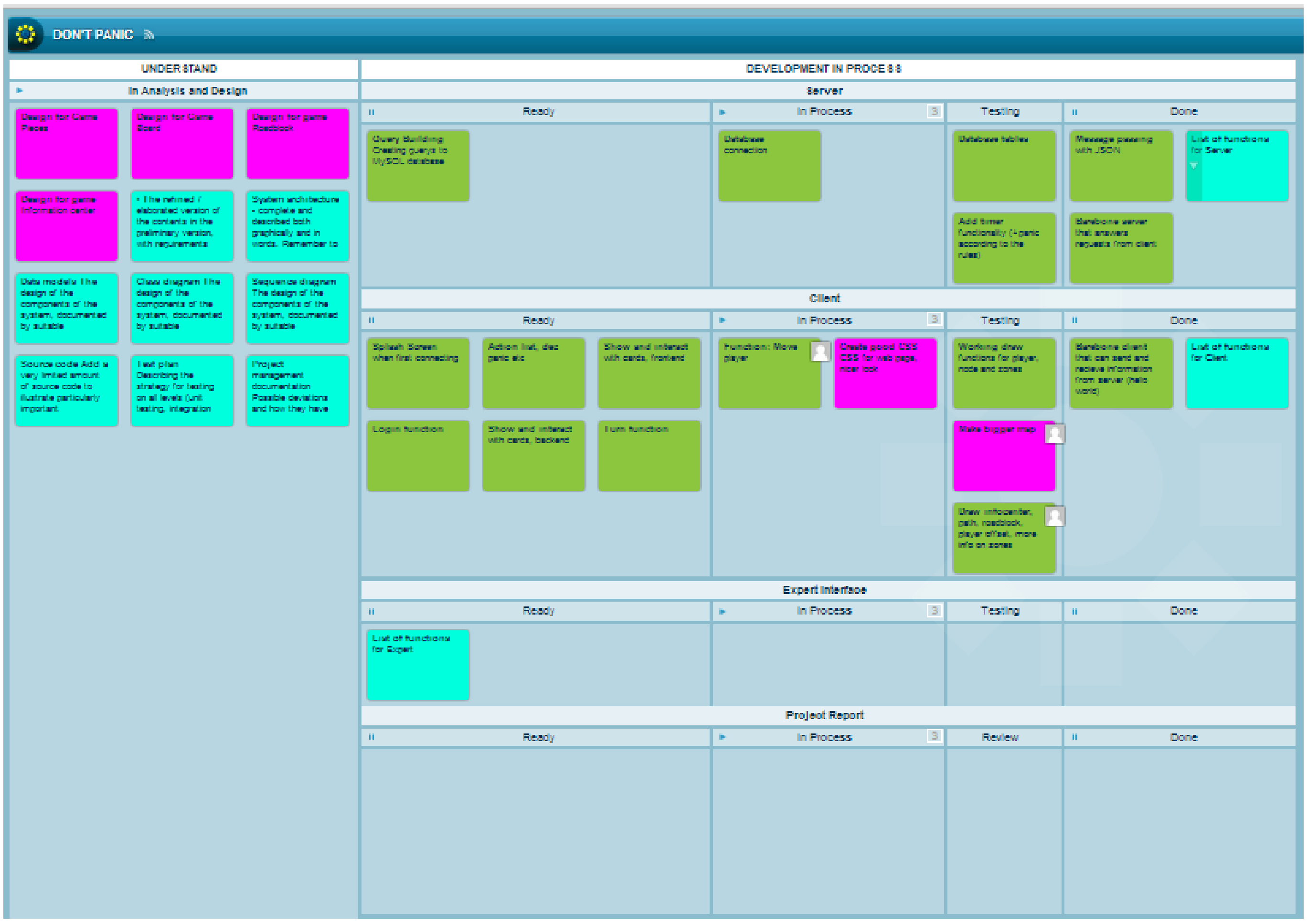


Figure 2.1: Process model, 'Extended waterfall model'

Due to the extensive need for documentation, Scrum is not a suitable model of this process. Moreover, the intended model of process cannot be fitted in a waterfall model. Therefore, the model used will be a mixture of the best properties from both models. First of all the architecture will be designed along with the production of a detailed, sensible documentation. Then a simple, “bare bone” version of the game will be developed. Finally, the required features will be iterated on this version.

2.1.1 Kanban board


DON'T PANIC

UNDER STAND

In Analysis and Design

Design for Game Pieces

Design for Game Board

Design for game Roadblock

Design for game information center

• The refined / elaborated version of the contents in the preliminary version, with requirements

System architecture - complete and described both graphically and in words. Remember to

Data models The design of the components of the system, documented by suitable

Class diagram The design of the components of the system, documented by suitable

Sequence diagram The design of the components of the system, documented by suitable

Source code Add a very limited amount of source code to illustrate particularly important

Test plan Describing the strategy for testing on all levels (unit testing, integration

Project management documentation Possible deviations and how they have

DEVELOPMENT IN PROCESS

Server

Ready	In Process	Testing	Done
Query Building Creating querys to MySQL database	Database connection	Database tables	Message passing with JSON
		Add timer functionality (+panic according to the rules)	Barebone server that answers requests from client
			List of functions for Server

Client

Ready	In Process	Testing	Done
Splash Screen when first connecting	Function: Move player	Working draw functions for player, node and zones	Barebone client that can send and recieve information from server (hello world)
Action list, dec panic etc	Create good CSS CSS for web page, nicer look	Make bigger map	List of functions for Client
Show and interact with cards, frontend		Draw infocenter, path, roadblock, player offset, more info on zones	
Login function			
Show and interact with cards, backend			
Turn function			

Expert Interface

Ready	In Process	Testing	Done
List of functions for Expert			

Project Report

Ready	In Process	Review	Done

Figure 2.2: Project management, 'KanBan board'

The Kanban system fitted the project development process well, as a core prototype of the game was initially developed. Secondly, plans were made to add features such as a replay of a game session, an Expert interface and a database. It was decided to use an online version of a Kanban board called LeanKit Kanban. LeanKit was easy to use and enabled customizing the separation of tasks and processes on the board, as well as categorizing tasks by color. In addition, it enabled customizing the Kanban board, such as adding an extra "Testing" section for tasks.

2.2 Roles and responsibilities (tentative)

These are the roles assigned to the different team members:

- Customer/supervisor contact - Sindre Svendsrud
- Server manager - Jim Frode Hoff
- LaTeX configuration manager - Jim Frode Hoff
- Client manager - Stian Aurheim and Jørgen Foss Eri
- Expert interface manager - Adrian Arne Skogvold
- Team manager - Jørgen Foss Eri
- Test manager- Jens Even Berg Blomsøy
- Database manager - Jens Even Berg Blomsøy
- Documentation manager - Sindre Svendsrud

The LaTeX configuration manager was handed to Jim because he was the only one with prior experience to LaTeX. The rest of the roles were handed to the person who wanted that specific role. This was done because none of the group members had any more experience or knowledge about the roles than any of the others.

Customer/supervisor contact
The customer/supervisor contact is responsible for the communication with the customer and the supervisor. It is important that he shares important information such as time and place for meetings with the rest of the group.

Server manager
The server manager is responsible for the development of the server. That does not mean he has to do all the coding himself, but he has to make sure that everything that needs to be done on the server side is done. The main task is to implement the game rules.

LaTeX configuration manager
The responsibility of the latex configuration manager is to export the project report to latex and configuration of github.

Client manager
The client manager is responsible for the development of the client. That means that he has to make sure that the client is displayed correctly. Typical work here is the drawing of objects such as draw player or draw node.

Expert interface manager
The expert interface manager is responsible for the expert interface. The expert interface sets up the game. The main task is to implement a way for the expert to set up the game.

Team manager
The team manager is responsible for the progress of the project. He makes sure deadlines are met, meetings are attended and that the team members are engaged.

Test manager
The test manager is responsible for writing the tests of the system. He should also make sure that the tests are performed.

Database manager
The database manager is responsible for the database, that means the design of the database (ER diagram) and the implementation of the database. Typical work here is to make database queries.

Documentation manager
The documentation manager is responsible for the documentation of the project. The main tasks are to create and maintain the structure of the report.

2.3 Time plan

A time schedule to view planned tasks and their deadlines has been created in a Gantt chart (available as attachment).

A short summary of the development can be seen as:
Version 1: Server game logic, client event handler, (simple) client view and communication module
Version 2: Addition of the administration interface and the expert client, extend view and game rule support
Version 3: Adaptation to other clients, extra features beyond the core

2.3.1 Milestones

During the project there are certain milestones to be completed in time, both project report and technical milestones.

Project report milestones
10 February - Delivery - Project report: Preliminary version
15 March - Delivery - Project Report: Mid-semester version
19 April - Delivery - Project report: Final comments from supervisor
27 May - Delivery - Project report: Final version

Technical milestones
18 February - Barebone client that can communicate with the server
18 February - Barebone server that can communicate with the client
28 February - Basic prototype of the game
28 February - Database connection
11 March - Game v.1, a simple, working prototype of the game
22 March - Database queries
19 April - Expert interface for setting up the game
10 May - Game final

These are the code milestones which were set for the project in regards to the client/server/database functionality. The reason the milestones is in that order is because many of them depend on the milestones prior to them.

2.4 Work breakdown structure

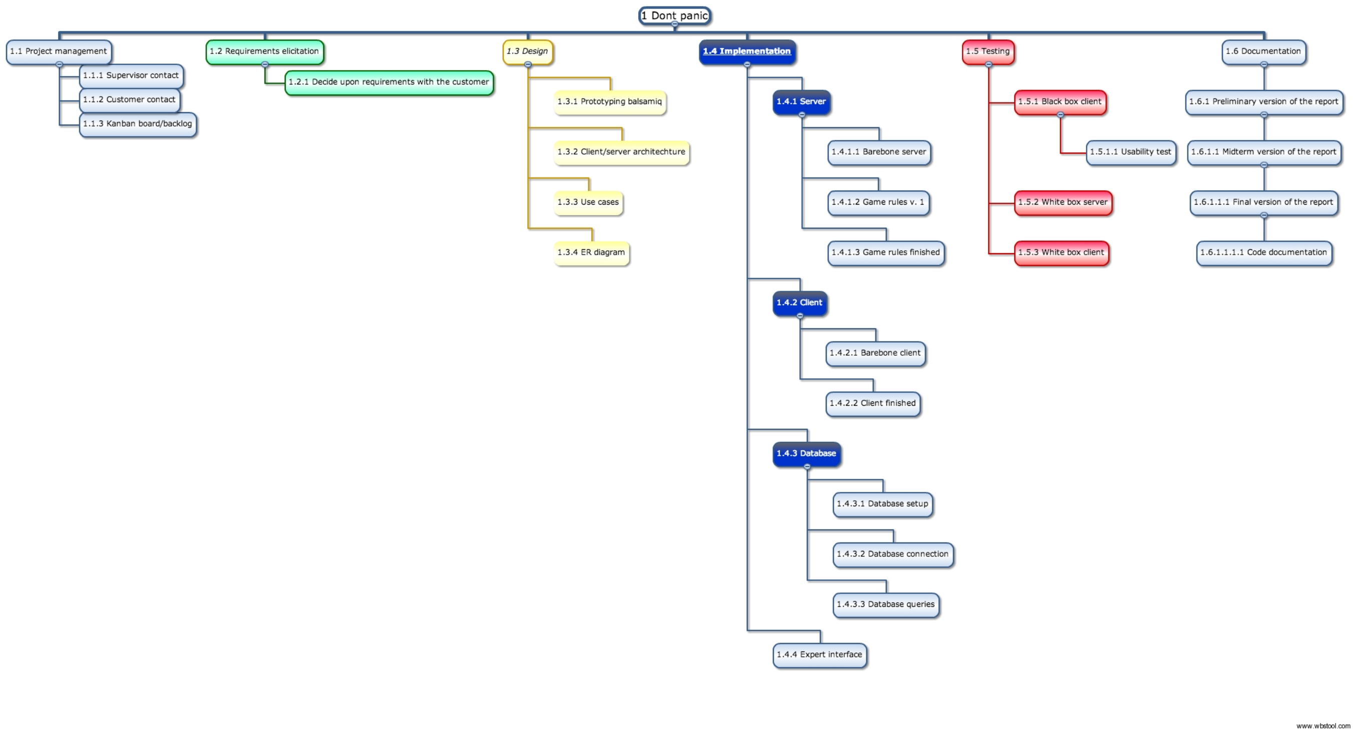


Figure 2.3: Project management, 'Work breakdown structure'

2.5 Risk list

Description	**Likelihood (1-9)**	**Impact (1-9)**	**Importance**	**Preventive action**	**Remedial action**
Data-loss (docs or code)	3	7	21	Continuous saving and version control. Distribute data among all group members.	Try to retrieve data from computer/repository. Start from scratch, if necessary.
Network failure	3	3	9	N/A	Switch to another network. Ask IDI for help, if necessary.
Computer crash(es)	4	7	28	Continuous upload to repository	Try to retrieve data. Worst case scenario: Retrieve most up-to-date data from repository. Use computers from IDI (P15).
Organization/ communication failure	5	7	35	Be watchful of correct distribution of information.	Find out where communication failed and restore the organization.

Personnel absent due to sickness	6	3	18	Continuous upload to repository. Be prepared to pitch in on others' assignment, inform the group that you are unable to meet on the day of sickness. & Get update and data from the person in question. Bring the person up-to-speed on project, work done while he was away.	
Great personal conflicts	3	8	24	Be prepared to withstand discussion and criticism. Tell the other members of the group how you feel. Go to the supervisor/ & professor for advice if necessary.	Resolve the issues with the help of a neutral part (supervisor/professor).
Absent personnel	7	3	21	Point out the importance of attendance to appointed hours. Inform ahead of time if leaving for vacation. See also *Personnel absent due to sickness.*	Point out the importance of attendance again. Notify student assistant if it becomes a problem. See also *Personnel absent due to sickness.*
Loss of personnel	2	8	16	Regular uploads of code and information to repository	Notify supervisor of loss of personnel. Split work assignments to personnel left in group.
Incorrect system requirements	3	8	24	Be sure everyone has read and understood the project requirements. Ask the customer if there any uncertainties.	Resolve the incorrect requirements. Find out what went wrong. Be confident that the other requirements are correct by confirming with the customer.
Inexperienced with development technology	8	3	24	Be prepared to search for information needed. Ask for help from supervisor if necessary. Choose technology we are already comfortable with, if possible.	Search and retrieve needed information. Get acquainted with the technology needed.

Misunderstand project	2	9	18	Make sure everyone has read and understood the project goals, and that our goal(s) fit the customers needs. Regular meetings with the customer for discussion regarding our goals/customer needs.	Notify supervisor of grave errors/ misunderstandings. Work out a rescue plan with the customer.
Misunderstand subproblem	2	8	16	Make sure everyone involved has read and understood the sub project problem/solution.	Correct the mistakes and restart resolvement of subproblem.
Error estimation of time needed	5	5	25	Be prepared for incorrect time estimate, including error margins, avoid bursts.	Do bursts, expand time schedules for work/work longer hours.

Table 2.1: Risk assesment list

2.6 Example of status report

At the end of each week a status report will be sent to the customer and supervisor. This is an example of a status report that follows the given template:

Status report week 10 group 10

1 Introduction
We now have a working prototype!

2 Progress summary
The players can now move and de-panic zones, and turns are implemented. A timer for increased panic is implemented. The drawing functionality for objects works for the most part, and the database connection is now up and running. Css has been implemented for nice looks. Messages passing between the client and server (with json) work correctly.

3 Open / closed problems
The database connection was a huge problem. It was detected that IDI had a firewall turned on. Much time was wasted due to error detection in the code (which was pretty much correct all the time).

4 Planned work for the next period
Make database queries, event and information cards, roles and figure out how to implement effects (could be tricky). Produce documentation as always. Finish the requirements for the midterm report.

5 Updated risks analysis
No updates needed.

2.7 Communication

During the project period it is important to continuously communicate with the customer. This way the customer is always informed on what has been done and enables giving feedback on what is desired next. In addition, the supervisor needs to be informed about the progress/work at all times. Finally, it is crucial that all team members communicate well.

2.7.1 Interactions with the customer

Regular meetings with the customer took place either weekly or bi-weekly. The meetings were scheduled through email. Prior to each meeting a presentation of the current state of the game was prepared. In addition, questions that needed answers regarding game functionality and rules were written. Simple logs of these meetings were kept to enable later reflections on the meetings. The fact that the customer is located at NTNU made it easy to set up schedules and have meetings.

