SKNAT

-

Team A5

JANUARY 30, 2014
TEAM A5

Alexandre Portugal (1231727)
Bella Dunovska (1250580)
Martin Mihov (1229174)
Samuel Hill (1240663)
Tendaishe Nyamapfeni (1251414)

1. Introduction

1.1. Purpose

Through this SRS we aim to obtain a relatively clearly defined specification, from which we may create an initial prototype, and incrementally update our definitions in line with emerging requirements. We aim to make our objectives clear to any potential stakeholders, and to ensure that team members have a unified view of the project. This should be maintained throughout incremental development.

1.2. Scope

'Sknat' is a 'collaborative and competitive multi-player objective-based game' in which users will be able to compete in teams of 2v2. Players will be based on different terminals, with one host and the remainder will be clients. Each player has a graphical display and through real-time networking can interact with the other players and environment to complete objectives. The environment will be a 2 dimensional map with obstacles generated with a degree of randomness. The graphical interface, will allow the players to indicate movement/actions through keyboard and mouse input. Users will be able to host games and join games by specifying the IP address of the host.

The software in this version of specification will not provide any text-based communication.This may be later implemented, but as far as the gameplay is concerned, we do not consider it to be vital to collaborative effort especially given the fast-paced nature of the game. VoIP will therefore provide an alternative for player communication.

The goal of the game is to provide a game that is sufficiently complex to meet the requirements that we have been set for marking.

To quote some of the recommended features from the client:

- *'The game could/should have features such as:*
- *HCI issues considered well (e.g., it is easy to get started)*
- *A team of cooperating players*
- *A number of competing teams*
- *Different roles of the different players*
- *A graphical user interface,*
- *An auto player (AI component)*
- *Networking*
- *A random element'*

Game essentials:

- ✓ The game will have a menu interface, allowing the user to quickly navigate to a game in under 30 seconds.
- ✓ Networking will allow the players to collaborate and compete through unique perspectives
- ✓ The map structure will benefit players with presence of mind, so that obstacles can hinder and help players to gain a strategic advantage.
- ✓ With sufficient complexity in the gameplay such that collaboration and strategy contribute to success, there will be room for players to adapt and improve at 'Sknat' giving users the benefit of a sense of progression.
- ✓ The game will benefit users by providing an AI mode, which will allow users to progressively improve without the need to challenge others.

✓ We will provide enjoyment to the user over a prolonged period of time through varied, interesting and fast-paced gameplay.

1.3. Definitions, Acronyms, and Abbreviations

➢ **FPS**: frames per second
➢ **UI**: user interface
➢ **HCI**: human computer interaction
➢ **TCP**: transmission control protocol
➢ **UDP**: user datagram protocol
➢ **SVN**: subversion
➢ **IP**: internet protocol
➢ **GUI**: Graphical user interface
➢ **Port Scanning**: Probing client for open ports and potential vulnerabilities

1.4. Overview

The rest of the specification will elaborate on the game concept in the next section. Followed by detailed requirements engineering. The analysis models section will diagrammatically indicate functionality break-down and modularity. Within the appendix we will show some of the independent processes which led to us achieving our specification, including UI diagrams.

The SRS is organised into the aforementioned sections, see contents page.

2. General Description

This is a new self-contained product, built from scratch. It represent an arena based hack and slash game. Optimal playing requires four people in teams of two against each other. Therefore, an equipment of four computer and a network connection is necessary.

The game is fast - paced and particularly competitive.

Team members are dependent on each other and winning strategy requires active collaboration. This is highlighted by the different roles of each player and its functions.

It will make use of both keyboard and mouse controls.

It will implement a client-server model.

It is a real-time game which will require fast and reliable connection. After conducting some test and measuring the speed of package transfer we have decided to use UDP over TCP connection. Initially, implementation using TCP was planned based on its reliability. However, the test have shown that UDP provides connection which is sufficiently reliable and much faster than TCP.

The server will perform the majority of processing. This will largely be through exchanging co-ordinates and trajectory vectors.

For the UI the team will implement 2-dimensional graphics, displayed within a game frame. This will likely be done using java swing and java 2d, although some other option will be taken into consideration as appropriate during the project.

2.1. Product Perspective

The game is intended to be entertaining and suitable for a broad audience. The design and the approach adopted by the team will allow future expansions and improvements. The game is competitive but promotes collaborative play.

2.2. Product Functions

The game will include a main menu providing a user with several options such as connecting and starting a new game, options, quick and concise tutorial summarizing the rules.

The main objective of the game is reaching the highest possible amount of scores in a certain time period. The team with higher overall score wins. Opposite teams are supposed to collect points and to prevent the enemy time from doing so. Points can be obtain through collecting keys and unlocking main objective in the centre of the map. Keys appear at random location throughout the game.

Each user can choose from a set of characters with different capabilities affecting its play.

Collaboration is encouraged by the nature of the gameplay. One player is required to capture and maintain control of the objective. Therefore the other player's gameplay must change to protect their ally.

