Requirements and Analysis Document

Johan Blomberg
johblomb@student.chalmers.se

Gustav Grännsjö
grannsjo@student.chalmers.se

Jonatan Gustafsson
jongust@student.chalmers.se

Robert Palm
palmro@student.chalmers.se

Version 5
28/5 2016
This version overrides all previous ones

Contents

1 Introduction

1.1 Purpose of application

The aim of this project is to realise a strategy game with gameplay inspired by games like *Duke* and *Fire Emblem*.

1.2 General characteristics of application

The application will be a desktop, standalone (not online), single player vs AI (Artificial Interface), with a graphical UI (User Interface) for Windows, Linux and OS X.
The game is played using six buttons, four to move the cursor around, one to select/confirm actions and one to deselect/abort actions.
The application will be turn based. A turn consists of the movement of the active player's units around the board (a player can chose to not move any units or to move them to the same space). When the active player moves a unit next to a unit of the other player the units can "fight" (a player can chose not to "fight"). Units can have different "weapons" which works in rock, paper, scissors "triangle", units also have different stats which impact different aspects of the game.
A turn ends when the human player either has no moves left or when they chose to. Turns then alternate between the human player and the AI until one of them wins. There is no time restriction to a turn (but the AI will finish as fast as possible). The game ends when a win condition is reached, for example when all enemy units are dead.

1.3 Scope of application

The game will lack any kind of narrative, focusing entirely on the gameplay. Sound design (background music and sound effects) will also be omitted. Player-versus-player modes will not be supported. The graphics of the game will be simple. The game will lack mouse input/control.

1.4 Objectives and success criteria of the project

It should be possible to play a full game, controlling and attacking multiple characters in combat against computer-controlled units, and winning/losing depending on certain conditions as defined by the game.

1.5 Definitions, acronyms and abbreviations

- Turn – The period of time in which the computer and player performs their available actions before passing control to the other.

- Unit – A character controlled either by the player or the computer. A unit can move and attack. The game ends when the player or computer is out of usable units.
- Stats – Statistics. The health, attack, speed/movement, weapon(-type) values of a unit.
- Game board – The board the game is played upon. Moving outside the boundaries of the board is impossible. Also referred to as a "map".
- Move, Movement – The action of a unit's position on the board.
- Weapons – Units can have different types of weapons, which defines how well they perform in combat against other units.
- Tiles – The game board consists of a square-grid of tiles.
- Terrain – Tiles can have different kinds of terrain, which can impact combat and movement.
- Combat – When two units fight. Meaning they remove their respective attack from each other's health.
- Cursor – The cursor is the visual representation of where the action marker is.
- Marked – The space where the cursor is.
- Selected – When a unit is marked and then the action button is pressed.
- Move Buttons – Up, down, left and right. Moves the cursor.
- Action Button – Selects and confirms actions/units.
- Abort Button – Deselects and abort actions/units.
- Scenario – A combination of a specific game board along with a corresponding set of units.

2 Requirements

2.1 Functional requirements

The player should be able to:

1. Select a map/scenario to play.
2. See the different stat values of the units on the board.
3. Select and move his units (As long as they haven't attacked yet).
4. Attack enemies with his units.
5. End their turn.

2.2 Non functional requirements

2.2.1 Usability

Usability should focus on a target audience that is all ready familiar with strategy games and its standard paradigms.

2.2.2 Reliability

N/A

2.2.3 Performance

There should be no memory leaks and the game should not drive sluggish as its a lightweight class application.

2.2.4 Supportability

The application should be easily detachable from its view and it is therefore of outermost importance that the model is not dependant on a specific view. See MVC.

2.2.5 Implementation

The application should be as platform independent as possible and should therefore be written in Java.

2.2.6 Packaging and implementation

The program should be delivered as a runnable .jar, possibility to extend into a .zip with a readme and rulebook is possible for future releases.

2.2.7 Legal

There are some legal issues concerning the graphics used in the game, but that isn't covered here.

2.3 Application models

2.3.1 Use case model

See appendix for UML diagram and detailed descriptions of use cases.