Throughout the meetings, the customer sometimes requested new tasks that went beyond the initial game functionalities. The was done because the customer was satisfied with the work so far, and wanted to provide extra challenges. These challenges were not required for this project, but if possible, they would add extra features to the game. An example of this was the possibility of using Sifteo Cubes as a game client, where the cubes would be used as zones. This was mentioned by the customer during one of the meetings. As this was not an important functionality, it was decided to focus 100% on developing the JavaScript/HTML5 client, and perhaps test the Cubes at a later stage when the client was fully functional.

Date	Called by	Purpose	Preperation	Agenda	Notes
3/11/2013	Customer	Update on where we are with the project	Implemented timer and cards (at least core functionality of cards), a task given to us on prior meeting	Discussed how the amount of people should affect the panic level in the zones. Panic in a zone COULD be proportional to the amount of people, not important. Discussed events and how they could be solved by finishing a "quest" of different steps (example: if there is a fire: move people out, block zone, put out fire). These "quests" were not a requirement, only a suggestion. Discussed the use of Sifteo Cubes (an interactive game system with electronic gadgets) as a client, not a requirement. Important that we have a working game+client; other clients are "bonuses".	This week we will work mostly with the report, not the game. This was explained to the customer.

Table 2.2: Customer meeting log

2.7.2 Interactions with the supervisor

The meetings with the supervisor took place bi-weekly and were scheduled through email. At these meetings the supervisor was informed about the work done since the last meeting. Problems and the group dynamic were also discussed. The supervisor provided feedback on the reports, which was much appreciated.

2.7.3 Interactions within the group

As there were meetings several times a week, much of the communication between the team members was done talking on a daily basis. In addition, meetings were planned and problems discussed through a facebook group. Facebook was chosen over e.g. email because all team members use facebook often, and long discussions through emails were regarded as messy.

Chapter 3

Prestudy

* *denotes the chosen alternative, if there were multiple choices.*

3.1 Environment

The first and probably most important decision to be made in this project was to decide on a language or software to serve as a development environment. There were several good alternatives, but finding one which was suited to both the task and accessible in terms of experience needed from the developers became a minor challenge.

3.1.1 Front end language

Java
The first and most obvious choice for any project on NTNU is Java, as it is the language used in most courses and all the team members are familiar with it. However, there are several issues when choosing Java for the front end. First of all, the ugly nature of the Swing framework is well known (although there are some alternatives). In addition, an applet in Java requires a plugin to work, if it is to be deployed on the web. A desktop version was an option, but that would limit the game to desktops only. Personal preference was also an issue with some of the developers, mostly regarding the Java's static typing and rather verbose syntax.

Unity
A program like Unity would lessen the amount of code on the client side, as creating a user interface and board for the game would be almost drag-and-drop, with some scripting to handle mouse interaction and data transfer with the server. However, in a boardgame that does not require more than 2D graphics, Unity would be overkill. It also requires a plug-in to work.

XNA
This is a game development framework in C# that runs on the xbox360 and windows platforms. Given that none of the team members were familiar with C#, and the restrictiveness in terms of portability, this was not a good option.

HTML5 *
The choice eventually fell on HTML5 as the best front end technology. With the proper browser, it runs on nearly all platforms, which was a key requirement for the game. Its simplicity in drawing 2D objects with the canvas element would be useful and make development faster. A couple of the team members were already familiar with HTML5 and JavaScript, which would present and manipulate the canvas. Furthermore, it would be useful for the whole team to increase the knowledge of

JavaScript, as it is the most widespread programming language on the web.

3.1.2 Back end language

Java
Again Java presents itself as the most obvious option, as its use as a back-end language with the Spring framework is widespread. Personal preference was one of the decisive factors, as well as productivity. Compared to the alternatives, it seemed like this would be the most time-consuming option.

Python
With web frameworks such as Django, Flask and others, Python was quickly named as a decent option. Most of the team members had some experience with the language and frameworks, and writing Python is quick (and fun, according to some). As an engine for a real time application though, it was considered unsuitable.

Node.js / JavaScript *
This platform was the most foreign to the team, yet it showed a lot of promise. First of all, the entire project could be written in one language, namely JavaScript, which would really hammer the concepts of prototyping (in JS) and give the team more experience with this popular language. Node is event-driven and relies heavily on asynchronous functions, which suited well with the imagination of producing an event-driven game. In the end, these factors made this option the best fit for making "Don't Panic".

3.1.3 Data transfer protocol

JSON *
Given that all the writing would be done in JavaScript, choosing JSON (JavaScript Object Notation) was easy. As JSON looks exactly like JavaScript objects, it was easier to understand and work with JSON than for example XML, which looks more like HTML and did not really suit the needs for a simple protocol to send commands and JS objects through.

3.1.4 Database

MongoDB
An analysis was made of MongoDB which actually stores the data in JSON documents instead of tables. This would be convenient, since it had already been decided to use JSON for data interchange. However none of the team members had any knowledge of MongoDB and it would be time consuming to learn it.

NoSQL
NoSQL is efficient for storing a large amount of data that does not necessarily need to be structured. It does not offer any functionality beyond storage (like keys). It is faster than relational databases like MySQL. However there was no need for storing a large amount of data, and the data was structured. Therefore this was not a very good option.

MySQL*
MySQL is the world's most popular and used open source database. It is used by e.g. facebook, wikipedia and google. MySQL is a relational database management system and therefore fits well with the data. The team members had some experience with MySQL from earlier courses such as

TDT4145 Data Modeling, Database and Database Management Systems and IT1901 Project 1. In addition the team knew that IDI could provide a MySQL database on their server, which was convenient. MongoDB seemed like a promising alternative, but this option was considered more time consuming than MySQL. That is why MySQL was chosen.

3.2 Frameworks

Socket.io *
This is the go-to JavaScript library for real-time web applications using Websockets. It contains a client-side library that runs in the browser, and a server-side library for node.js. Like node.js, it is event-driven.

Node-mysql *
This is a node driver for mysql. It enables connection to mysql database with JavaScript.

jQuery *
The jQuery library simplifies access to the DOM, provides animations and easy element content manipulation.

Express *
This is a web development framework for node.js, that simplifies access to routing, requests and sessions.

3.3 Versioning

Subversion
This is often the standard versioning system used at NTNU, as a repository is provided by IDI, and the team members have used it in several courses already. It is a centralised system and more mature in its development than Git, the alternative. However, it is slow in comparison. Branching is cumbersome and if the central server is not available, it can cause significant trouble.

Git *
In Git, all clones of the repository act as a back up, and the system itself is distributed, where a clone on Github (in this case) acts as a communication channel between the users. Some of the team members already had experience using Git, and found it a lot easier and faster to setup and use in practice.

3.4 Project management tools and processes

Google Drive
Google Drive is a file storage and synchronization service provided by Google that enables collaborative editing of the project documentation. For this project documentation Google docs was used. As this is a collaboration based tool, it suited the structure of the work method.

Dia
In addition to the documentation tools for diagrams provided by Google Drive a program for creating various diagrams, Dia was used. This program has templates of almost all UML designs.

Wbstool
Wbstool was used to make the work breakdown structure chart.

Kanban
Kanban is a method for developing software with an emphasis on just-in-time delivery, while making certain not to overload the developers of the system with work. At the heart of Kanban lies the Kanban board; a visual process management tool consisting of a Kanban board and cards. Each card represents a task that can be assigned to members of the development team. The board is divided into sections, separating tasks that have only been defined, from tasks that are in progress and tasks that are finished. Using this system, any member of the group can create and assign tasks to other group members, and keep an eye out for who is doing what at any given time.

3.5 Existing solutions

Since this is an original board game developed by the customer, there are no alternative electronic solutions of this game already developed. However, there is a large number of other board games that have been adapted into a digital version using various technologies.

Examples:
- Chess: http://plainchess.timwoelfle.de/
PlainChess is a chess implementation built completely using HTML5 technologies. The game engine is written in JavaScript and relies on the frameworks jQuery and jQuery UI, and games can be played both with and without the use of an internet connection.

- Planet Sudoku: http://planetsudoku.com/
Planet Sudoku is a robust, customizable HTML5 Sudoku game supporting different kinds of Sudoku rules and difficulty settings.

- Bombermine: http://bombermine.com
Bombermine is a massively multiplayer online adaption of the classic strategic puzzle game Bomberman by Hudson Soft/Konami. The game was made with HTML5 and JavaScript, using the AngularJS and async.js frameworks. Bombermine won the Best Web-Only Game at the Mozilla Game On 2013 competition, and although it is not as similar to a traditional board game like Don't Panic is, it really shows the possibilities for HTML5/JavaScript games.

After having reviewed these example games, it was clear that the chosen technology of HTML5 and JavaScript was a good choice for creating the game.

Chapter 4

Requirements

4.1 Functional requirements

FR1 - Expert Interface
A crisis management expert should be able to set up a game template for teams to use. This interface should be able to configure as many variables in the game as possible(game rules), and be able to create maps, change the number of players, where they start, what events can occur and when, and create information cards.

FR2 - Game Manager
An expert should be able to enter an existing game as a Game Manager. The GM should be able to trigger events and modify the game objects on the fly to make the game more dynamic, as well as comment on player actions. The GM should be able to monitor activity on the server, existing sessions and online players.

FR3 - Player Profiles
Each player should have a profile that records the players performance in played games. This includes tracking wins and losses, listing game replays and other metrics that are relevant.

FR4 - Replay
An expert should be able to view finished games as a re-play, to evaluate player performance. These replays should be stored in , to be viewed at any time.

FR5 - Game functionality
In addition to adapting the functionality of the board game version to an electronic platform, as specified in the appendix, the number of people in a zone should affect how panic spreads between zones, and inside them.

FR6 - Physical interaction
Preferably, the game should be able to respond to commands sent by interaction devices such as Arduino or Sifteo cubes. They could be used to represent zones, players or be used as controllers for movement.

4.1.1 Use cases

The use cases are mainly based on the functional requirements of the game and are a graphical representation of the users' interactions with the board game. They document all the different ways in which the user can interact with the game.
A detailed set of use case diagrams and textual use cases are provided below.

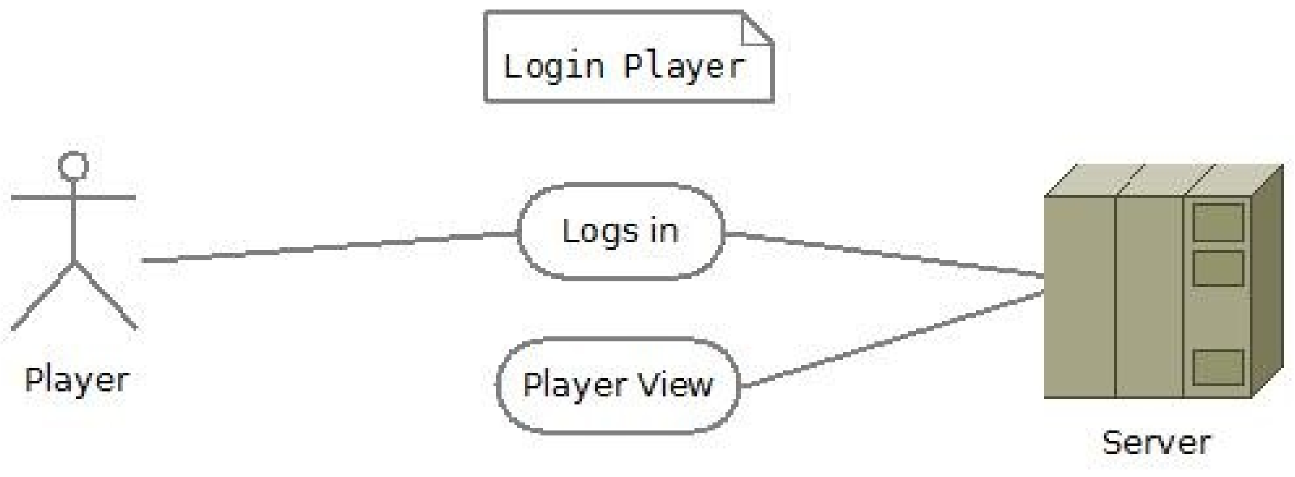


Figure 4.1: Use cases, 'Log in for player'

ID	**01**
Name	Login Player
Goal	To be connected to the server
Actors	Player, server
Start requirements	None
End requirements	- The player gets logged in. - The game is displayed.
Case	- The player clicks on the option for player, in the middle of the HTML starpage - The player gets prompted with the login form. - The player gives login-info. - The player clicks the login button to the rigth of the form. - The player is now logged in. - The player is moved to the player's page.
Alternative Case	Wrong password
Previous Use Case	None
Spawned Use Case	05

Table 4.1: Use Case: Login player

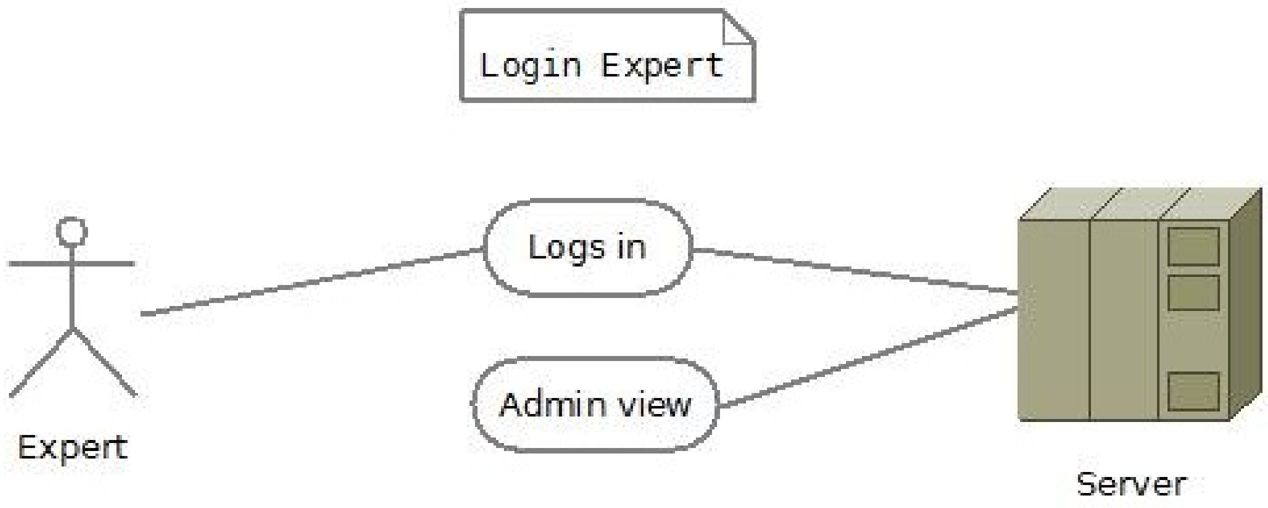


Figure 4.2: Use cases, 'Log in for expert'

ID	**02**
Name	Login Expert
Goal	To be connected to the server
Actors	Expert, server
Start requirements	None
End requirements	- The expert gets logged in. - The expert view is displayed.
Case	- The expert clicks on the option for expert, in the middle of the HTML starpage - The expert gets prompted with the login form. - The expert gives login-info. - The expert clicks the login button to the rigth of the form. - The expert is now logged in. - The expert is moved to the expert's page.
Alternative Case	Wrong password
Previous Use Case	None
Spawned Use Case	03