Each player has health points. They are lowered by his or her opponents through attacks. When a player reaches 0 health points he or she dies and restarts at the initial position.

Both teams start with equal health points and capabilities in terms of attacking.

2.3. User Characteristics

The system does not require any specific skills in order to be used. It is suitable for a wide range of customers. The rules are intuitive and straightforward and do not require any previous knowledge or experience. Therefore, the functionality can be easily understood and applied by all potential users.

The game is suitable for all age groups.

Short and descriptive manual is available for all users if certain functionality is unclear.

2.4. General Constraints

- The implementation language should be Java.
- There is a fixed deadline to deliver the system.
- The team is required to make use of SVN system.
- The academic staff put some constraints regarding the usage of external libraries or game engines.
- The team adopted an agile approach to the system's development.

2.5. Assumptions and Dependencies

- The game is supposed to be played by exactly four users.
- The game requires fast and reliable connection between the four players.
- Mouse and keyboard controls are both required for optimal play.

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

3.1.1.1. Graphical User Interface (GUI)

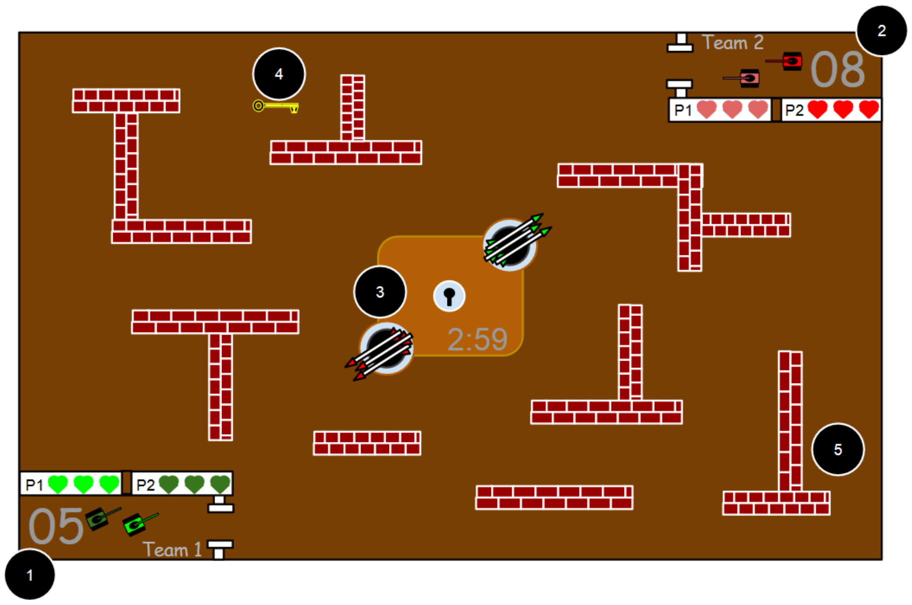


FIGURE 1: GAMEPLAY GUI

Object Numbe r	Name	Description
1	Team 1 Base	This section shows the base territory of team 1. This area is used as the point of re spawn for team 1 only. The base also displays some key information to the team; their score, team name, health of each player's tank.
2	Team 2 Base	Likewise the team 2 base territory functions in the same way as the other team's base territory. The only difference is positioning, team 1's base territory is located in the bottom-left corner of the screen and team 2's base territory is located in the top-left corner of the screen.
3	Objective Territory	This territory is part of the main objective of the game. The player has to collect a key which appears in random locations of the map and bring it here in order to gain points for the team.• A 15 page document (max size) 12pt size, A4 with proper margins in PDF format. • Based on IEEE Standard 830-1998 • A project schedule (e.g., in form of a Gantt chart) • Submit as a Team on Canvas by Friday 31 Jan 12:00 noon.
4	Primary Objective	This is the key which appears randomly on the map. As a player you have to take this key to the objective territory in order to get points. If a player carrying the key gets destroyed before reaching the objective territory; he will lose the key and it will appear in another location.
5	Walls/ Boundaries	These are objects which the tanks cannot pass through. (we are still contemplating the idea of destroyable wall/boundaries)

