SETTLE AND DESTROY - (SAD)

Group 13
Jonas Wikberg
Christofer Hjalmarsson
Daniel Westerberg
Saul Amram
André Sikborn Erixon

1. Preface

1.1. *Define expected readership of the document*

The requirements document is written to be read by the stakeholders of the project.

Role in the development project	**Reason for using the requirements document**
Development project leader	• Scope of the project, divide the project in phases • Obtain agreements from project sponsors, development manager on the scope, and schedule for phases • Track development progress
Requirements analyst	• Test the application against the requirements
Development team member	• Design and code the application
Maintenance team member	• Learn the application to take over the production support from the development team • Support users in production
Development manager	• Understand planned phases and coordinate with the development of other applications • Track development progress
Project sponsors	• Provide the motivation for the project • Sign off on the planed phases and their scope
Business experts (often designated by the project sponsor)	• Confirm that the requirements reflect the business needs. • Sign off on the planned phases and their scope

1.2. *Describe version history.*

Version	Rationale	A summary of changes	Date	Author(s)
1.0	First version		2008-01-04	André S. Erixon Saul Amram Christofer Hjalmarsson Daniel Westerberg Jonas Wikberg

2. Introduction

2.1. *Gameplay information*

This is a real-time strategy war-game with opportunities to play both multiplayer and a special training ground. The first thing a player has to do is to choose an alias to play with. Then you can choose to start your own game, or join another player’s game that’s waiting for more players. The first thing you have to do when you start your own game is to enter the number of players that will participate. In theory the number of players in a game is unlimited, but since all the slots have to be filled a maximum of eight is a rule of thumb. A player that chooses to join an already existing game has to enter the ip-address of the host to be able to connect to the game.

When players join a game they must press the "ready" button to inform the host that they are ready to start. The host can not start the game until everyone joined are ready. The game starts, and due to its nature players will die as time passes. When this happens you can either choose to stay in the game and observe, or simply leave and join another game. The game ends when all players except one are dead. This one survivor is therefore named the winner. As soon as the game has a winner, people will disconnect and then either decide whether they want to play more or perhaps get back to things they did before they started playing.

2.2. Need for the system and information about its users

2.2.1. Society need for the system

We live in a society where people spend most of their time working or performing other various duties. Long term, everyday work can be tedious and monotonous and may eventually lead to people ending up in a burnt out state. One way to reduce this probability is to supply means of good entertainment. For some people, sitting in front of the TV watching shows can be an effective stress-reducing factor, whilst others may prefer a more controllable form of media, - computer games. The game we supply, try to do just that by giving its users a moment of enjoyment that can make their day into a day worth remembering.

2.2.2. Who are the users

The probable user is males between twelve to thirty years old. The common characteristic between two unique users of our game will probably be having a high need for entertainment and an interest in casual computer gaming, computers in general or strategy games. Another important attribute to be attracted to the game, would be having a high need to beat friends in competitions using their own skills, not wanting to rely on solely luck.

The reason for not also including females to the probable users is that males tend to play computer games more than females. The age range has been set as it is because a kid of younger age than twelve may not have the somewhat developed strategical mind that makes the game more appealing.

2.2.3. What problems does the system solve for its users

The aim of our game and the incentive for a user to play our game is to satisfy the users need for entertainment. Users in the age range we are aiming for, tend to have quite long and tedious working days. The importance of escaping the stress of work/school and the monotonous everyday life should not be neglected. The game will provide ways to be entertained for a moment, letting the mind relax from chores and assignments. This will make days overall, enjoyable and more endurable.

The sub goal of our game is that it will provide strategical challenges to those interested in winning strategical games. Letting users compete against each other will make each game session unique and promote users to develop different tactics depending on their opponent's actions, - to win the game.

2.3. System functions and integration with other systems

2.3.1. System functions

The system will provide a main menu upon launch of the game application. The main menu will supply the user with a choice about what kind of game mode they want to play.

A single-player training mode is available where users can work on their gaming skills without having to feel any time limit or pressure from opponents. No internet connection is needed for this mode.