Table 4.2: Use Case: Login expert

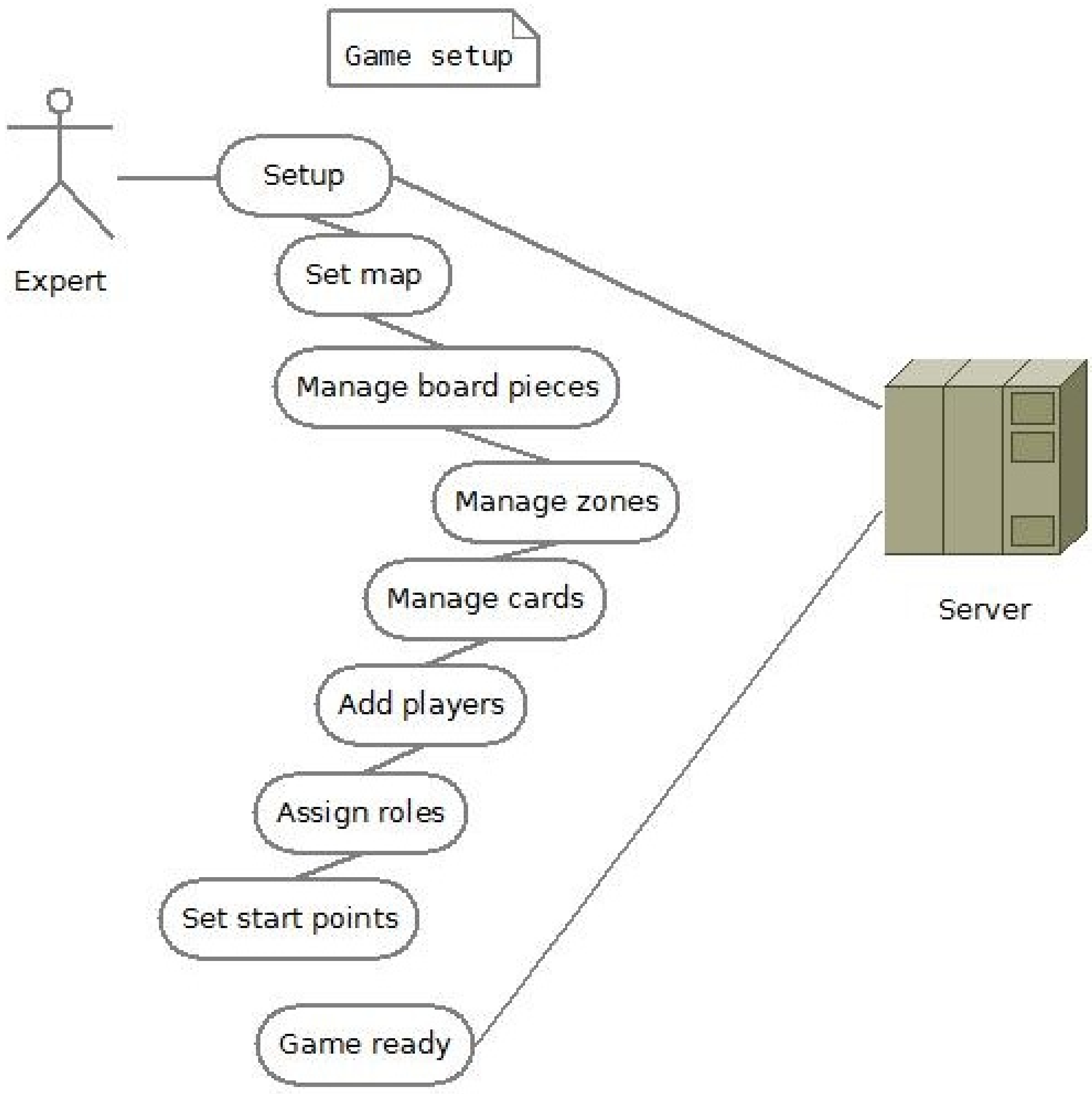


Figure 4.3: Use cases, 'Game setup'

ID	**03**
Name	Game Setup
Goal	To create a successful game session
Actors	Expert, server
Start requirements	The expert is logged in
End requirements	The expert is able to create a game setup The expert is able to save the game setup
Case	- The expert creates the appropriate map for the game. By ploting nodes into the canvas and creates paths between them. To create a zone the expert selects a minimun of 3 paths and clicks for create zone. - The expert adds the wanted board pieces by clicking on the corresponding buttons for the different game pieces, while in the wanted node. - The expert selects what type of zone the selected zone should be, and does this for each zone. - The expert set the number of people for each zone, by selecting a zone and using the initial people button at the top of the canvas. - The expert sets the initial panic for each zone by selecting the zone and the corresponding button for initial panic, at the top of the canvas. - The expert manages the cards, there is an initial set of cards for the game, but if the expert wants he can add more, or special cards at the top of the page, over the canvas for drawing the map. - The expert adds the wanted number of players to the game, by selecting a node and by using the add player button. - The expert can set a individual starting point to each player by selecting the wanted node and by using the button for add player at the top of the canvas. - The expert assigns roles to each player, by selecting a player from the form over the canvas for creating the map.
Alternative Case	None
Previous Use Case	02
Spawned Use Case	04, 05

Table 4.3: Use Case: Game Setup

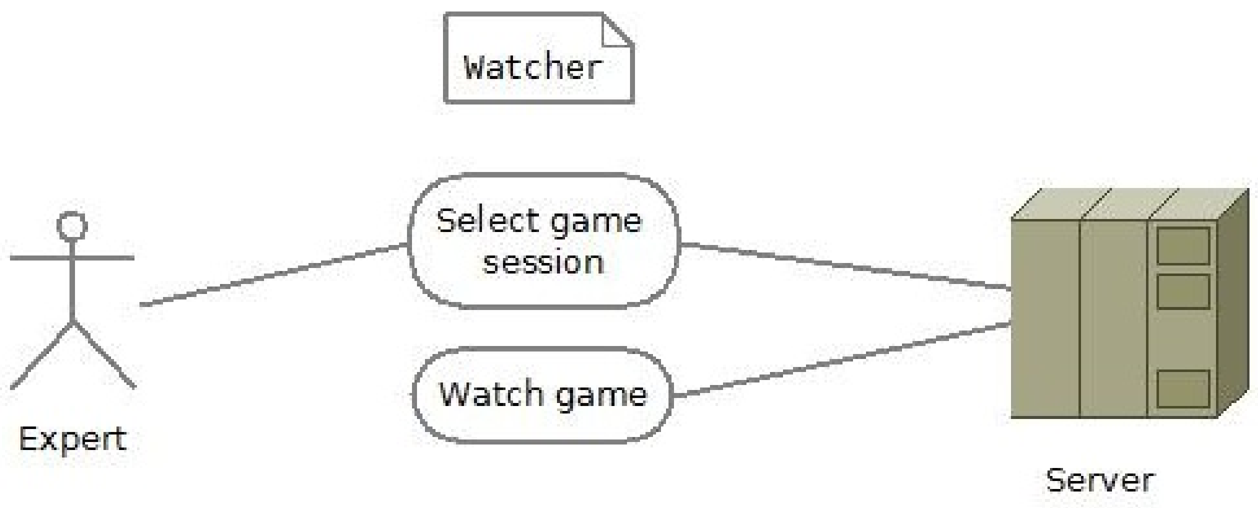


Figure 4.4: Use cases, 'Watcher'

ID	**04**
Name	Watcher
Goal	To get a non player version of the game
Actors	Expert, server
Start requirements	The expert is logged in, a game is running
End requirements	The expert is able to watch the wanted game
Case	- From the monitor game option at the top of the expert page, the expert gets a list of games in session, the expert selects one of these to start monitoring the game. - The server provides a game window in which the expert is not participating as a player
Alternative Case	None
Previous Use Case	02
Spawned Use Case	None

Table 4.4: Use Case: Watcher

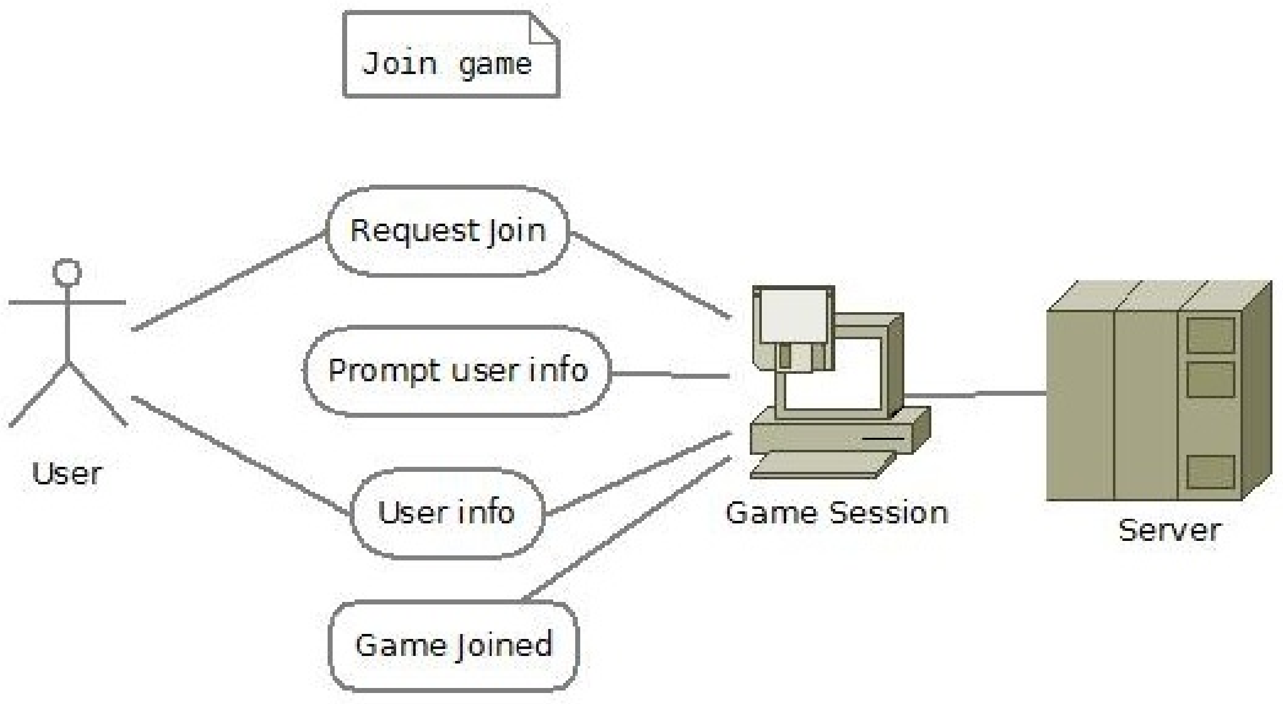


Figure 4.5: Use cases, 'Join game'

ID	**05**
Name	Join Game
Goal	To successfully join a starting game
Actors	User, game session, server
Start requirements	A game has been created by the exper
End requirements	A user is able to join the appropriate game
Case	The user clicks on join options for the game in the middle of the page - The game asks for user info in a popup form, the user enters the info. - The user joins the game
Alternative Case	The user gives incorrect info and is not added to the game
Previous Use Case	03
Spawned Use Case	06, 07

Table 4.5: Use Case: Join Game

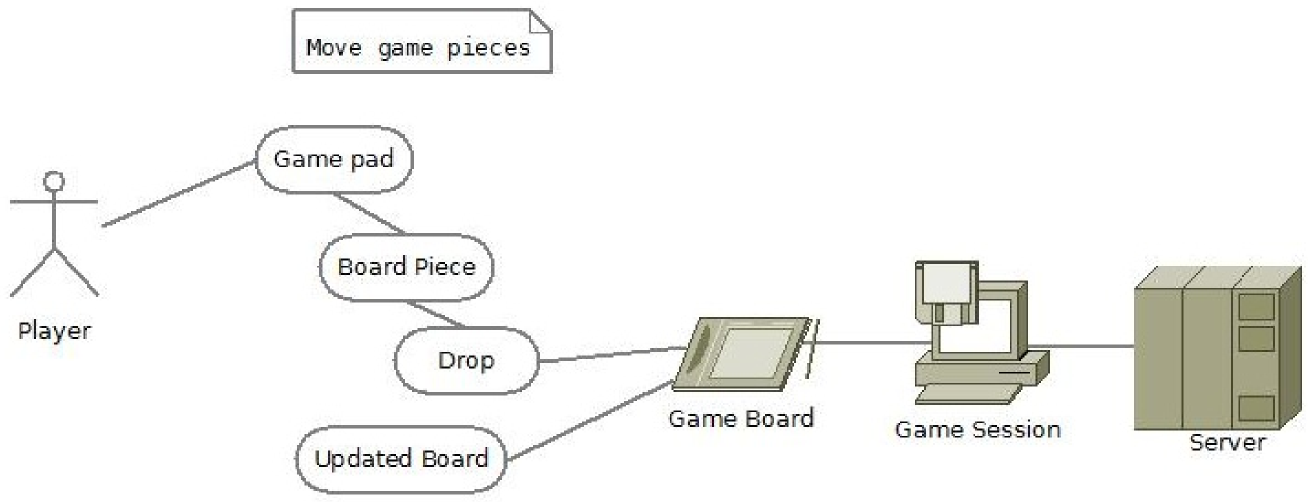


Figure 4.6: Use cases, 'Move game piecec'

ID	**06**
Name	Move game pieces
Goal	To move a game piece to a wanted location
Actors	Player, game board, game session, server
Start requirements	The player has joined a game The player in question has the turn
End requirements	The player is able to move the selected piece to the wanted position.
Case	- The player uses the mouse pad to select the wanted object. - The player drags the object through the path to the wanted location. - The game board is updated.
Alternative Case	The player selects an immovable object The player moves the object to an unobtainable location
Previous Use Case	06
Spawned Use Case	None

Table 4.6: Use Case: Move game pieces

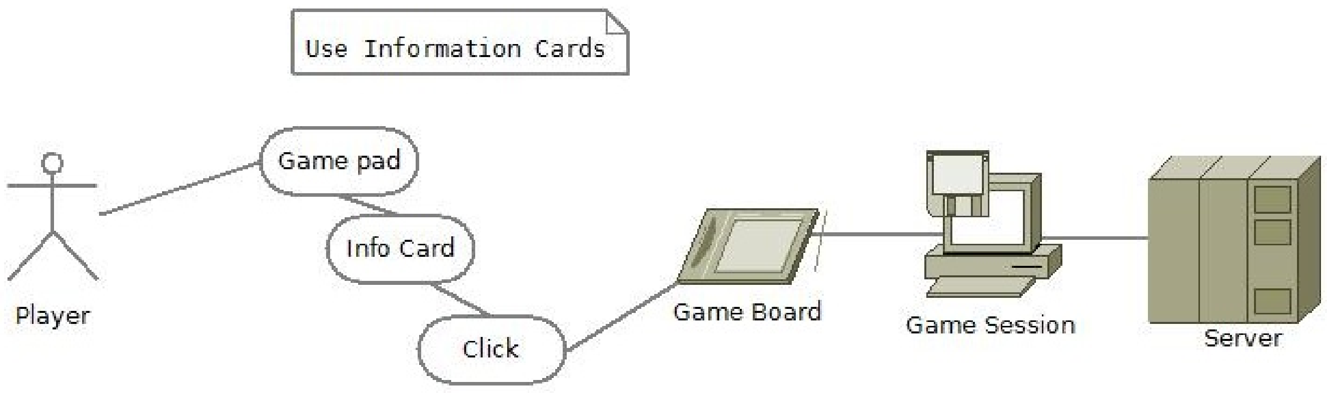


Figure 4.7: Use cases, 'Use information cards'

ID	**07**
Name	Use information cards
Goal	To use an information card to affect the board
Actors	Player, Game Board, Game Session, Server
Start requirements	The expert has created a game The player is logged in The player is part of a game The player has an information card
End requirements	The card effect is carried out on the board The player does not have the used information card
Case	The player clicks on the wanted information card under his player profile to the side of tha canvas. - The information card effect is carried out on the board - The player loses his information card
Alternative Case	None
Previous Use Case	05
Spawned Use Case	None

Table 4.7: Use Case: Use information cards

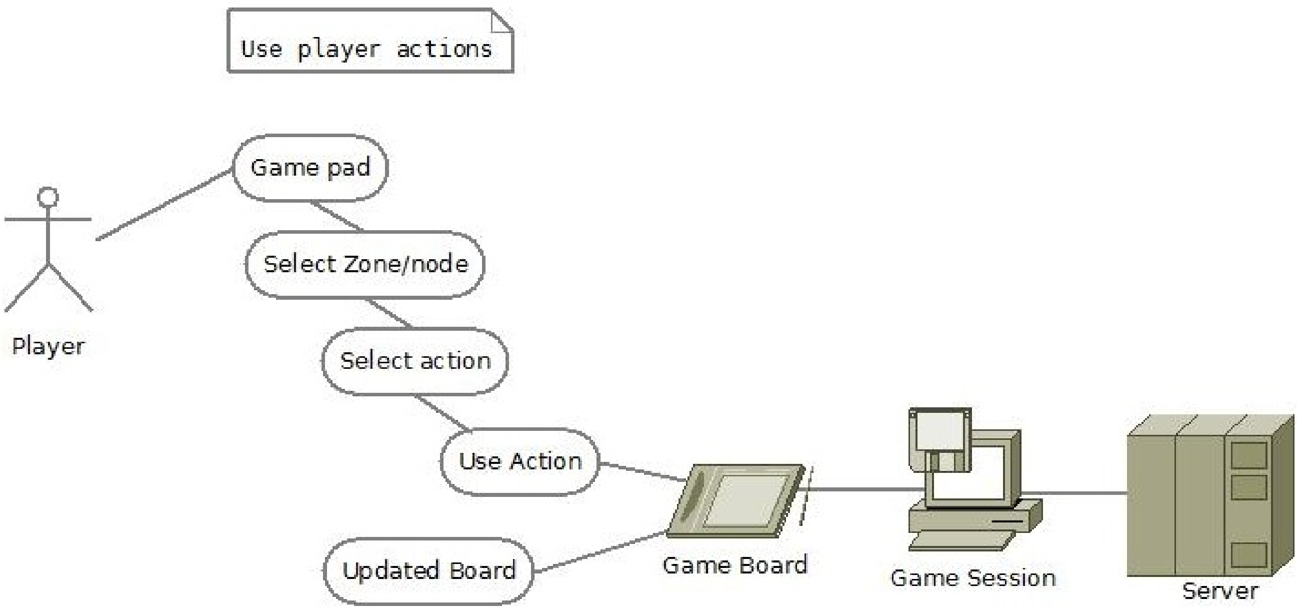


Figure 4.8: Use cases, 'Use player actions'

ID	**08**
Name	Use player action
Goal	The player uses an action and the game board is updated
Actors	Player, Game Board, Game Session, Server
Start requirements	The user is logged in The user is a player in the game A game is in action
End requirements	The player uses an action The effect is updated on the board
Case	- The player selects a adjacent node or a zone with the mouse pad. - The player selects an action wich will appear at the top of the canvas after selecting a node or zone. - The action is used on the target. - The game board is updated.
Alternative Case	None
Previous Use Case	05
Spawned Use Case	None

Table 4.8: Use Case: Login expert

4.2 Non functional requirements

All non functional requirements comply with the definitions as stated in the ISO 25010 standard (replacing ISO 9126). Only relevant requirements are mentioned in this report.