2.3.2 Use cases priority

High priority:

- Unit movement
- Unit Combat

- Cursor movement
- Unit selection
- Turn Cycle
- Unit health & dying

Mid priority:

- Display attack range of unit
- Display movement range of unit
- Experience gain
- Tooltips
- Status screens

Low priority:

- Consumable items
- Map selection
- Shopping
- Camera panning
- Save/load games
- Options
- Pre-battle preparations
- Tutorial

2.3.3 Domain model

See appendix for domain model.

2.3.4 User interface

The GUI of the application will be static (non customisable but scalable) and is supposed to use existing conventions from similar games (see references) for handheld devices. A marker is controlled by the arrow-keys and when the yes/no-keys are pressed certain actions take place. What action takes place is contextual to the state of the game.

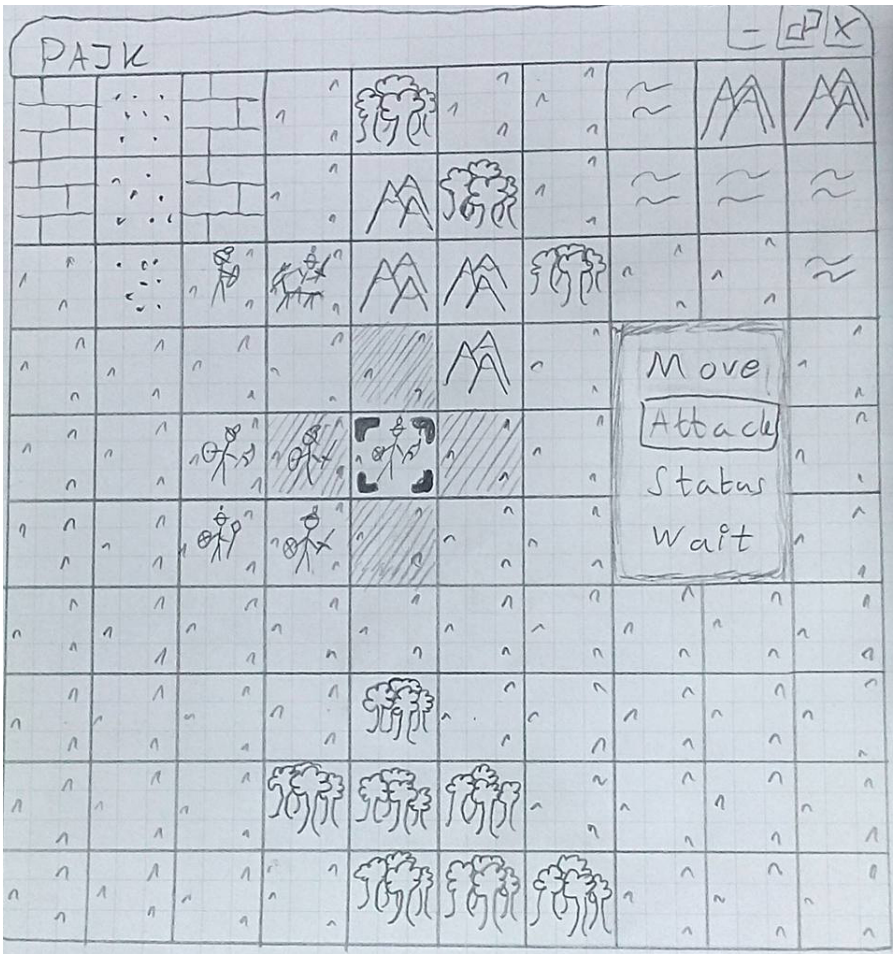
2.3.5 References

Nintendo. (2016). *Fire Emblem Gameplay*. Hämtad från
www.nintendo.com/games/detail/2ftO9Q5JYqfowOmxJcf_5cpDcBQpv0wV