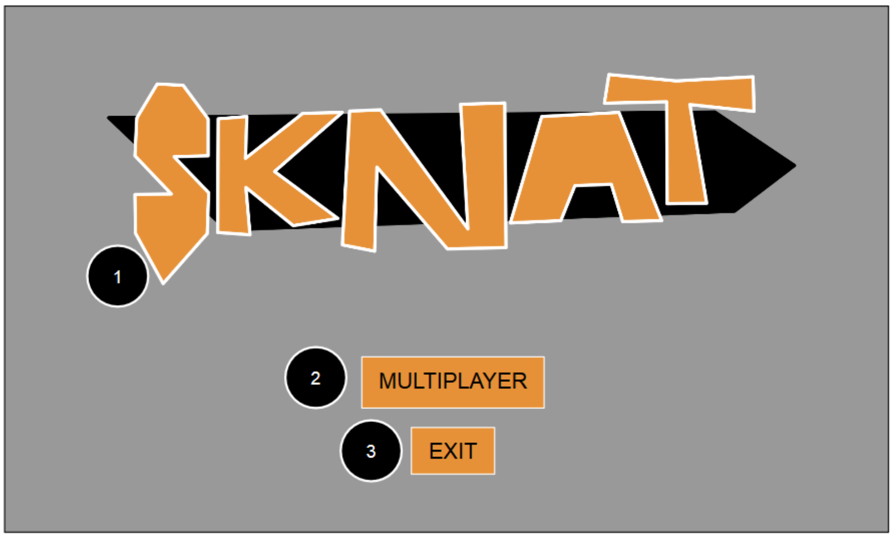


FIGURE 2: START MENU

Object Number	Name	Description
1	Logo	This is where the game logo will be.
2	Multiplayer button	This button will allow the user to go the multiplayer screen
3	Exit	This button will allow the player to exit game.

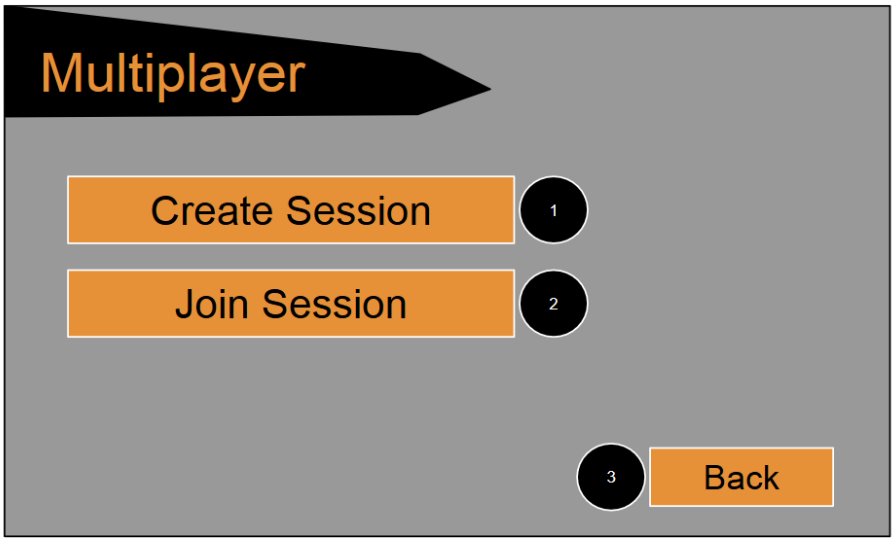


FIGURE 3: MULTIPLAYER

Object Number	Name	Description
1	Create Session Button	This button will allow the player to create a new game session.
2	Join Session Button	This button will allow the player to join an existing game session.
3	Back Button	This button will allow the player to go back to the Start Menu

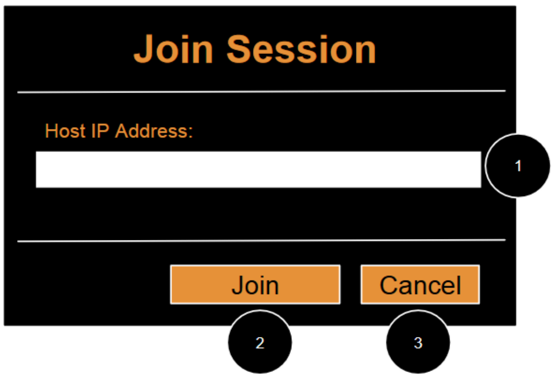
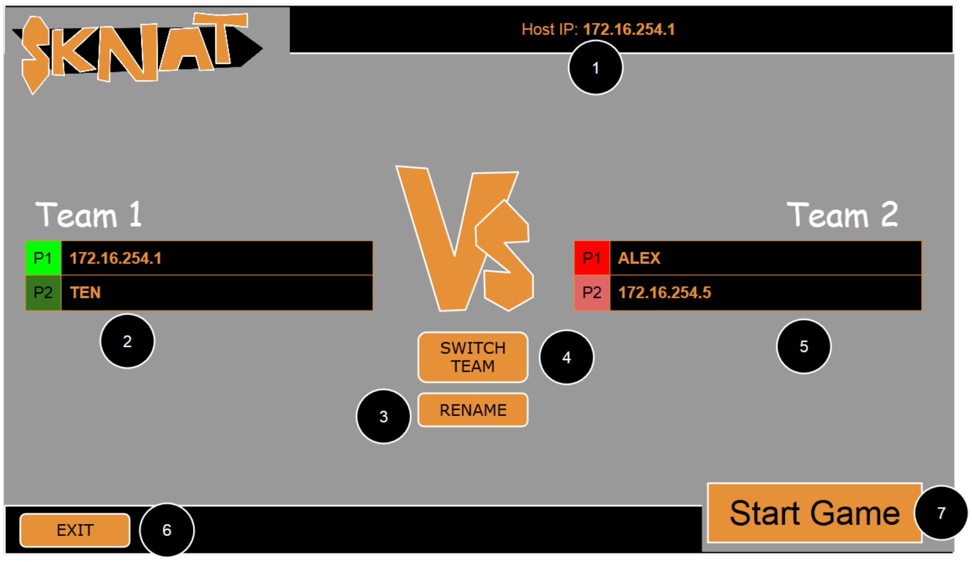