4.2.1 Quality in use

1: Efficiency
Like regular board games, actions should not be difficult to execute. The players are working against

the clock (the panic increase timer). Hence, when designing the user interface, one of the aims should be to minimize the number of clicks required.

2: Context Coverage
The system should be flexible enough to accommodate individual experts' preferences and needs in their simulations. By relying on the settings given by the expert through the expert interface form, the best possible flexibility can be ensured.

4.2.2 Product quality

1: Functional suitability
Functional completeness should be achieved to include the core functionality of the board game, as well as the functionality specific to the electronic version, like the expert interface and panic- and people management.
Core functions must be without game-breaking bugs to ensure functional correctness.

2: Operability
Usability is considered important, as the users should spend time playing the game and learn how to manage panic, rather than how to operate the game.
By exploiting recognisability from classic board games, a lot of interaction can be made intuitive, given that most people already know how to play board games.
Users of the game will most likely not be as proficient with computers as "gamers" in general. Therefore, it would be a good idea to make the game accessible without having to install any software other than an internet browser.

3: Transferability
The client should be usable on as many platforms as possible (Mac, Windows, Linux, Mobile platforms), and in the best possible case be able to interact with devices such as Arduino. HTML5 with node.js was chosen for this reason, as it can run on nearly any device without the need for time consuming installation procedures, thereby increasing portability.

4.2.3 Technical requirements

These requirements have been copied from the "Don't Panic" specifications provided by the customer.

Don't Panic DPS has to meet to the following requirements:
- All interaction between the server and client SHOULD be performed using well documented protocols and standard protocols.
- The DPS Game rules SHOULD be platform independent. Consequently, high level languages such as Java, Processing, Python COULD be considered as good candidates.
- The overall architecture SHOULD be scalable to run multiple Game sessions in parallel without decreasing the quality of already running games sessions.
-Already existing frameworks for game development for such as Unity, Microsoft XNA Game Studio, or management tools such as RedMine COULD be used as platforms to help speeding up the development of the game. The choice should be driven by a framework comparison analysis considering both technical requirements and already existing skills/experience among group participants.

Chapter 5

Design and architecture

5.1 User interface

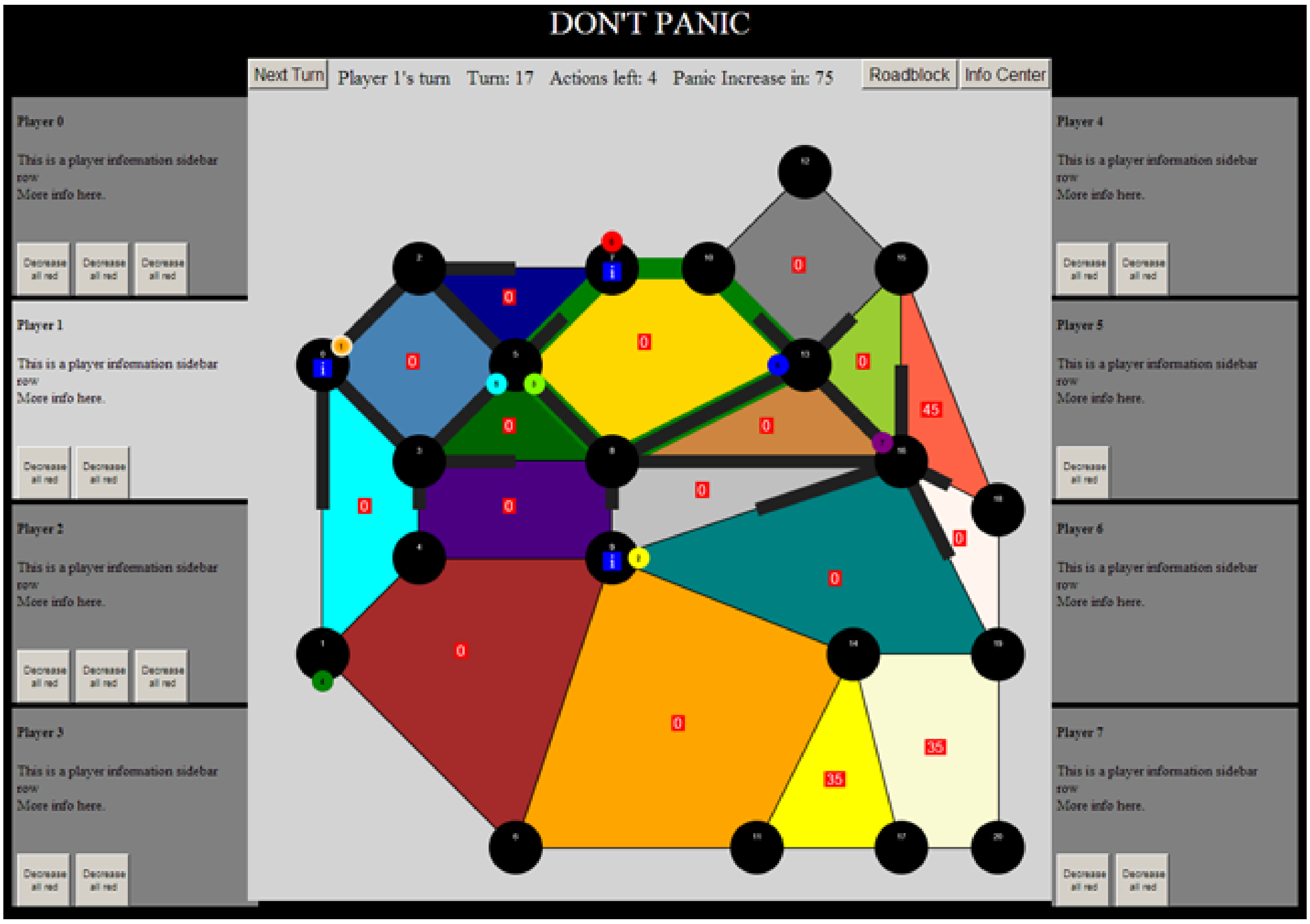
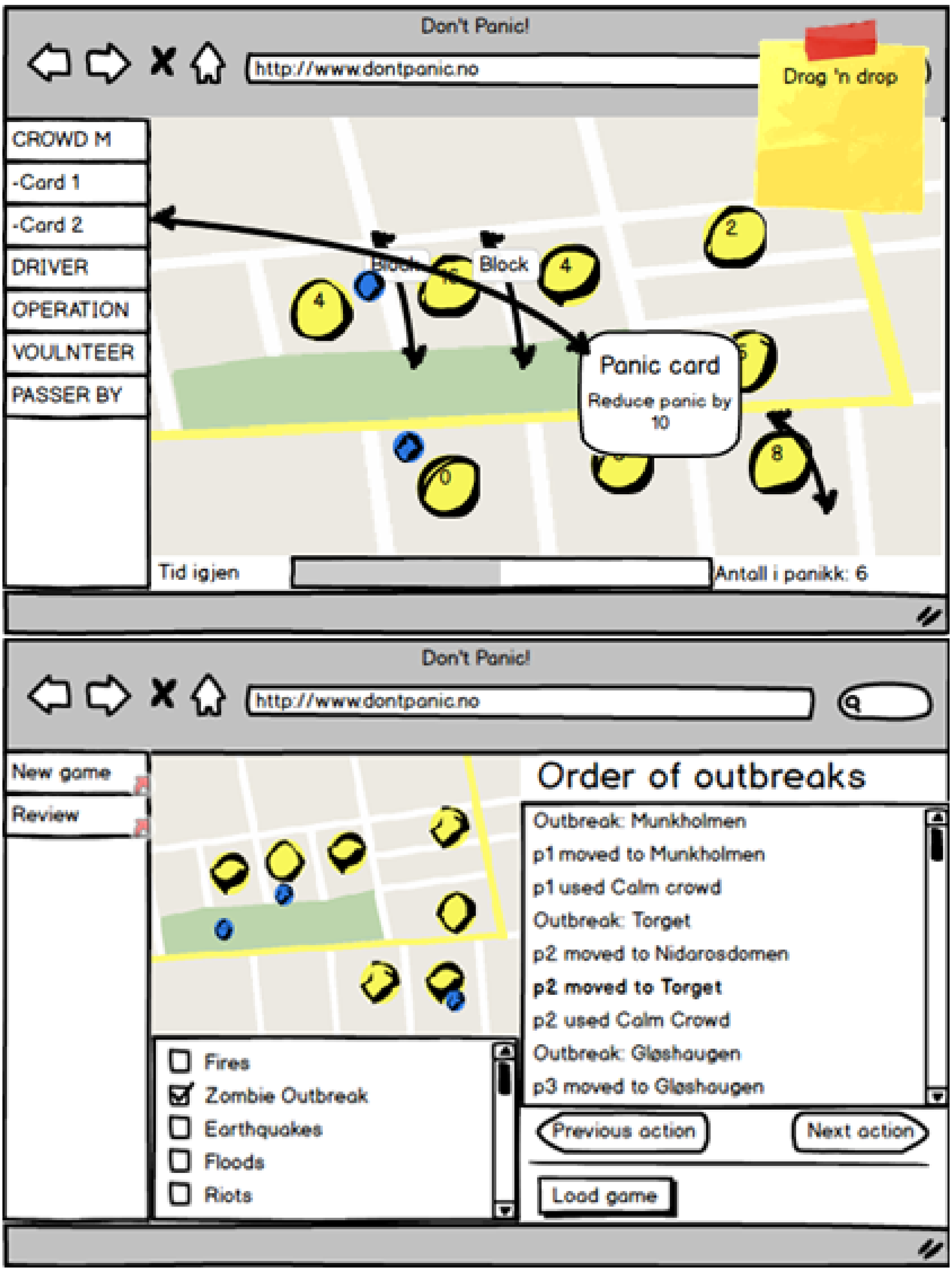


Figure 5.1: User interface, ' The user interface of the game, complete with a board, pieces, cards and information tables'

5.1.1 User interface design

The customer had already designed a physical board game of Don't Panic, which formed the basis for designing the electronic version. In the earliest stages of the designing Balsamiq Mockups was used, an online tool for interface designing.

Mockups is designed to be an easy and efficient tool used in the early stages of interface designing, and it can be used to generate click-through prototypes for interfaces. Through myBalsamiq it can also be used as a collaborative tool, supporting project-based collaboration and real-time changes.



Outbreak: Munkholmen
p1 moved to Munkholmen
p1 used Calm crowd
Outbreak: Torget
p2 moved to Nidarosdomen
p2 moved to Torget
p2 used Calm Crowd
Outbreak: Gløshaugen
p3 moved to Gløshaugen

Figure 5.2: User interface, ' Early design of the board, cards and the expert interface'

5.1.2 Realisation of the user interface

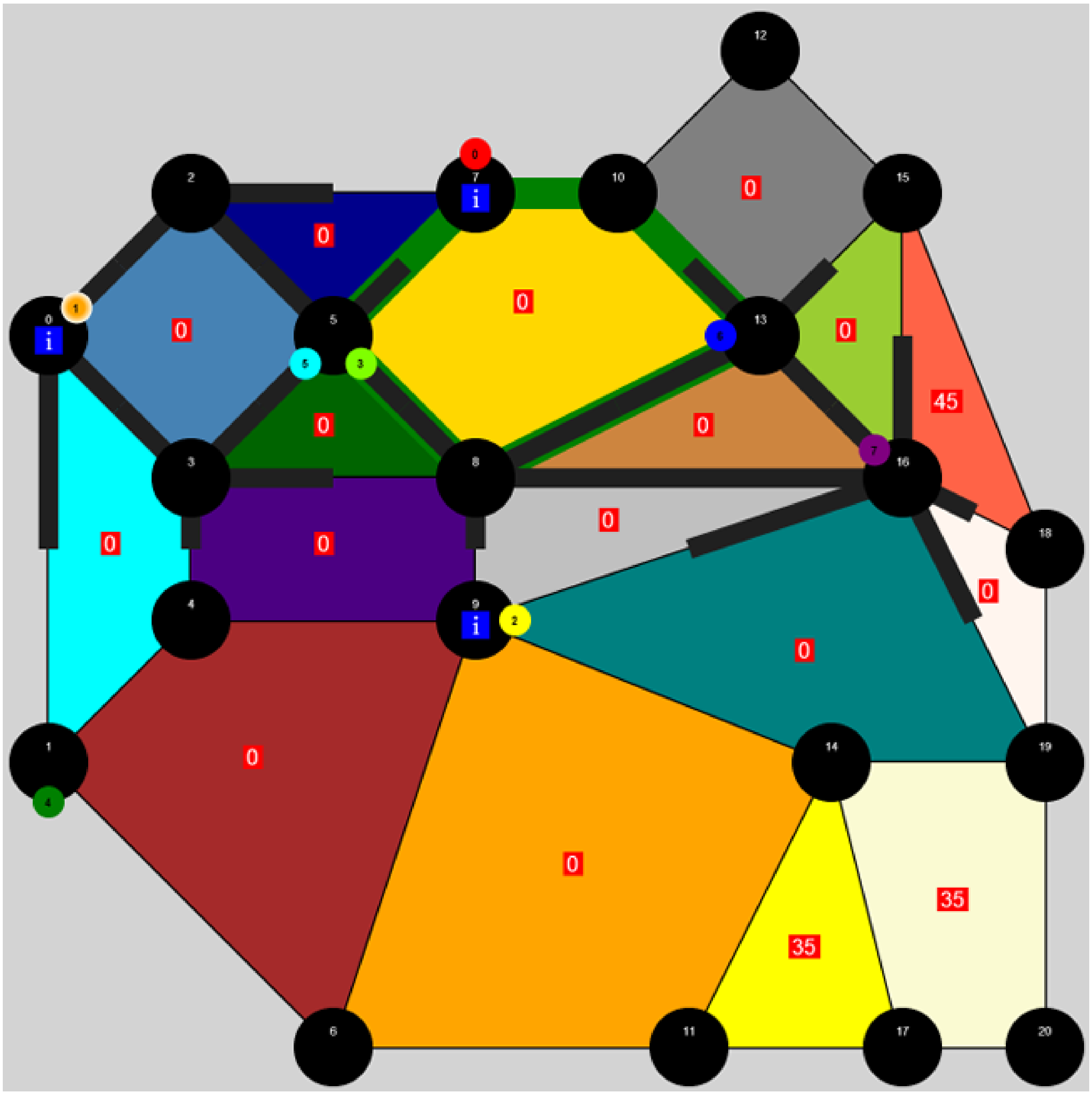


Figure 5.3: User interface, 'The canvas'

The main part of the interface is the HTML5 canvas. The board and pieces of the game are all drawn within this canvas using JavaScript. The players are able to control their respective player pieces by dragging them on the board from one node to another using the mouse, like one would do using the hands when playing the physical version of the board game. One of the main goals of the project was to preserve the physical interaction of playing a board game in the electronic version, so it was decided that implementing a drag-and-drop functionality for the player pieces would be a good idea.