It is possible to host a multiplayer game for up to a total of six players. The host is able to set the game settings for the upcoming game according to his preferences. Internet connection is required.

Users can join an already hosted multiplayer game by connecting to the host’s IP. Both Internet connection and knowledge about the host’s IP number is required to be able to join a game.

2.3.2. The context/environment in which the system is to be used

The game could be played in most environments on a computer where you have access to either an internet connection or a local network for example in school, at home or at your work. If the game host has an internet connection, both players within and outside the host’s local network can participate. It is possible to use training mode without network connection at all. However, the game will require intense attention for a period of time (10-30 minutes) why the user must have a certain amount of time to spend playing the game. You can't play multiplayer mode on one computer (split screen).

2.3.3. Computer requirements and integration with other systems

The game does not require any high performing computers (memory, graphics card, processors etc.) and computer as old as 4-7 years will be able to run it (No explicit demands will be stated since it's very unlikely to be a factor). The game will be written in java and it will be necessary to have JRE 1.5 (Java Runtime Environment) installed which can be downloaded for free and it will thereby be able to be played on all different operative system where you can install JRE. It will only be tested and guaranteed supported on JRE with Windows XP/VISTA (differences with JRE on different platforms can occur even if it shouldn't).

2.4. How the system fits in the business or strategic objectives of the commissioning organization

2.4.1. Long-lasting appeal

It the society, the need for entertainment will never change. The latest fashion and trends may change with the wind but people will always need something entertaining to do with their spare time. Thus our game’s purpose can never become out-of-date and will always be a valid purpose.

A game that is too repetitive will quickly become boring and its players will stop playing it. For a game to last for a longer time it must show a good variety between game sessions. The possibility to face human opponents in the multiplayer mode makes each game round unique and different from the last. Because of this variety the entertainment factor will stay high even after many games played.

This is what makes our game having a long-lasting appeal.

2.4.2. Small maintenance costs

Expensive maintenance can be devastating to company economics. The game we supply will have low maintenance costs because of several reasons.

- No large ongoing support to its users is needed because of the extensive help document.
- No major updates are going to be developed.
- No costs for having online servers handling multiplayer games.

2.4.3. Future development

If necessary it is very possible to develop the system further.

A possible business solution would be to open up an Internet portal where players meet, chat and have a graphical view of possible games to join. This would also mean supplying servers that can support this multiplayer interaction.

A tournament ranking system where people can gain/lose rank points depending on their achievements could be implemented. This would make it easier to make the users spend money on the system by giving some kind of advantage, extra game functions or members only-tournaments which require real money.

Commercial could then be introduced both in-game and on the Internet portal to increase profits.

3. Glossary

API = Application Programming Interface. An application programming interface (API) is a source code interface that an operating system or library provides to support requests for services to be made of it by computer programs.

AWT = Abstract Window Toolkit. AWT is Java's original platform-independent windowing, graphics, and user-interface widget toolkit.

Client = A user that has not started a game, but has joined another user's game.

Game controls = See I/O

Game round = A game round is a description of a complete gaming sequence. The complete game round starts at the initialization of the game when the player/players gets control of his/their game situation through the game controls and ends as the one of two the last players in a multiplayer game kills is opponent. Generally a game that is terminated in other ways (by exiting the game) or exceptional ways (system collapse) could be referred to as a game round.

Host = A user that has initiated a game or application thus making it possible for other users to join the same game by connecting.

I/O = In computing, input/output, or I/O, refers to the communication between an information processing system (such as a computer), and the outside world - possibly a human, or another information processing system. Inputs are the signals or data received by the system, and outputs are the signals or data sent from it. The term can also be used as part of an action; to "perform I/O" is to perform an input or output operation. I/O devices are used by a person (or other system) to communicate with a computer. For instance, keyboards and mice are considered input devices of a computer, while monitors and printers are considered output devices of a computer. Devices for communication between computers, such as modems and network cards, typically serve for both input and output.