FIGURE 4: MULTIPLAYER - JOIN SESSION

Object Number	Name	Description
1	Host IP	This text box will take input from the player. The input has to be the IP address of the Host.
2	Join Button	This button will allow the player to join an existing session.
3	Cancel Button	This button will allow the player to cancel the operation



Object Number	Name	Description
1	Host IP Address	This just displays the Host IP address, making it easy to access and give to other players, so that they can join the game session.
2	Team 1	This will display the team members of team 1 and will also allow them to identify who is player 1 and who is player 2.
3	Team 2	This functions the same as team 1, except it is for team 2.

4	Switch Team Button	This button will allow players to switch team. (if there is a free slot)
5	Rename	This will players to identify themselves with nicknames instead of IP addresses.
6	Exit	This will allow a player to exit the session and go back to the previous menus.
7	Start Game	This button will allow the user to start the game.

3.1.2. Hardware Interfaces
Mouse and keyboard are the main hardware interfaces of this game.

3.1.3. Software Interfaces
None.

3.1.4. Communications Interfaces
It is likely that we will implement Voice over IP for users to aid and promote collaboration in fast-paced gameplay. Through the interface, the team may decide whether to address allies, or all players.

3.2. Functional Requirements

ID/Name	**Description**	**Details / Constraints**	**Use Case**
F1 - Tank Movement	A player should be able to move his tank on a 2d screen.	Tank movement is done by user input of w (forward movement), s (backward movement), a (left movement) and e (right movement). **Should not be able to move his tank to a position which there would be a wall in.**	UC1
F2 - Non player tanks visibility	All friendly and enemy tanks should be shown on the game screen at all times.	None.	UC2
F3- Health points	User should be able to see his tanks remaining life points as well as enemy/ friendly tanks life points.	When a tank is hit by a projectile, his life points should be updated to show his new life total.	UC3
F4- Key Spawn	A key should spawn at the start of the game or when one point is earned by a team.	Only one key should be active on the game at any time.	UC4
F5- Key Pickup	A spawned key should be able to be picked up by a tank.	Tank must be standing over the key to pick it up. Keys are automatically picked up. This, added to the killing of enemy tanks is one of the main competitive factors of our project.	UC5

F6- Key Mechanics	The key must be removed from the player when he reaches the centre of the map. If he keeps control of the centre for a pre-determined amount of time a point should be added to the team score.	This is where cooperation becomes	UC6
F7- Score	Score for each of the teams should be displayed at all times.	When a player scores, the score amount for that players team should be incremented.	UC7
F8 - Projectiles	On mouse click a projectile should be launched to the position of the mouse originating from where the player's tank was.	If the projectile hits an enemy tank, or a wall it should disappear. If it hit an enemy tank, its life points should be reduced.	UC8
F9 – Map Centre owned	The map's centre's ownership should change to a team when that team brings a key to it.	If at any time, when the map centre is controlled, there are more tanks of the team that does not own the centre on top of it, then the centre becomes without an owner. This is where cooperation becomes extremely important, as a team who is not in sync will have a very hard time getting points.	UC9

3.3. Use Cases

1. Name: Process movement from input	**ID:** UC1
Goal: To process input from the user into movement **Event**: A key (w, a, s or d) was clicked on the player's keyboard. **Overview**: This use case describes player movement. **References**: F1	
Initiator Actions:	**System Response:**
1 – Pressed keyboard key.	
	2 – Check if there isn't a wall in the direction the player wants to move to. *(If there is, Use Case ends here)*

	3 – Communicated movement to the server.
	4 - User's tank position was changed on the screen.

2. Name: Process tank position	**ID:** UC2
Goal: To update tank positions of other tanks **Event**: Message from the server host. **Overview**: This use case describes non player tank visibility on map. **References**: F2	
Initiator Actions:	**System Response:**
1 – Message from server	
	2 – Process Message.
	3 – Update position of the tanks in the screen.

3. Name: Health point update	**ID:** UC3
Goal: To update health points of tanks **Event**: Message from the server host. **Overview**: This use case describes health points display on map **References**: F3	
Initiator Actions:	**System Response:**
1 – Message from server	
	2 – Process Message.
	3 – Update health points of each tank on screen.