The panic levels for each zone are printed inside the zones. This was very cumbersome with the physical version, as the players were forced to update all the zones manually each time the timer counted down, as well as when event cards and information affected the zones. This is now handled automatically on the server.

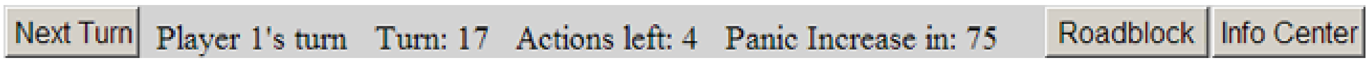
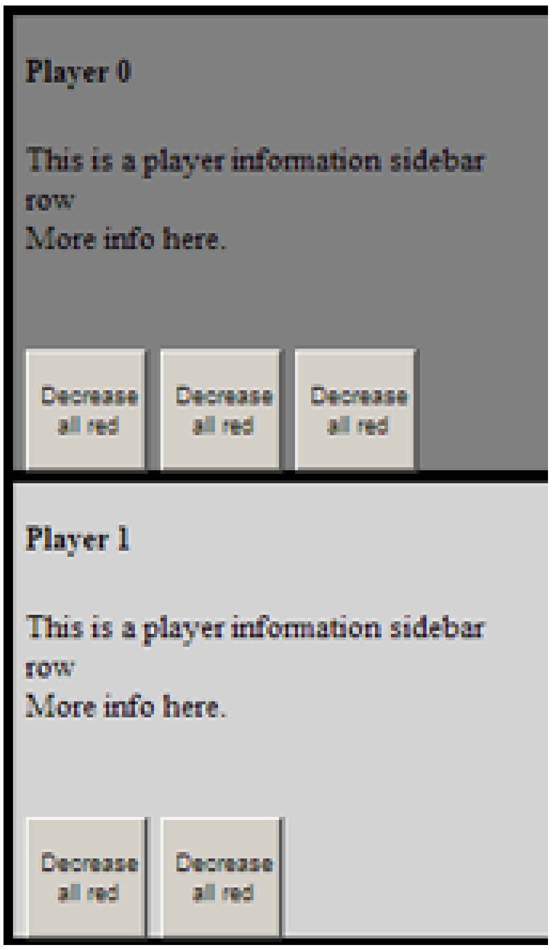


Figure 5.4: User interface, 'The head table'

An information table is placed above the canvas. This table contains the button the players would use when ending their turns, as well as buttons for placing information centers and roadblocks on the nodes they are located. The table also contains information about whose turn it is, how many turns have passed, how many actions the active player has left and a timer which counts down to the next increase in panic.

On each side of the canvas there are sidebar rows. Players have their own row, which is highlighted when it is their turn. The rows contain information on players, as well as their information cards. Since this is a collaborative game, all the cards should be visible to the players so they can openly discuss how the cards should be used, as well as trade cards between themselves. This preserves the verbal interaction dynamics of a physical board game. The cards are implemented as buttons with text corresponding to their effect in the game.


Player 0

This is a player information sidebar row
More info here.

Player 1

This is a player information sidebar row
More info here.

Figure 5.5: User interface, 'Sidebar rows'

5.2 Client/Server architecture

A client-server model was chosen as the architectural pattern. This was highly desired by the customer, as they wanted different clients to work with the server. Therefore using a different architecture was never considered as an option.

5.2.1 Initial suggestion for a high level architecture from the customer

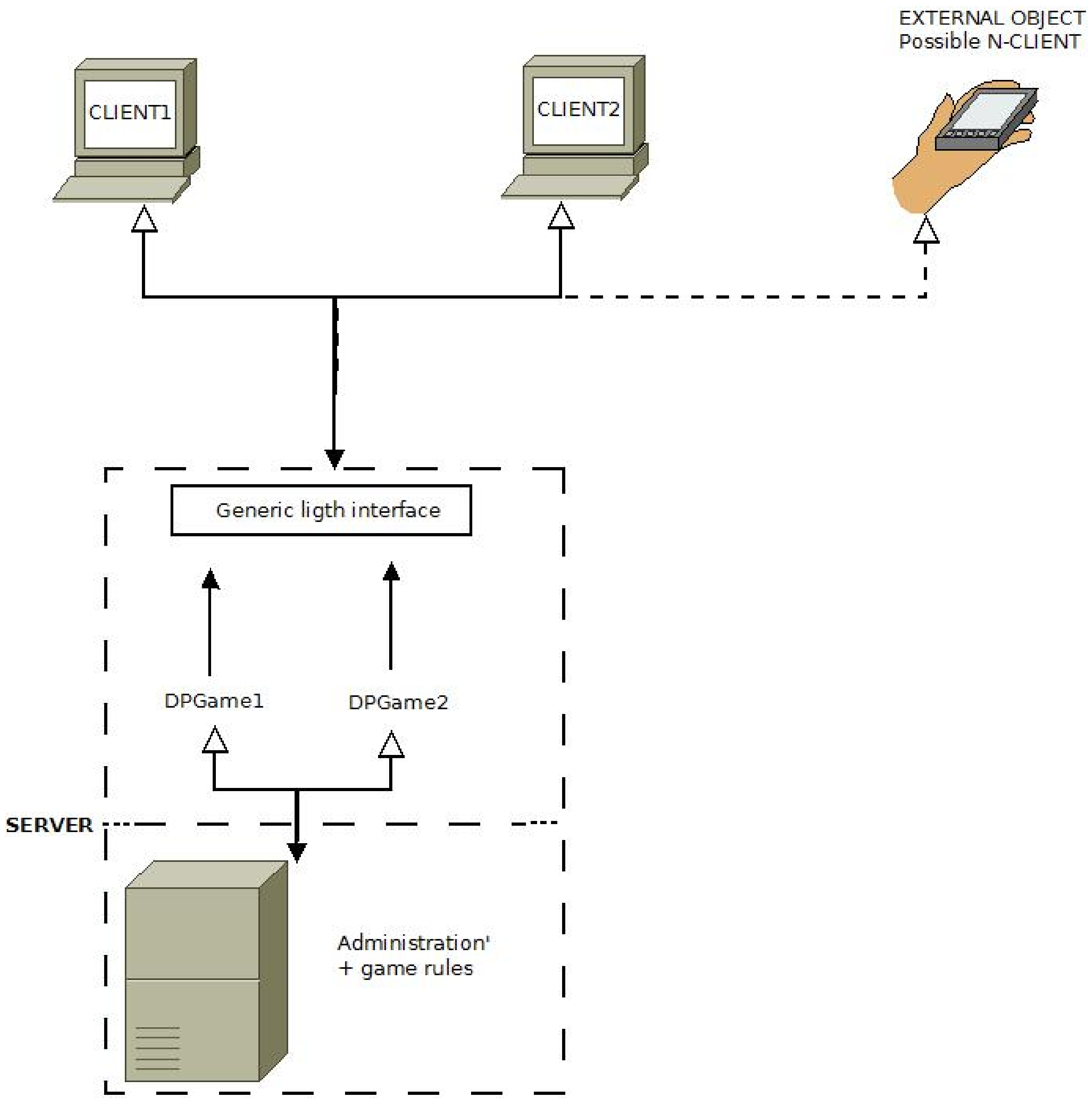


Figure 5.6: System architecture, 'Initial suggestion'

5.2.2 High level system architecture

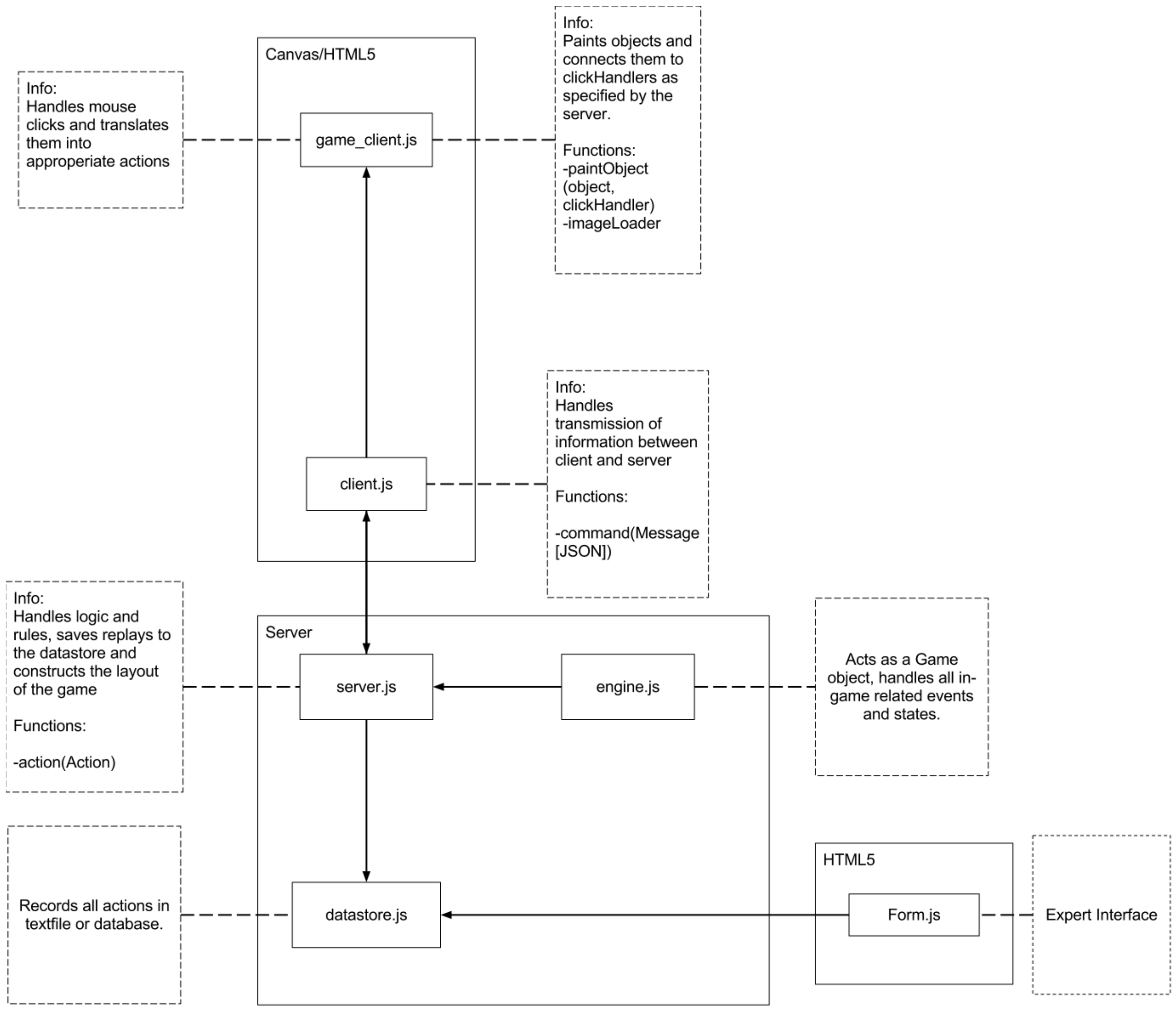

Canvas/HTML5
game_client.js
client.js
Info:
Handles mouse clicks and translates them into approperiate actions
Info:
Paints objects and connects them to clickHandlers as specified by the server.
Functions:
-paintObject (object, clickHandler)
-imageLoader
Info:
Handles transmission of information between client and server
Functions:
-command(Message [JSON])
Server
server.js
engine.js
datastore.js
Info:
Handles logic and rules, saves replays to the datastore and constructs the layout of the game
Functions:
-action(Action)
Acts as a Game object, handles all in-game related events and states.
Records all actions in textfile or database.
HTML5
Form.js
Expert Interface

Figure 5.7: System architecture, 'High level system architecture'

5.2.3 Object diagram

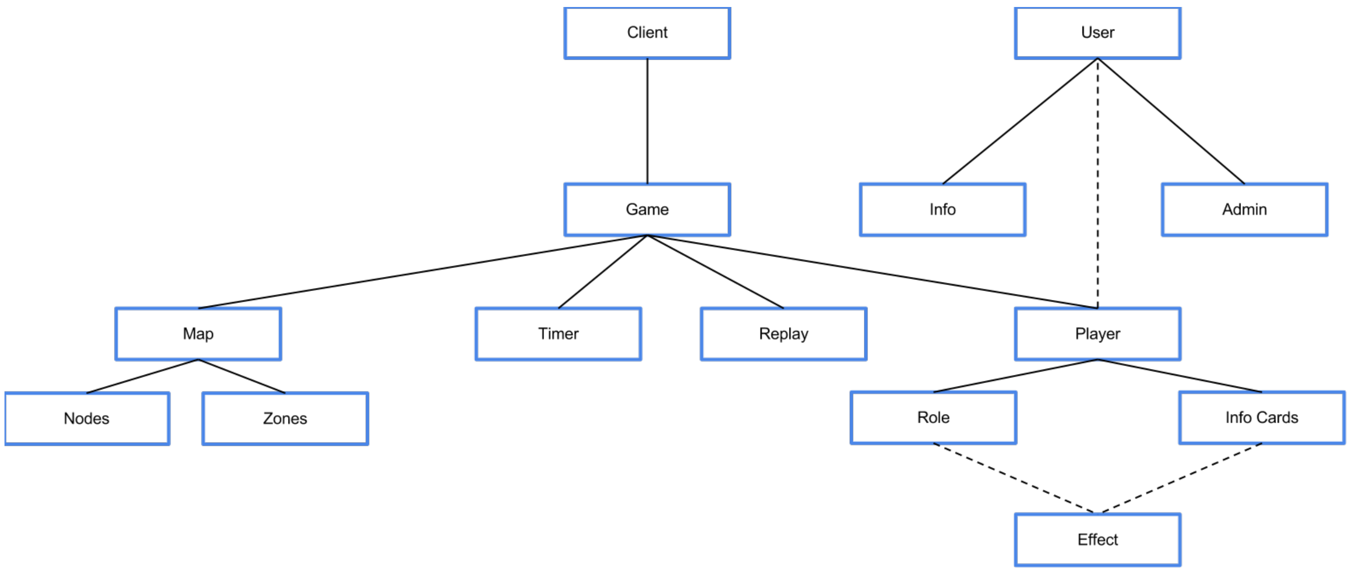


Figure 5.8: Object diagram, 'Game tree'

5.2.4 ER diagram

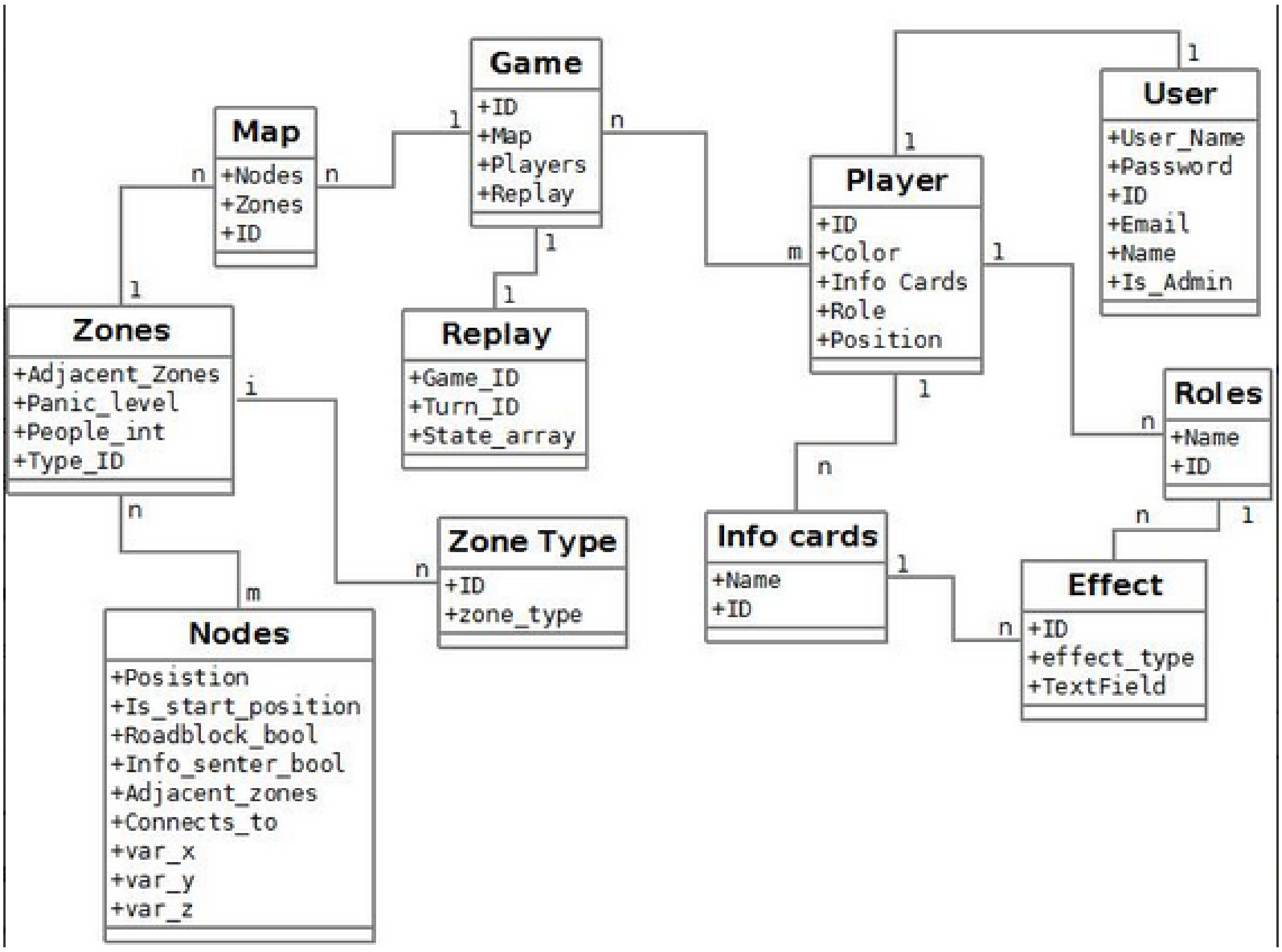


Figure 5.9: 'ER diagram'

5.2.5 Object hierarchy

This chart is intended to show a high level view of the hierarchy, and how the objects are connected together to avoid loops.