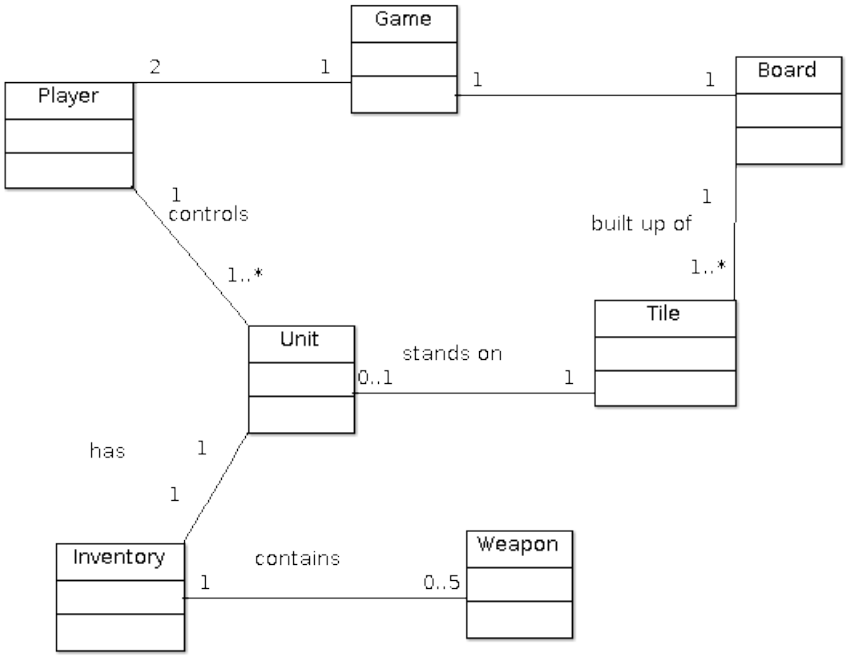
Catalyst Games. (2012). *The Duke Rules*. Hämtad från
www.catalystgamelabs.com/download/The%20Duke%20Rulebook.pdf