4. Name: Key Spawn	**ID:** UC4
Goal: To spawn new keys **Event**: No keys currently on map **Overview**: Key is spawned on the map. **References**: F4	

Initiator Actions:	System Response:
	Check if the map centre is not owned by any team **(If it is, Use Case ends here)**
	Spawn a new key

5. Name: Key Pickup	**ID:** UC5
Goal: Key disappears from map. **Event**: Tank moves over a key. **Overview**: Key is picked up by a tank. **References**: F5	
Initiator Actions:	**System Response:**
Tank moves over the key	
	Key disappears
	Tank is updated on screen to be shown holding a key.

6. Name: Key Use	**ID:** UC6
Goal: Key is used on map centre **Event**: Tank moves over the map centre while holding a key. **Overview**: Key is used up. **References**: F6	
Initiator Actions:	**System Response:**
Tank moves over the map centre	
	Key disappears
	Map centre is updated to show that a team now owns it.

Name: Score Update	**ID:** UC7
Goal: Score is changed **Event**: Centre of the map has been held for a pre-determined amount of time. **Overview**: Score is updated. **References**: F7	

Initiator Actions:	**System Response:**
Centre of the map has been held for a pre-determined amount of time.	
	Score of the team that holds the centre is updated.
	Map centre becomes ownerless.

Name: Create new projectile from input	**ID:** UC8
Goal: To process input from the user into a projectile. **Event**: A mouse click happened on the screen. **Overview**: This use case describes tank projectiles **References**: F8	
Initiator Actions:	**System Response:**
1 – Pressed mouse click.	
	2 – Create a new projectile with direction set to where the mouse is being pointed.

Name: Map centre control	**ID:** UC9
Goal: To update map centre control. **Event**: There are fewer tanks from the team that controls the centre on it than there are from the other team. **Overview**: This use case describes Map centre control. **References**: F9	
Initiator Actions:	**System Response:**
1 – There are fewer tanks from the team that controls the centre on it than there are from the other team.	
	Map centre becomes ownerless.

3.4. Classes/Objects

3.4.1. Game Class

The main class in which the game loop is running.

3.4.2. Menu Class

A class, providing basic functionalities for creating a menu screen.

3.4.3. Game Object

An abstract class, representing all objects in the game. Contains the coordinates and the dimensions of the object.

3.4.3.1. Movable

Abstract class that represents an object which can move. Has speed, direction and functions to move to object.

3.4.3.1.1. Character

An abstract class for the player's character. Has health points, projectile characteristics and methods for shooting and moving.

3.4.3.1.2. Projectile

A class for a projectile. It can be shot by a character and if it hits another character, it deals damage or heals the target.

3.4.3.2. Environment Object

An abstract class for representing objects on the map. The objects can be obstacles, collectables or just graphical object.

3.4.4. Map

A class that contains the current map, position of obstacles, characters and projectiles.

3.4.5. Network

A class that starts a new thread for all the network functionality. It initializes a UDP socket and allows sending and receiving information.

3.5. Non-Functional Requirements

3.5.1. Performance

The game should operate with at least 30 fps 90% of the time in terms of graphics rendering. There should be consistency between clients so that on 99% of trajectory impacts, both screens are synchronized (i.e. one player believes they have avoided missile while other player sees impact)

3.5.2. Reliability

The game should not crash ~100% of the time. The exceptions to this being OS Errors, and disconnection from the server. Packet loss should be handled by ensuring that the server updates the position of movables at 100Hz. This should ensure that packet loss and lack of synchronization are not obvious

3.5.3. Availability

At this stage of our project development there are no availability requirements.

3.5.4. Security

It is important that networking is implemented correctly, so that a given host may not be overwhelmed with requests. If there are vulnerabilities in the service running between the client and host, on their respective ports, there is a potential for these to be exploited, i.e. port scanning. The non-functional requirement should be expressed as, only explicitly relevant data to the game should be shared on the network.

3.5.5. Maintainability

Modularity and correct programming practices are a non-functional requirement that must be met. This will be measured in terms of the complexity (time) required to fix simple problems. Simple parametric changes should take no longer than 30 minutes to implement.

3.5.6. Portability
We shall independently verify that the game can operate on multiple platforms. We will do this on Linux and Windows. Cross-platform performance will be tested.

3.6. Inverse Requirements
There are no further constraints and inverse requirements.

3.7. Design Constraints

3.7.1. Standards Compliance
All language and images included in the game will comply with the current University of Birmingham guidelines for decency and equal opportunities.

3.7.2. Hardware Limitations
A keyboard and a mouse are required to play the game.

3.8. Logical Database Requirements
The game does not require a database. Instead it will contain objects with all the needed information and functionality.

3.9. Other Requirements
The game requires a connection to be established between all 4 terminals. Sockets allowing sending and receiving UDP packages must be opened.