5.2.6 Sequence diagrams

The sequence diagrams show the interactions between the files, functions and methods. It depicts the objects and files interactions in the right time sequence. The diagrams also show the calls each method or a file sends to the next file, function or method.

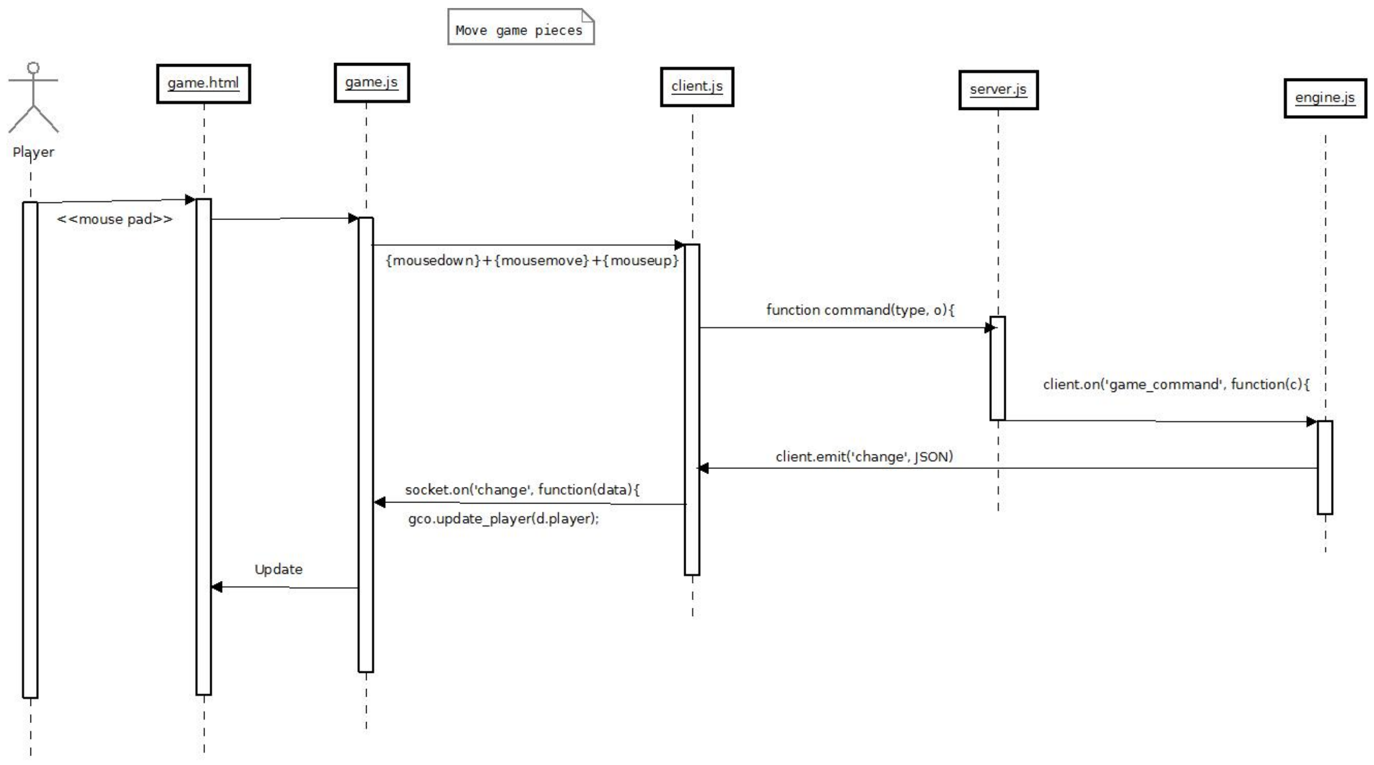

Move game pieces
Player
game.html
game.js
client.js
server.js
engine.js
<<mouse pad>>
{mousedown}+{mousemove}+{mouseup}
function command(type, o){
client.on('game_command', function(c){
client.emit('change', JSON)
socket.on('change', function(data){
gco.update_player(d.player);
Update

Figure 5.10: Sequence diagram, 'Move game piecec'

The above diagram show the interaction between the JavaScript files when a player decides to move his or hers game piece. The player initializes the sequence by clicking on the game piece in his HTML view. The first file after the HTML view is the game.js, the file that handles and interprets player inputs. The game.js file interacts with client.js which is the file handling every visual update of the game board. The file server.js Imports http and makes a server that socket.io can listen to; it connects the game interactions and game view with the engine. The file engine.js handles the game logic and game rules. When the interaction is at this point the engine.js fires a change to the client that in turn updates the view in the HTML file.

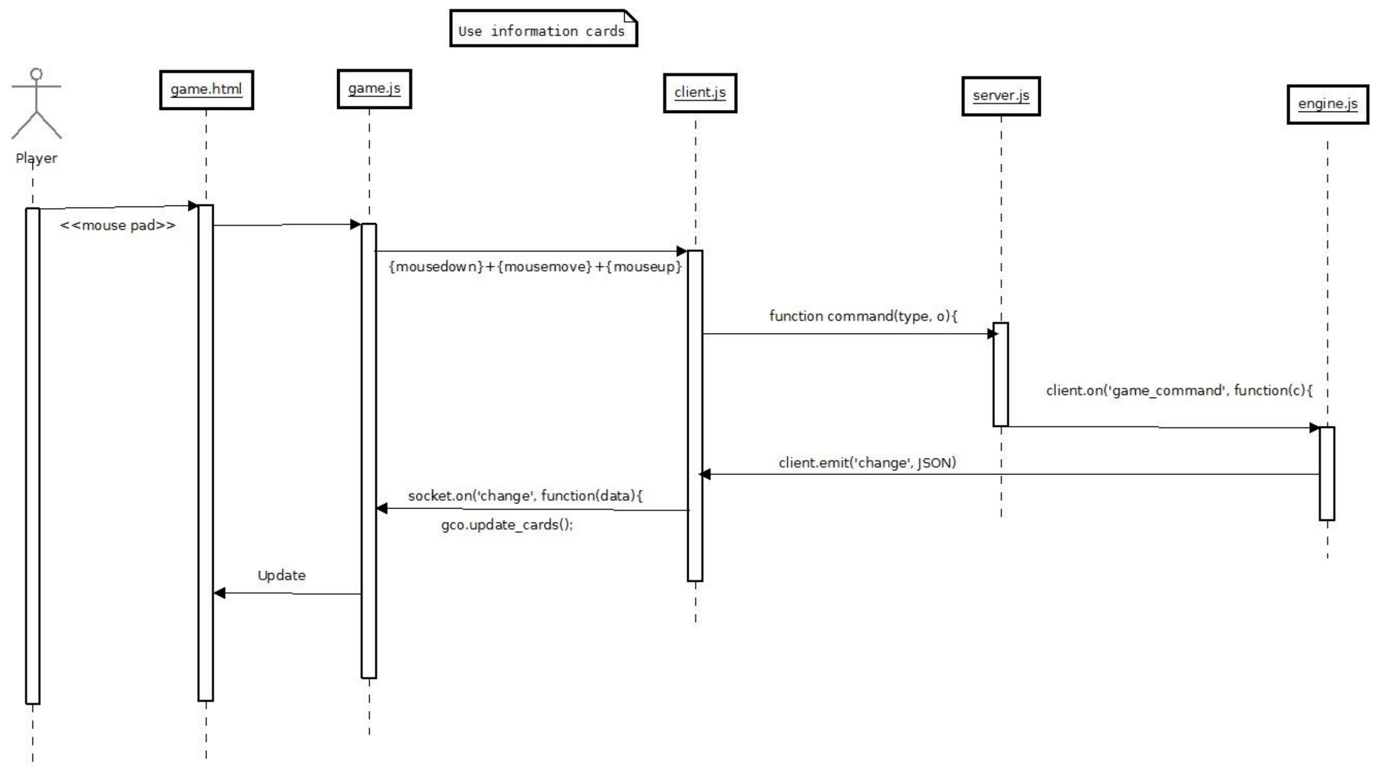

Use information cards
Player
game.html
game.js
client.js
server.js
engine.js
<<mouse pad>>
{mousedown}+{mousemove}+{mouseup}
function command(type, o){
client.on('game_command', function(c){
client.emit('change', JSON)
socket.on('change', function(data){
gco.update_cards();
Update

Figure 5.11: Sequence diagram, 'Use information card'

The above diagram shows the interactions of the JavaScript files when a player uses an information card in the game. The sequence is very similar to the previous diagram but is different in the types of data or methods it sends between the files. At the end it updates different parts of the game (gco.update();).

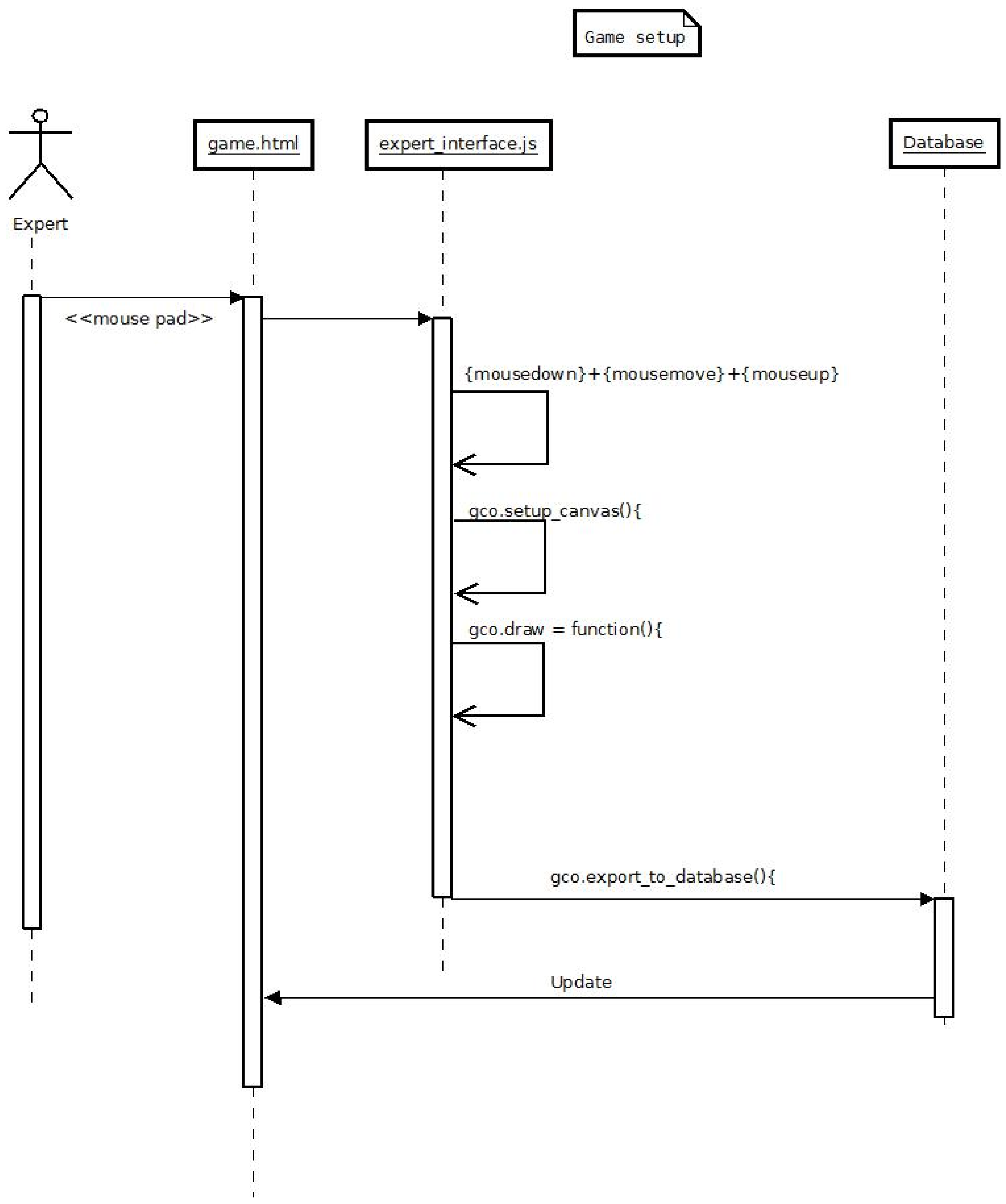


Figure 5.12: Sequence diagram, 'Game setup'

The above diagram shows the interaction of the JavaScript files when an expert creates a game, the details of this is explained thoroughly in the use cases. The expert uses the mouse pad to interact with the game.html page, and is directed to the expert interface. The file expert interface.js is where all the dragging of the nodes, and creating and placing of the zones and players are created. When the expert is finished the game object is stored to the database. The last interactions is when game.html is updated when it get notifications from the database. The game can now be chosen from a list in game.html and played.

Chapter 6

Testing

6.1 Test plan

The game will be tested thoroughly by both Black Box and White Box testing. Black Box testing will be the main point in this test plan. White Box testing will be done throughout the project and continuously, while the code is being written. An analysis was made of the pros and cons of writing small automated tests for the game, and the conclusion was that this would be more work for not enough profit. When the core of the game is finished the game development process switches from intertwined code to adding rules and actions to the game. This can be seen as layers. When the layering on top of the game core is done, it is not that difficult to test the added methods by itself. For this reason automated tests will not be written.

For the Black Box testing a series of tests will be written, based upon the requirements specified in Chapter 2: Requirements. The Black Box testing implies thorough interaction between the team and the game. The aim is to see how the different objects on the map interact with each other, to find incorrections and glitches. In addition to testing the game for errors usability tests will be gone through. The usability tests will be done with an external actor to ensure independent feedback.The testperson will go through the different use cases to ensure satisfiability.