Appendix

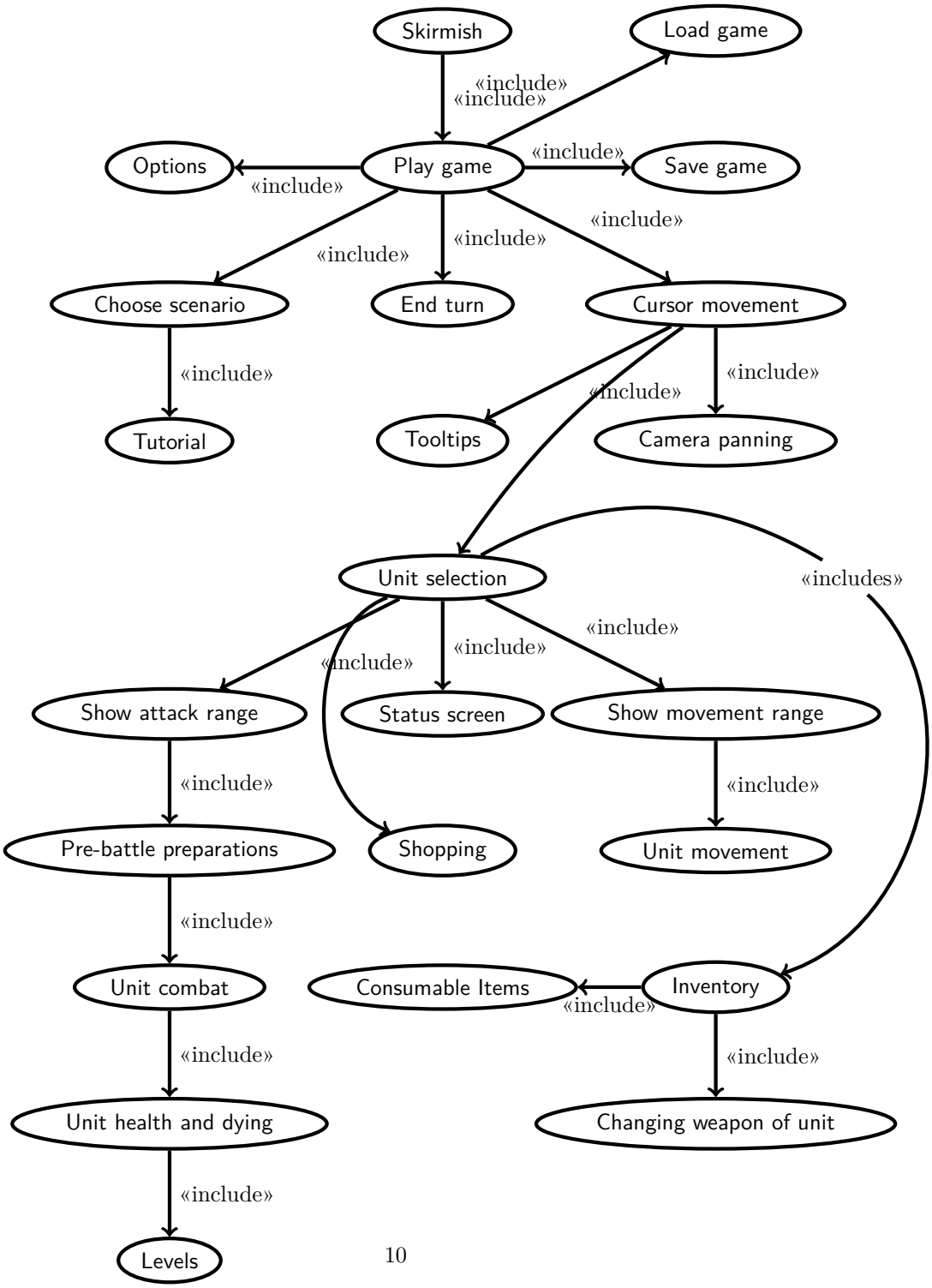
GUI



Domain model



Use case model



Use cases

1. Unit movement

Summary: How the player will move their units on the board. UC Attack can proceed after this UC.
Priority: High
Extends: –
Includes: Cursor Movement and Unit Selection
Participators: Player and Computer

	Actor	System
1	Player selects unit (see Cursor Movement UC and Unit Selection UC)	
2		Highlight squares available to move to
3	Player selects square to move to	
4		Units moves to selected square

2. Unit combat

Summary: The player selects an enemy unit to attack with one of their units after it has finished moving.
Priority: High
Extends: –
Includes: Cursor Movement and Unit Selection
Participators: Player and Computer

	Actor	System
1		Finish moving unit
2		Highlight squares adjacent to unit
3	Select from one of the highlighted squares	
4		Perform attack between selected enemy and player unit
5		Both units gain experience or are removed

3. Cursor Movement

Summary: When the player moves the cursor on the game board.
Priority: High
Extends: –
Includes: Pressing buttons
Participators: Player and Computer

	Actor	System
1	Player uses arrow keys	
2		Highlights the selected tile

4. Unit Selection

Summary: When the player wants to highlight a unit in order to perform a specific action with it
Priority: High
Extends: –
Includes: Pressing buttons, Cursor Movement
Participators: Player and Computer

	Actor	System
1	Player uses action button	
2		Display options menu

5. Turn cycle

Summary: Once the player is satisfied with the commands they have issued, or have used all available commands, their turn ends and the computer's turn begins.
Priority: High
Extends: –
Includes: –
Participators: Player and Computer

	Actor	System
1	Player finishes all their available actions or manually chooses to end their turn	
2		Player turn ends, enemy turn begins
3		Enemies finish all their actions
4		Enemy turn ends, player's turn begins

6. Unit health and dying

Summary: When a unit has been dropped to zero health by an attack from an enemy unit.
Priority: High
Extends: –
Includes: Pressing buttons
Participators: Player or AI and Computer

	Actor	System
1	Active unit deals damage >= current health left of target unit	
2		Unit is removed from play
3		Active unit gains experience points see UC Experience
4		The turn continues or ends.