4. Analysis Models
Please refer to Appendix 1 for UML diagram

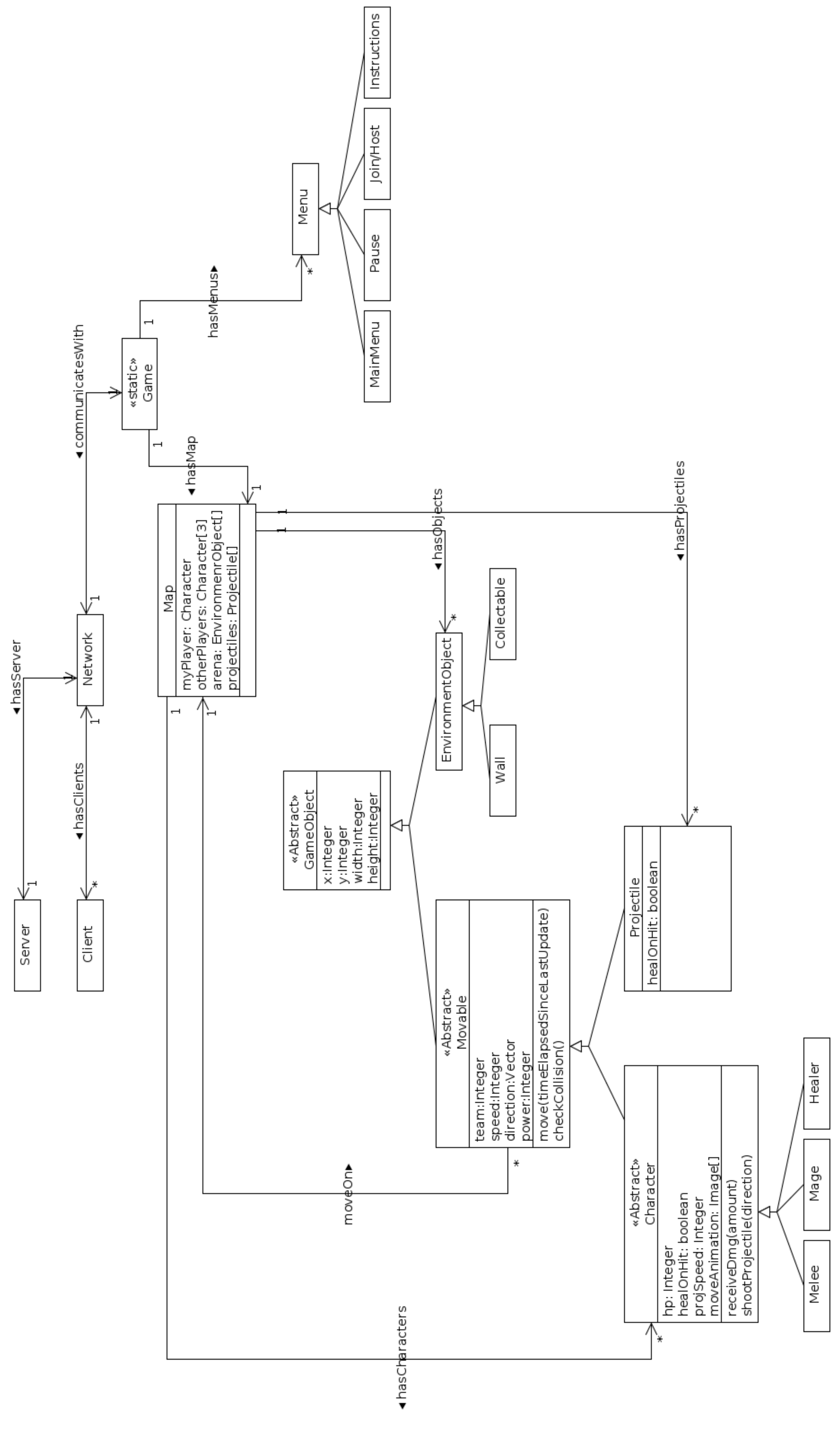
5. Change Management Process
If the need of a change is identified by any of the team members it will not be implemented prior to approval of the whole team.

The change will be proposed and explained on the following team meeting and if approved implementation plan will be specified. The impact of the change will be carefully assessed and all affected artefacts should be carefully investigated.