6.1.1 Black Box testing

The Black Box testing will shadow every part of the game. It is important that every part of the game is as error free and glitch free as possible. These are the test areas which will be looked closely at:

- The start of the game, such as login and menus
- The art of the game. These include the board model, player models and other object models contained in the game
- The movements of the game, moving board pieces around and the use of information cards
- The view of the game board, borders and accurate frames
- The game flow, how the board game reacts to different turns
- The events and triggers within the game
- The interaction of the gamepad within the game
- The game rules (the tester needs to be familiar with the rules)
- A test for objects overlapping within the game, clipping
- Testing for multiplayer version, running more than one game, and as many as possible at one point

- Testing memory overload by leaving it on for an extended period of time. This is one of the few negative testing features that the game will go through
- A test for platform compatibility, since this is HTML5 based the platform will be different web browsers

After testing has been done thoroughly, the testing phase will go on to tests with people unattached to the project. Unattached people will be used because it will be helpful to get feedback from someone outside the group. If there is something missing or incomprehensible, it provides the opportunity to correct or improve the error. The test person will go through the test provided, based on the use cases from the requirements. Common formalities such as voluntariness, choices and uncomforts will be gone through before the tests are made.

Some points that will be confined to are:

- Under the tests the tester will not receive any help (unless unforeseen events occur that requires it).
- The tester has the choice to abort the tests at any moment
- The tester should think aloud, so that the choices made will be easier to understand
- The supervisors (the team) should take notes
  - of problems during the tests.
  - when the tester is unsure about what to do.
  - when the tester does something wrong.
  - if the tester does not know what to do at all.
  - of any unforeseen events that occur during the tests.

After the tests are done a SUS sheet will be provided for the tester where he can evaluate the different parts of the system. Here is also the time for discussion and inputs from the tester. The supervisors have at this point taken several notes, and will have some questions about some of them to ask the tester.

The result from each test and the comments will be shown in the tables below. Each table will provide:

- Test number
- Test case
- Comments about the test
- Problems during the tests and comments of these
- Proposals for solutions
- Improvements absolutely needed
- Small tweaks wanted

6.1.2 System usability testing

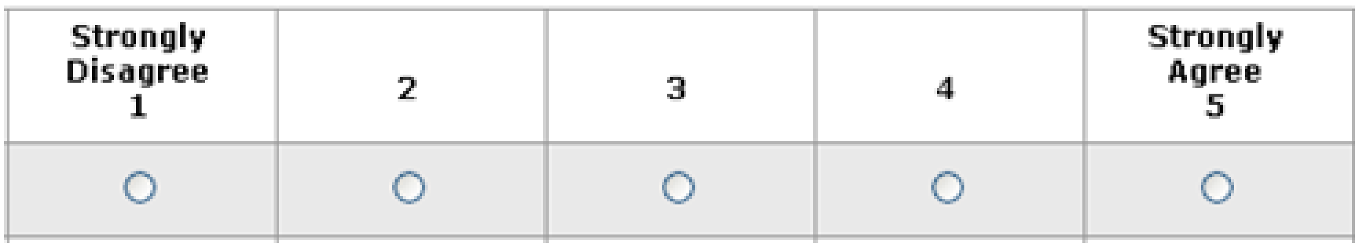
The tester is provided with the SUS (System Usability Score) sheet. The SUS sheet is a questionnaire containing 10 statements. The tester is expected to respond to each statement by choosing one of five options, depending on the degree of agreeability.

The statements are:

1. I think that I would like to use this system frequently.

2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

These are the 5 choices:



Strongly Disagree 1	2	3	4	Strongly Agree 5
○	○	○	○	○

Figure 6.1: Testing, 'SUS choices'

6.1.3 White Box

For the white box testing we have done small tests locally without pushing it to git. We have tried to keep one branch on our git repository free from errors, and only pushing working and error free versions of the game. The white box testing process has been an continuously task throughout the game development. The line "console.log('error');" has of course been of great help. The testing we have done of white box has not been documented, as we found nothing of importance to document when we decided to not use unit tests. Since this is a smaller project, members of the group has had widespread and deep knowledge of the complete source code, and therefore white box testing has been possible.

6.1.4 Tests

6.1.5 Black Box tests

ID	**01**
Name	Game Start
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. Regarding software, only a web browser is required.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.*
Goal	The test is to verify that the player can log in and start a game without problems.
Actors	Player, Game Board, *Game Session, Server*
Test requirements	The player can connect with the server. The player can not log in with incorrect credentials. The player can join the selected game. The player cannot join an unintended game.
End requirements	The player cannot connect with the server. The player can log in with incorrect credentials. The player cannot join the selected game. The player can join an unintended game. The player disconnects.
Disruption criteria	The player tries to connect with the server. The player logs in with his credentials. The player tries to join a game.
Test case	The player tries to connect with the server. The player logs in with his credentials. The player tries to join a game.
Alternative case	The player logs in with incorrect credentials. The player tries to join an unintended game.
Test results	
Test comments	
Improvements needed	

Table 6.1: Black Box Test: Game Start

ID	**02**
Name	Gamepad Interaction
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. Regarding software, only a web browser is required.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.*
Goal	The test is to verify that the gamepad is usable with the game and to the team's satisfaction.
Actors	Player, Game Board, *Game Session, Server*
Test requirements	An expert has created a game. The player is inside a game session.
End requirements	The gamepad works with every part of the game.
Disruption criteria	The gamepad cannot move or interact with the game board. The gamepad can pick up objects but not hold on to them. The gamepad cannot move the objects to another point. The player disconnects. Unexpected faults with the gamepad.
Test case	The player checks the gamepad for interaction possibilities with the game board. The player then checks if the gamepad can move objects around with the gamepad. The player verifies that the gamepad can move objects to another location.
Alternative case	*None*
Test results	
Test comments	
Improvements needed	

Table 6.2: Black Box Test: Gamepad interaction

ID	**03**
Name	Game art.
Time schedule	Approximately 10 minutes.
Enviorment requirements	The test computer must have internet access. Regarding software, only a web browser is required.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.*
Goal	The test is to verify that the game art is according to the standards and has no deviations.
Actors	Player, Game Board, *Game Session, Server*
Test requirements	An expert has created a game. The player is inside a game session.
End requirements	The board model has no deviations (fragments, overlappings, clippings, out of frame, correct colors and form). The player models are as they should be (correct colors, png, way and position). Other object models (blockade and information center) are satisfiable.
Disruption criteria	The board model has one of the mentioned deviations or other impurities. The player model has one deviation or an unexpected fault. Other object models have faults or deviations. The player cannot move or place the intended objects on the board. The player disconnects
Test case	The player checks the board thoroughly for deviations from the planned model. The player then checks the player model for impurities, also while the model is moving. The player checks the other game pieces, information center and blockades, while they are being placed on the board or moving.
Alternative case	*None*
Test results	
Test comments	
Improvements needed	

Table 6.3: Black Box Test: Game art

ID	**04**
Name	Game Movement
Time schedule	Approximately 10 minutes.
Enviorment requirements	The test computer must have internet access. Regarding software, only a web browser is required.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.*
Goal	The test is to verify that the game movements are functional and according to the standards.
Actors	Player, Game Board, *Game Session, Server*
Test requirements	An expert has created a game. The player is inside a game session.
End requirements	The movement of the player model is satisfiable. The placement of the information center model is satisfiable. The placement of the blockade model is satisfiable. The movement of the information cards is satisfiable.
Disruption criteria	The movement of one of the objects is glitchy, flickering, moving out of bounce or something unexpected happens while the player is moving one of the objects. The player cannot see the object after it is moved. The object appears in the wrong place after it is moved or placed on the board. The player cannot locate the object intended to move. The player loses the object without human fault, while holding it in the "drag and drop". The player cannot move or place intended objects on the board. The player disconnects.
Test case	The player checks the player model while in movement and after it is dropped on the board. The player checks the information cards after and while they are in movement. The player checks the placement of the blockades. The player checks the placements of the information center.
Alternative case	*None*
Test results	
Test comments	
Improvements needed	

Table 6.4: Black Box Test: Game movement

ID	**05**
Name	Game flow
Time schedule	Approximately 10 minutes.
Enviorment requirements	The test computer must have internet access. Regarding software, only a web browser is required.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.*
Goal	The test is to verify that the game handles turns correctly.
Actors	Player, Game Board, *Game Session, Server*
Test requirements	An expert has created a game. The player is inside a game session.
End requirements	The game turns work as they were intended. Turns do not change before the player gives the command. The player is able to use his or her complete turn, use the right amount of actions within a turn.
Disruption criteria	The game turn changes before the command is given. The game turn does not change when the command is given. The player is unable to use his or her complete turn, missing actions. The current player does not change, or the previous player can still move its player model. The player is able to use more than the given actions. An unexpected deviation occurs. The player disconnects.
Test case	The player uses the the intended actions within its turn. The player gives the command to change game turn.
Alternative case	The player tries to use more actions than there are in a game turn. The player tries to move the previous player model.
Test results	
Test comments	
Improvements needed	

Table 6.5: Black Box Test: Game flow

ID	**06**
Name	Game events
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. Regarding software, only a web browser is required.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.*
Goal	The test is to verify that the game events are as intended.
Actors	Player, Game Board, *Game Session, Server*
Test requirements	An expert has created a game. The player is inside a game session.
End requirements	The game timer resets upon the right time. The panic levels increase when the timer runs out. The panic level in the correct zones decreases when the information cards are used. The panic level increases in the correct zones when event cards are presented. The panic levels decrease when people are moved out of a zone. Effects from event cards are carried out as expected (i.e. the player must skip a turn or he loses 2 actions in his next turn).
Disruption criteria	The game timer does not reset upon the expected time. The panic levels do not increase when the time runs out. The panic levels do not decrease when information card effects are used. The panic level does not decrease when people are mowed out of a zone. The panic levels increase or decrease in the wrong zone when an effect is applied to the game. Effects from event cards are not recognized. The player disconnects.
Test case	The player checks that the timer is reset when the time runs out. The player checks that the panic level is increased in all eligible zones when the time runs out. The player checks that the panic levels decrease in the correct zones when an information card is used. The player checks that the panic levels are increased in the correct zones when an event card with this effect is viewed. The player checks if all the effects from the event cards are upheld.
Alternative case	*None*
Test results	
Test comments	
Improvements needed	

Table 6.6: Black Box Test: Game events

ID	**07**
Name	Testing memory overload
Time schedule	Approximately 2 hours.
Enviorment requirements	The test computer must have internet access. Regarding software, only a web browser is required.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.*
Goal	The test is to verify that the game does not run out of memory and can continue for an extended period of time.
Actors	Player, Game Board, *Game Session, Server*
Test requirements	An expert has created a game. The player is inside a game session.
End requirements	The game should not run out of memory.
Disruption criteria	The game runs out of memory. The game freezes. The player disconnects. Unexpected faults with the game.
Test case	The player(s) tries to use up as much memory as possible.
Alternative case	*None*
Test results	
Test comments	
Improvements needed	

Table 6.7: Black Box Test: Memory Overload

ID	**08**
Name	Multiple parallel sessions
Time schedule	Approximately 20 minutes
Enviorment requirements	The test computer must have internet access. Regarding software, only a web browser is required.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.*
Goal	The test is to verify that the server is capable of running several game sessions at the same time
Actors	Player, Game Board, *Game Session, Server*
Test requirements	An expert has created several games. A set of players are inside a game session. Another set of players are in another game session.
End requirements	All game sessions run smoothly, and the server is able to store the separate replays.
Disruption criteria	One session takes control of events in another session. The server is unable to handle the sessions, and freezes. A game session closes unexpectedly. The players disconnect. Unexpected faults with the server.
Test case	The players enter separate game sessions. The players then begin playing the game as normal. At predetermined times, the players perform actions at the same time in the separate sessions.
Alternative case	*None*
Test results	
Test comments	
Improvements needed	

Table 6.8: Black Box Test: Parallel sessions

ID	**09**
Name	Platform compatibility
Time schedule	Approximately 20 minutes.
Enviorment requirements	The test computer must have internet access. Regarding software, a range of different web browsers are required. - Internet Explorer - Mozilla Firefox - Google Chrome - Safari - Opera
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.*
Goal	The test is to verify that the game is compatible with different web browsers.
Actors	Player, Game Board, *Game Session, Server*
Test requirements	An expert has created a game. The player is inside a game session.
End requirements	The game works on all platforms.
Disruption criteria	The game does not load in one of the web browsers. The game only works partially in one or more browsers.
Test case	The player checks if the game can be loaded in all web browsers. The player goes through use cases 6 and 7 to check for functionality with all web browsers.
Alternative case	*None*
Test results	
Test comments	
Improvements needed	

Table 6.9: Black Box Test: Game Start

6.1.6 Usability tests

ID	**01**
Name	Player logging in
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.* The test person has to leave - unlikely ** Schedule a sufficient time slot beforehand.* ** If that proves too difficult, schedule a new meeting.*
Actors	Player, *Server*
Test requirements	As described in Use Case 01
Disruption criteria	As described in Use Case 01
Test results	Could not log in.
Test comments	Was not prioritized to implement users.
Improvements needed	

Table 6.10: Usability Test: Player logging in

ID	**02**
Name	Expert logging in
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.* The test person has to leave - unlikely ** Schedule a sufficient time slot beforehand.* ** If that proves too difficult, schedule a new meeting.*
Actors	Expert, *Server*
Test requirements	As described in Use Case 02
Disruption criteria	As described in Use Case 02
Test results	Could not log in.
Test comments	was not prioritized to implement users.
Improvements needed	

Table 6.11: Usability Test: Expert logging in

ID	**03**
Name	Game Setup
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. The expert is logged in
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.* The test person has to leave - unlikely ** Schedule a sufficient time slot beforehand.* ** If that proves too difficult, schedule a new meeting.*
Actors	Expert, *Server*
Test requirements	As described in Use Case 03
Disruption criteria	As described in Use Case 03
Test results	Successfully made a game template.
Test comments	Difficult for a first time user to know how to set up the game.
Improvements needed	Add descriptions on how to use the different features as well as make the shortcuts visible.

Table 6.12: Usability Test: Game setup

ID	**04**
Name	Watcher
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. The expert is logged in
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.* The test person has to leave - unlikely ** Schedule a sufficient time slot beforehand.* ** If that proves too difficult, schedule a new meeting.*
Actors	Expert, *Server*
Test requirements	As described in Use Case 04
Disruption criteria	As described in Use Case 04
Test results	Successfully joined a game session and watched the game.
Test comments	
Improvements needed	

Table 6.13: Usability Test: Watcher

ID	**05**
Name	Join Game
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. The expert has created a game session
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.* The test person has to leave - unlikely ** Schedule a sufficient time slot beforehand.* ** If that proves too difficult, schedule a new meeting.*
Actors	Player, *Game Session, Server*
Test requirements	As described in Use Case 05
Disruption criteria	As described in Use Case 05
Test results	Successfully joined a game.
Test comments	
Improvements needed	

Table 6.14: Usability Test: Join Game

ID	**06**
Name	Move game pieces
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. The player has joined the game
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.* The test person has to leave - unlikely ** Schedule a sufficient time slot beforehand.* ** If that proves too difficult, schedule a new meeting.*
Actors	Player, Game board, *Game session, Server*
Test requirements	As described in Use Case 06
Disruption criteria	As described in Use Case 06
Test results	Successfully moved game pieces.
Test comments	The game movements were all accordingly to the game rules
Improvements needed	

Table 6.15: Usability Test: Move game pieces

ID	**07**
Name	Use information cards
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. The player is ingame, and has information cards.
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.* The test person has to leave - unlikely ** Schedule a sufficient time slot beforehand.* ** If that proves too difficult, schedule a new meeting.*
Actors	Player, Game Board, *Game session, Server*
Test requirements	As described in Use Case 07
Disruption criteria	As described in Use Case 07
Test results	Successfully used information cards
Test comments	The effects of the information cards was done properly.
Improvements needed	

Table 6.16: Usability Test: Use information cards

ID	**08**
Name	Use player action
Time schedule	Approximately 10 minutes
Enviorment requirements	The test computer must have internet access. The player is ingame
Test risk analasys	Connection failure - Plausible ** Try to connect again. If the error persists, use another computer.* ** If switching computer does not resolve the connection failure, use another network.* Hardware fault - unlikely ** Use another computer within the group.* The test person has to leave - unlikely ** Schedule a sufficient time slot beforehand.* ** If that proves too difficult, schedule a new meeting.*
Actors	Player, Game Board, *Game session, Server*
Test requirements	As described in Use Case 08
Disruption criteria	As described in Use Case 08
Test results	All different user actions was successfully used.
Test comments	Add and remove roadblock/information center, decrease panic and move people all worked accordingly to the game rules.
Improvements needed	

Table 6.17: Usability Test: Use player action

Chapter 7

Final Product

7.1 Completed features

FR1 - Expert Interface This feature is largely finished. It includes the ability to design a city map, set variables(rules) for the game, create event- and information cards used during a game and set the number of players and where they start.