7. Experience gain

Summary: Summary: When a unit has ended combat it gains experience based on the unit's level and the enemy's level.
Priority: Mid
Extends: –
Includes: –
Participators: Player and Computer

	Actor	System
1	Finished combat with an enemy units while still alive	
2		Both users gain experience points, is shown in an experience bar and as a number.
3		Turn continues

8. Tooltips

Summary: When the cursor is hovering over a tile, information is shown about the unit or the terrain.
Priority: Mid
Extends: –
Includes: Pressing buttons
Participators: Player and Computer

	Actor	System
1	Cursor is placed on tile (see UC Cursor Movement)	
2		Shows information about terrain. If unit is standing on terrain shows information about unit.

9. Status screens

Summary: A screen that can be brought up by the player to view detailed info about any unit.
Priority: Mid
Extends: –

Includes: Pressing buttons
Participators: Player and Computer

	Actor	System
1	Player presses "info" button while cursor is on a unit.	
2		Displays a screen containing detailed info and statistics about the unit
3	Player presses "back" button	
4		Displays the game board again

10. Select and Use Consumable Items

Summary: A player can select a unit and choose to use a consumable item. The item will affect the unit by for example restoring its health.
Priority: Low
Extends: –
Includes: Unit Selection
Participators: Player and Computer

	Actor	System
1	Selects unit (see UC Unit selection) and chooses the Items option	
2		List of items player currently carries appears
3	Selects item in list (e.g. health potion)	
4		Unit is affected by the item (e.g. health increases)

11. Map selection

Summary: The player can select between multiple maps at the start of game, and choose one to play on.
Priority: Low
Extends: –
Includes: –
Participators: Player and Computer

	Actor	System
1	Selects option "maps" at start of game	
2		Gridview of all available maps appears
3	Selects a map	
4		Game starts with selected map.

12. Shopping

Summary: The player can acquire new items in exchange for accumulated currency.
Priority: Low
Extends: –
Includes: –
Participators: Player

	Actor	System
1	Moves a unit to a tile that contains a shop and select the option to use the shop from the unit menu	
2		Displays available items to buy along with their cost
3	Selects one of the items	
4.1		If the player has sufficient amount of money, move the item to their inventory
4.2		If the player lacks the needed amount of money, a message is displayed and nothing else happens

13. Camera panning

Summary: For boards large enough to not fit on the screen, the player can adjust their view by panning the "camera" around.
Priority: Low
Extends: –
Includes: –
Participators: Player and Computer

	Actor	System
1	Player moves cursor towards the edge of the screen	
2		Pans the map into view from outside the edge the cursor is close to.

14. Saving and loading games

Summary: The player can suspend a game and resume it at a later time
Priority: Low
Extends: –
Includes: –

Participators: Player

	Actor	System
1	The player chooses a menu option to suspend game	
2		Saves the state and positions of all units on the map and stores the data
3	Player begins a new session and chooses to resume a suspended game	
4		The stored data is loaded and the map is restored to the state it was before the suspension.