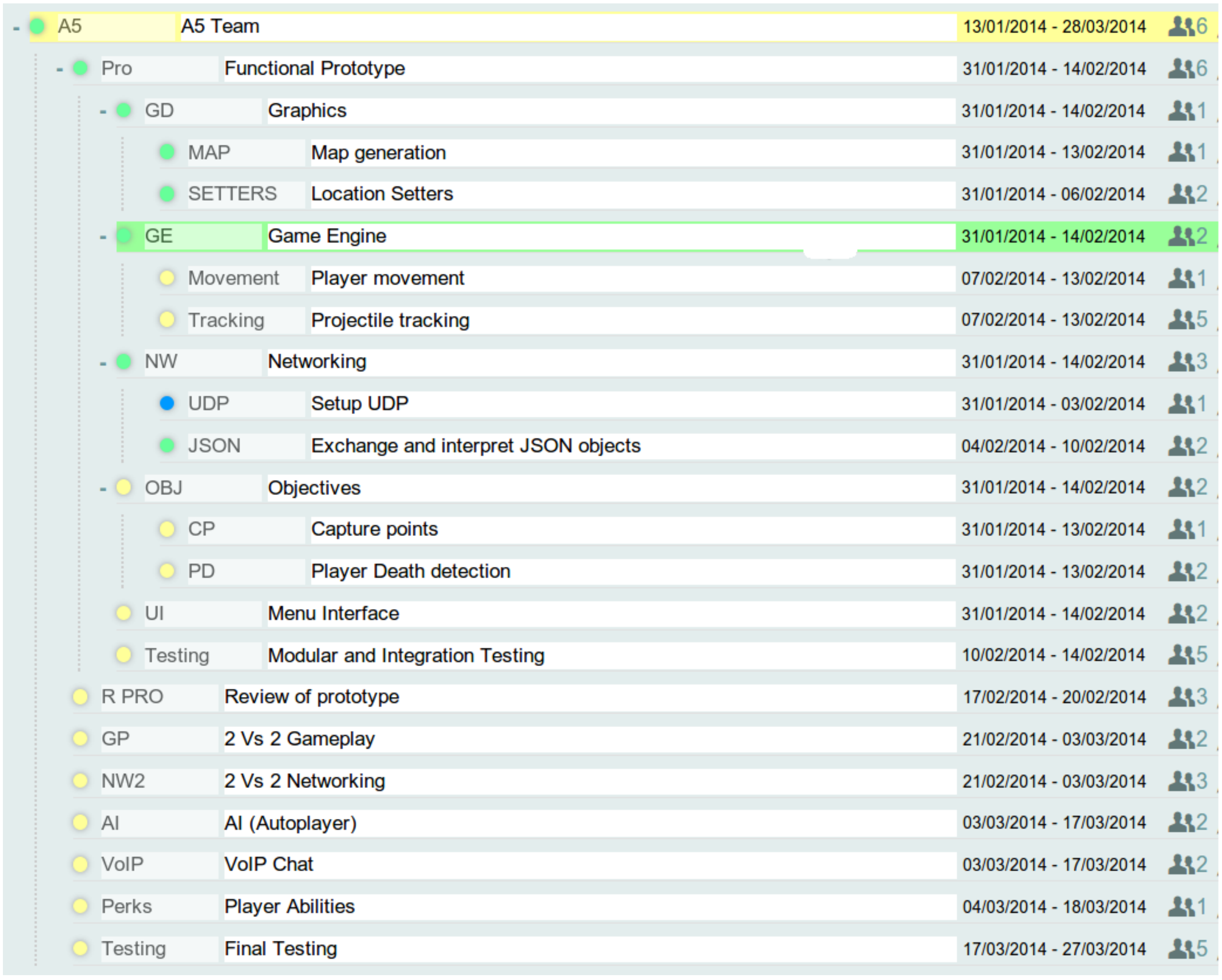
All team members should understand the change and its effects in depth. The appropriate roles will then be allocated amongst the team to successfully apply the change. The process will be documented and a log of changes will be kept in the common respiratory. Back-up copies will be kept for future reference, comparison or as a contingency plan in case of unsuccessful implementation. The project plan and the corresponding documentation will be updated accordingly.