FR5 - Game Functionality This is first and foremost all the functionality of the original board game, as specified in the appendix, and is the most comprehensive requirement. All of these functions are completed, and works as intended.

7.2 Uncompleted features

!!Why are they incomplete?!! Will write when code is finished FR2 - Game Manager

FR3 - Player Profiles

FR4 - Replay

FR6 - Physical Interaction

7.3 Suggestions for further development

Extension of events as "quests": Currently, events are simple triggers for a list of effects that take effect immediately, and cannot be stopped or specifically countered except by playing the game normally. By extending this feature to include "tasks" to be completed, like "quests" in RPGs, the game would be come more dynamic and varied. These task would include a list of actions the players have to complete in order to get a reward, or avoid a penalty. If a fire is not put out in a certain amount of rounds, for example, you could lose points or the zone could become permanently panicked.

Effect variation: Further iteration on the "effect" function can increase the number of possible effects that can be made, both for events and information cards. The ability for effects to move players, destroy pathways, and manipulate road blocks could be added.

Sifteo cubes: Sifteo cubes could be used as controllers or displays for a number of functions. They could be used to represent zones, if there are enough of them, or as players that can be moved through moving the cubes.

Chapter 8

Project evaluation

8.1 Team

Task assignment

To begin with the task assignment did not work well. The different people in the group did not know what the others were doing, and therefore some tasks were done twice. After a team leader was chosen, the task assignment were much better and the group worked more efficiently.

Roles assignment

The roles assignment worked well. None of the members in the group were dissatisfied with their role, and each role did their part.

8.2 Process model

How did lean waterfall work out?

8.3 Development language

How was javascript to work with?

8.4 Final product

8.5 Customer evaluation

Will be added when coding is complete.

8.6 Lessons learned

Early in the project there were daily meetings. Later on the need for daily meetings was regarded as unnecessary as the team was able to work individually with the appointed tasks. Therefore, the number of meetings was reduced to three days a week (Mondays, Wednesdays and Fridays). On Tuesdays and Thursdays the team members worked at home. Mondays were used as a sort of "kick-off"-day for the week and were used to plan specific tasks that were to be done during the week. This increased the efficiency.

During week 9 the group member responsible for sending status reports to the customer and supervisor was on vacation. Because of that, no status report was sent that week. The lesson learned from this is that communication within the team is crucial for the completion of all tasks in time.

In the start of the project, there were little communication within the group. No standards were set for coding purposes and later on when the database names did not match the names in the program, there were some unnescessary problems. This was solved by setting code standards and better communication between the different parts of the team.

Chapter 9

Sifteo cubes

In this section, we will present the possibilities for using Sifteo Cubes as a client for the Don't Panic server.

9.1 Usage for Don't Panic

As an effort to making the game more engaging, the gestures of navigating the Sifteo cubes will give the players a wider experience. We discussed with the customer about how the cubes could be used in order to make a new client. By making a physical board with each node as a custom socket, the cubes can represent players. This way, there are more possibilities of displaying the data. For instance, you could limit information by only showing the number of people of the adjacent nodes. By doing this, you force the players to move around just to find out what zones are in panic.

For some of the other functionality, we came up with some ideas that we found feasible. For moving people from one zone to another, it coould be possible to have a socket or sensor in each node, and by placing a legal cube in a node, you would begin moving people "into" your cube incrementally. When you have the desired amount of people with you, the player would move the cube to the new zone, drop off the people, and the return to its original node.

Since the server only needs to be able to send and recieve JSON, we are able to make a client based on any language or any component as long as it is able to read and write JSON.

9.2 Implementation

In order for the cubes to communicate with the Node.js server, it will be necessary to make a program translating the Sifteo events into JSON, and also translate JSON from the server to the cubes to display. We will call this program *the middleman*. Since the cubes IDI got are of the first generation, the middleman needs to interpret the events from the cubes with the C# API before sending JSON to the server. The middleman must also be able to recieve response from the server, and update the cubes accordingly. This could be achieved with some event-driven code.

9.3 Recommended work

We reasearched what was already avaible for communication between the cubes and a PC. As it sands at time of writing, there is no support for doing this in the official API tha Sifteo provides. The communication between USB and the cubes are strictly limited to running the game on the

PC, and sending display data for the screens. November 21, 2012, it was posted a feedback on the official developer boards with the title "Allow I/O through USB connection", where Sifteo said that runtime I/O requires updates in the cubes firmware. [1]

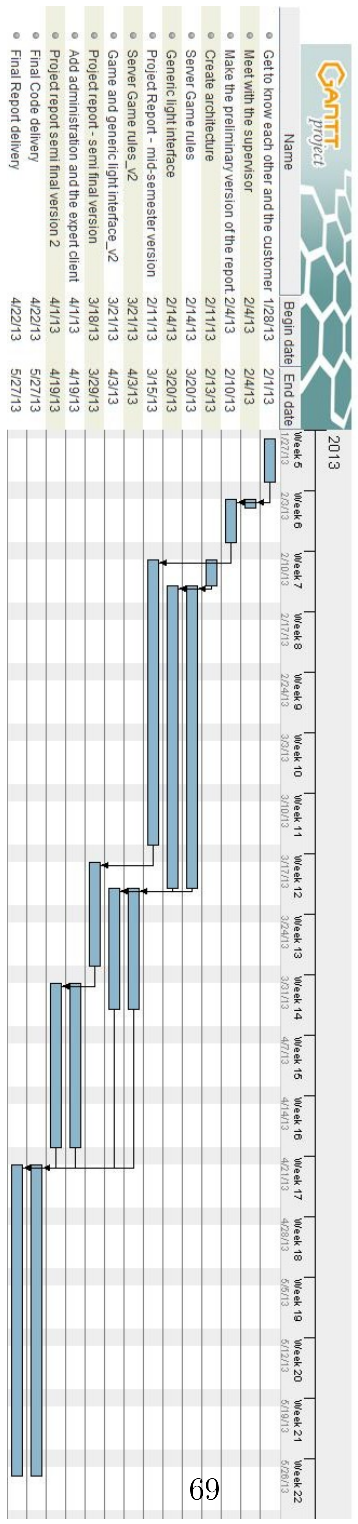
However, there have been examples of people going into the code Sifteo made for their first generation cubes, and intercepting the wireless data that is not avaible in the official API.[2] We have not been successful in acquiring any frameworks that does the job for us, so in order to make the Sifteo cubes able to communicate with the Don't Panic server, somebody needs to dive into the Sifteo code and find it themself. This is not a limitation by the server, but the API of the Sifteo cubes not including runtime I/O to PC. During time of writing, the site is down for maintenance, but the developer driven wiki for the first generation cubes might prove to be a good resoruce.[3]

It is difficult to estimate how long this will take to develop, seeing that we have not found a way for the cubes to communicate I/O in runtime. The time span required is dependant on prior knowledge to C# and wireless protocols, and how the events from the cubes are structured. We do not feel we have the competance to estimate the time required to make *the middleman* mantioned earlier.

[1]http://support.sifteo.com/entries/22448082-Allow-I-O-through-USB-connection
[2]http://thenxtstep.blogspot.no/2012/09/nxt-and-sifteo-cubes.html
[3]http://wiki.sifteocentral.com

Appendix A

Gannt Diagram



Name	Begin date	End date
Get to know each other and the customer	1/28/13	2/1/13
Meet with the supervisor	2/4/13	2/4/13
Make the preliminary version of the report.	2/4/13	2/10/13
Create architecture	2/11/13	2/13/13
Server Game rules	2/14/13	3/20/13
Generic light interface	2/14/13	3/20/13
Project Report - mid-semester version	2/11/13	3/15/13
Server Game rules_v2	3/21/13	4/3/13
Game and generic light interface_v2	3/21/13	4/3/13
Project report - semi final version	3/18/13	3/29/13
Add administration and the expert client	4/1/13	4/19/13
Project report semi final version 2	4/1/13	4/19/13
Final Code delivery	4/22/13	5/27/13
Final Report delivery	4/22/13	5/27/13

Figure A.1: 'Gannt diagram'

Appendix B

Don't Panic game rules

These are the rules of the physical board game, supplied by the customer.

Description of the game

Don't Panic is a cooperative game. You start the game as member of a "panic control team". During your day different potential panicking events will take place and you and your team will have to work together to calm people down and prevent the day from becoming the worst panic event humanity has ever seen. You and your teammates will assume a unique role within the team, with special abilities that will improve your team's chances, if applied wisely.

The aim of the game is to calm the situation down. In the game you can calm people by

- opening/blocking paths to help people to get off from frightening situations
- talking with them while other solutions are applied
- sharing information about how to manage the crisis
- moving people from one sector to another. But be careful! You only have a limited time to contain panic.

Every 5 minute the "panic level" will increase as a new "panic wave" arrives! If you and your team are unable to keep the panic contained and apply the necessary strategies to calm the people down in time, your city will become a mess and you and your team will lose the game.

A game turn

The play proceeds clockwise around the table with each player taking turns (in order) until the game ends. For each turn, the current player MUST

- take 4 ACTIONS
- draw 2 INFORMATION CARDS to add to his hand
- draw 2 EVENT CARDS and perform the corresponding actions on the board

Actions

A player gets 4 actions to spend on her turn. A given action may be performed more than once during a turn, so long as 1 action is spent for each instance. Each player's role will grant them special abilities that are unique to that player. Players may also pass if they have nothing else to do. Unused actions CANNOT be saved from turn to turn.

In each action the player can

- calm 5 people in a sector down
- move 5 people from one sector to another
- move through a maximum of 3 nodes
- create a barrier (is done with the help of another player)
- remove a barrier (is done with the help of another player)
- decide to spend all his actions in order to create an information center

Components

1 BOARD represents a real or virtual city. The board is divided into sectors and presents paths and key points. If all the paths in a sector are secured (i.e. blocked) the sector is secured (that is the panic will not augment in the sector at the next wave). However, only a non-secured path can let people pass from one sector to another.

8 PAWNS represent the players. The color of the pawn is linked to the draw role card. The pawn moves towards the sectors following key points. The player can act only on the sectors communicating with the key point.

1 TIMER calculates the next panic wave. When the timer rings each already panicked sector will be incremented by 5 panicked people (PL). In the non-panicked sectors nothing will happen. During the game panic waves happen initially every 10 minutes, then every 7 minutes and then every 5 minutes.

94 PLAYER CARDS :

- 6 ROLE CARDS. Each player assumes a specific role in the game which can do particular actions at low cost. The roles are detailed below.
- 48 EVENT CARDS. Event cards are, together with the Timer, the source of panic. Each round the player has to draw 2 event cards and apply their effects.
- 40 INFORMATION CARDS. Information cards diffuse information which is useful to manage panic. Playing an information card is at 0 cost (i.e. it is an additional action the player can take). Only one card per round can be played and only by the current player.

INFORMATION CARDS can be exchanged, but only if the two players are on the same key point. Each round the player has to draw 2 information cards but he can use them only from the next round. Take care! The number of information cards is limited! Once used, they cannot be put back into play.

5 INFORMATION CENTERS help lowering the panic. Once an information center has been constructed, the effects of the (draw)?? event cards on the adjacent zone are cut by half. However,

creating an information center is a highly costly action. To construct an information center a player needs to use all his actions for this round. A maximum of 5 information centers can be created on a board.

DISPLAYS WITH PANIC NUMBERS. Each sector of the game is equipped with a display to indicate the panic level (PL) in the sector. Chain Reactions: Once the panic number reaches quota 50 (that is 50 people panicked in the zone) the panic propagates to all nearby sectors (+5 panicked people).

10 BARRIERS help to block the spreading of panic throughout the sectors. To make a zone safe, all the paths have to be blocked. However, people cannot pass through blocked sectors. To create a barrier TWO players have to be on the same key point at the same time.

Sharing Information

Don't Panic is a collaborative game! Players are encouraged to openly discuss strategies during the game and share information. An information card can be used only once in a turn but it does not cost any action. Information can be "transferred" from one player to another. To transfer an information card from one player to another, the players have to be on the same key point. Only one card can be transferred at a time. The player who has the role of the Coordinator can transfer a card even if he is not on the same key point.

Roles

Each player is assigned a certain role, and there are six different roles:
COORDINATOR : Can share information even if he is not on the same key point
CROWD MANAGER: Can calm down 10 people in each sector instead of 5
DRIVER: Can move 10 people from one sector to another
OPERATION EXPERT: Can create/remove a barrier alone
VOLUNTEER: Can support one of the players (apart from the coordinator) duplicating their last action
PASSER BY: Can pass 1 information card to a player in an adjacent sector

Setting up the game

1. Place the board in the center of the table within easy reach of all the players. Put the displays on the board.

2. Shuffle the Role cards and deal 1 to each player. Each player takes their corresponding colored pawn. Place the pawn on the big matching colored key point. If the main key point is already taken, choose the small matching colored key point. Put excess Role cards and pawns (if any) back into the box.

3. Shuffle the INFORMATION CARD cards and deal them to the players face down. For a
   4 PLAYER GAME: 2 CARDS EACH.
   3 PLAYER GAME: 3 CARDS EACH.

2 PLAYER GAME: 4 CARDS EACH.
Place the remaining INFORMATION CARDS face down on the board in the appropriate sector.

4. Shuffle the EVENT CARDS and place them face down on the board in the appropriate sector.

5. Put the initial panic on the board: Each player draws 1 card from the EVENT CARDS and performs the corresponding action.

6. Turn the INFORMATION CARDS and communicate the possible actions you can perform.

7. Start the timer.

8. Play the game!

N.B. For a more challenging game session, switch the points 6 and 7.

Defeat and victory!

The game ends immediately in defeat for all players if all the map has a panic level higher than 50 panicked people. Players collectively win the game when the panic can no more spread because of the barriers, or if there is no panic on the board.

User manual

This is a high level description of how to use the program. When the program starts, the user will have three different choises:
1: Button - Play
1.1: A list of active game templates is displayed. Chose one of them, then the program will start a new game with the given template.
1.2: The user can now play the game by moving the players or using actions.
2: Button - Watch replay
2.1: A list of games that already has been played, is displayed. Chose one of them, then the program will enter the given replay.
2.2: Press the "next" button to show the next action.
3: Button - Expert interface
The expert interface is used to set up the game before it is played.
Button - Add effect, set wanted effect.
Button - Add role, set wanted role.
Button - Add player, set role and startnode for the player.
Button - Start!, starts the canvas so the user can create the map.
Mouse click on canvas - adds a new node to the map.
Button - Delete node, deletes the selected node from the map.
Button - Start node connection, starts a node connection in the selected zone and connects it to the next node that is selected.
Button - Add zone node, adds a node to a zone.
Button - Clear zone node, removes a node from a zone.
Button - Create zone, creates a zone if the nodes around it are zone nodes.
Button - Delete zone, deletes the selected zone.
Button - Set people, sets how many people in the selected zone.
Button - Set panic, sets the panic level in the selected zone.

Appendix C

Description of the Sifteo Cubes

Taken from the Sifteo FAQ

1 How do Sifteo Cubes work?

Sifteo Cubes is a hands-on interactive game system. They communicate wirelessly to respond to each other and being tilted, shaken, flipped, and pressed. The Sifteo Base stores your games and plays audio. You can play with 3-12 Sifteo cubes at once!
Sifteo Cubes use 1 AAA battery each. Sifteo Bases use 2 AAA batteries. Rechargeable nickel metal hydride (NiMH) batteries, standard alkaline, or lithium (non-rechargeable) batteries will each work. Batteries last for approximately 8 hours of gameplay.

2 Who are Sifteo Cubes for?

Sifteo Cubes are perfect for kids and kids at heart who geek out on brainy challenging games and cool new gadgets (recommended for ages 7 to adult). Our games can be played individually and with friends and family. Sifteo Cubes are sturdy and can withstand some abuse. Still, they are sophisticated electronic devices, and very rough handling (throwing, dunking in water, and other forms of cube torture) can break a cube permanently.

3 How do I play games on my cubes?

Sifteo Cubes come pre-installed with 4 games that work right out of the box. You can buy more games using the Sifteo desktop software (download here) and install them by connecting the Sifteo Base to your computer with the USB cable that comes with it. Sifteo games are 80-120 credits (about $8-12$) each. You can buy credits through the desktop software. View full tech specs and system requirements here .

List of Figures

List of Tables