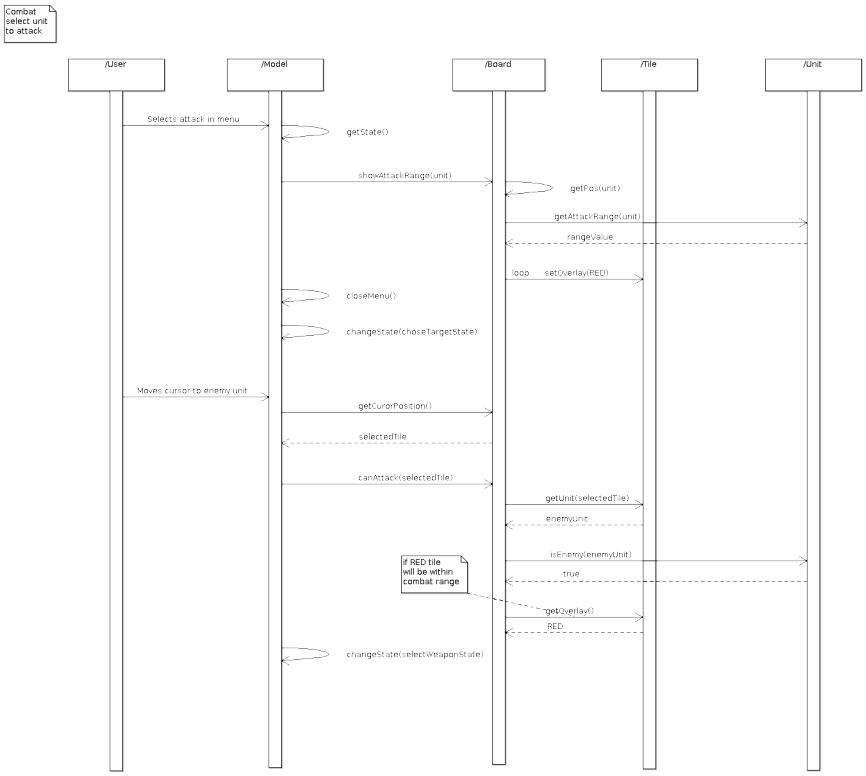
15. Options

Summary: The player can adjust various parameters of the game experience, such as sound and game language.
Priority: Low
Extends: –
Includes: –
Participators: Player

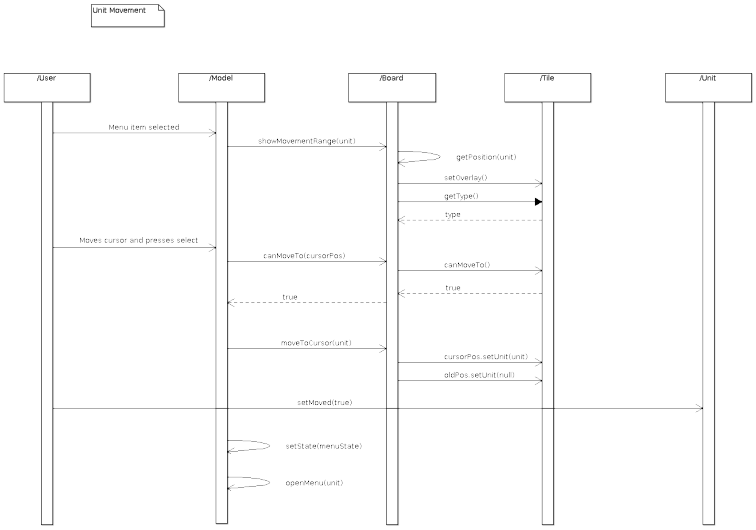
	Actor	System
1	Selects ”Options” from main menu	
2		Displays the settings that the user can change
3		
4	Adjusts desired parameters and selects ”Done”	
5		Applies the changed settings while returning to the main menu

Sequence Diagrams

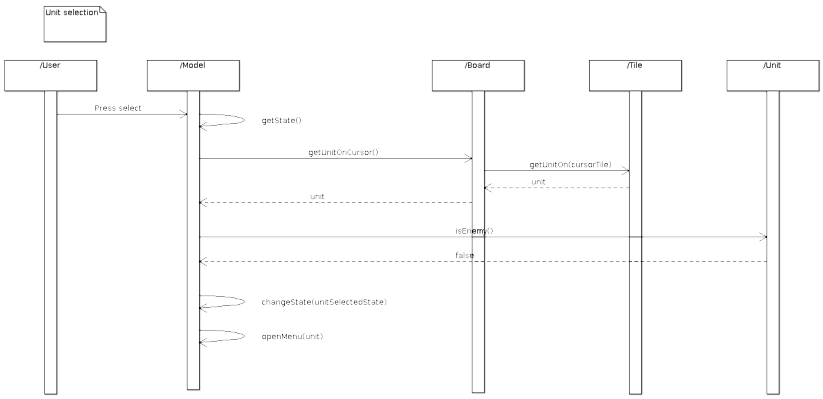
1. Combat



2. Unit Movement



3. Unit Selection



4. Cursor movement