6. Appendices

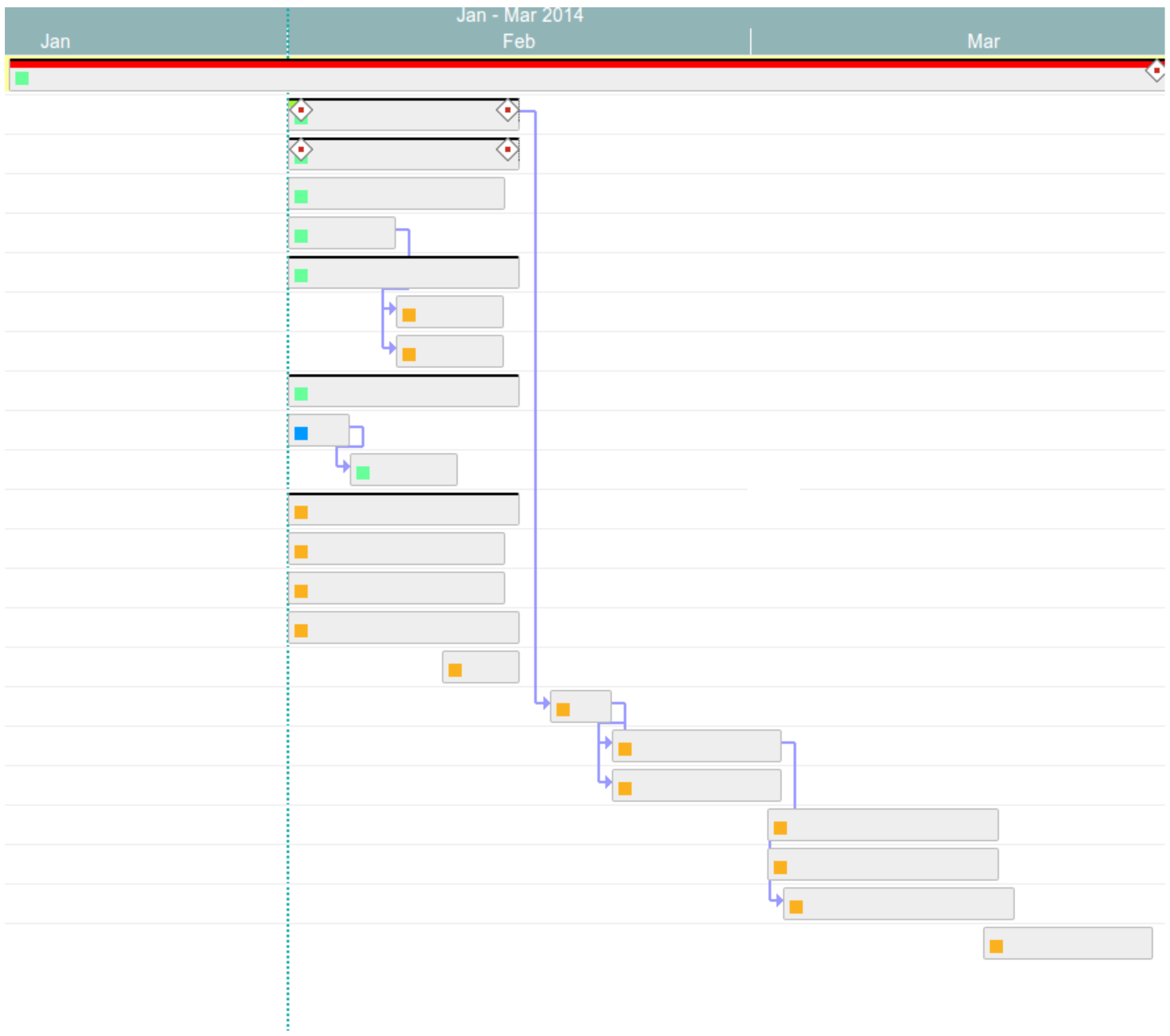
6.1. Appendix 1 - UML Diagram


Server
Client
Network
◂hasServer
◂hasClients
◂communicatesWith
«static» Game
hasMenus▸
Menu
MainMenu
Pause
Join/Host
Instructions
◂hasMap
Map
myPlayer: Character
otherPlayers: Character[3]
arena: EnvironmentObject[]
projectiles: Projectile[]
◂hasObjects
◂hasProjectiles
EnvironmentObject
Collectable
Wall
«Abstract» GameObject
x:Integer
y:Integer
width:Integer
height:Integer
«Abstract» Movable
team:Integer
speed:Integer
direction:Vector
power:Integer
move(timeElapsedSinceLastUpdate)
checkCollision()
moveOn▸
Projectile
healOnHit: boolean
«Abstract» Character
hp: Integer
healOnHit: boolean
projSpeed: Integer
moveAnimation: Image[]
receiveDmg(amount)
shootProjectile(direction)
Healer
Mage
Melee
◂hasCharacters
1
*

Work Breakdown Structure



- A5	A5 Team	13/01/2014 - 28/03/2014	6
- Pro	Functional Prototype	31/01/2014 - 14/02/2014	6
- GD	Graphics	31/01/2014 - 14/02/2014	1
MAP	Map generation	31/01/2014 - 13/02/2014	1
SETTERS	Location Setters	31/01/2014 - 06/02/2014	2
- GE	Game Engine	31/01/2014 - 14/02/2014	2
Movement	Player movement	07/02/2014 - 13/02/2014	1
Tracking	Projectile tracking	07/02/2014 - 13/02/2014	5
- NW	Networking	31/01/2014 - 14/02/2014	3
UDP	Setup UDP	31/01/2014 - 03/02/2014	1
JSON	Exchange and interpret JSON objects	04/02/2014 - 10/02/2014	2
- OBJ	Objectives	31/01/2014 - 14/02/2014	2
CP	Capture points	31/01/2014 - 13/02/2014	1
PD	Player Death detection	31/01/2014 - 13/02/2014	2
UI	Menu Interface	31/01/2014 - 14/02/2014	2
Testing	Modular and Integration Testing	10/02/2014 - 14/02/2014	5
R PRO	Review of prototype	17/02/2014 - 20/02/2014	3
GP	2 Vs 2 Gameplay	21/02/2014 - 03/03/2014	2
NW2	2 Vs 2 Networking	21/02/2014 - 03/03/2014	3
AI	AI (Autoplayer)	03/03/2014 - 17/03/2014	2
VoIP	VoIP Chat	03/03/2014 - 17/03/2014	2
Perks	Player Abilities	04/03/2014 - 18/03/2014	1
Testing	Final Testing	17/03/2014 - 27/03/2014	5



The chart is accessible by all stakeholders from the teamwork page. Tasks marked green are active, blue complete, and orange pending.