Java 2D = The Java 2D API is a set of classes for advanced 2D graphics and imaging, encompassing line art, text, and images in a single comprehensive model. The API provides extensive support for image compositing and alpha channel images, a set of classes to provide accurate colour space definition and conversion, and a rich set of display-oriented imaging operators.

Java NIO = New I/O, usually called NIO, is a collection of Java programming language APIs that offer features for intensive I/O operations.

Java Virtual Environment (JVE) = Drivers needed for the operative system that explains how applications made in Java shall be executed. Without proper JVE, java applications can not run on the computer.

LWJGL = LighWeight Java Game Library. LWJGL provides access to high performance cross-platform libraries such as OpenGL (Open Graphics Library) and OpenAL (Open Audio Library). Additionally LWJGL provides access to controllers such as Gamepads, Steering wheel and Joysticks.

MINA = Apache MINA is a network application framework which helps users develop high performance and high scalability network applications easily. It provides an abstract event-driven asynchronous API over various transports such as TCP/IP and UDP/IP via Java NIO.

OpenGL = Open Graphics Library. An environment for developing portable, interactive 2D and 3D graphics applications.

Operative system = An operative system is usually a program that handles the execution of others programs and at the same time giving the user a graphical interface, making it easier to interact with the system and different programs.

Protocol = In computing, a protocol is a convention or standard that controls or enables the connection, communication, and data transfer between two computing endpoints. In its simplest form, a protocol can be defined as the rules governing the syntax, semantics, and synchronization of communication.

Swing = Swing is a widget toolkit for Java.

TCP = The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite. TCP provides reliable, in- order delivery of a stream of bytes, making it suitable for applications like file transfer and e-mail. It is so important in the Internet protocol suite that sometimes the entire suite is referred to as "the TCP/IP protocol suite."

UDP = User Datagram Protocol (UDP) is one of the core protocols of the Internet protocol suite. Using UDP, programs on networked computers can send short messages sometimes known as datagrams (using Datagram Sockets) to one another.

Widget = A component of a graphical user interface that a user can interact with.

Windows = A graphical operative system. It was a system that contributed a lot to spreading computers into the homes of ordinary people. Due to the user-friendly interface, these people could interact with a computer even though they did not have any real computer knowledge.

4. User requirements Definition

4.1. The scope of the system

Topic	In	Out
Multiplayer mode for both local area network (LAN) and internet	X	
Save functionality		X
Turn based		X
The game is played in real time (buildings, troops)	X	
Training mode (Single player)	X	
Artificial Intelligence (AI), Computer that join as a player		X
2D graphical view	X	
Build and expand village	X	
More than one village per player		X
Build and expand troops	X	
Move and attack using troops	X	
Map that describe the game field	X	
More then one race to play (Different races to choose from)	X	
Sound effects		X
Mouse and keyboard to play the game	X	
Simple game chat	X	
Interface when creating and joining a game	X	
Tutorial that describe how to play the game		X
Observer mode for multiplayer when a player is dead	X	

Pause the game		X

4.2. The main factors that need to be taken in to account when designing and building the system

Creating a user interface that is attractive, easy to learn and simple to use
- To make the game successful
- To attract new users
- To make the game amusing

Making the game strategically challenging
- Keeping players interest in the long run
- To provide user value

Creating a balanced game
- One type of player race could become to strong in relation to others.
- Require extensive testing

Bad network connections
- Will result in latency which needs compensation.

All players need the Java Runtime Environment
- Information that need to be clearly stated to new users
- An obstacle for new users who don't have Java installed
- Making the game dependent on external factors

4.3. Use Case Narratives

4.3.1. Narrative A – Bill and his friends

Bill, a 16-year old high school student, comes home from school fairly early in the evening. After having spent several hours on studying for his next exam, he feels bored and is longing for something different and entertaining to do. He realizes to his surprise that he hasn't had time to play on his computer for several days. Therefore he turns on his computer and starts searching the Internet for a game to play. He finds a strategy war-game on a web site, downloads it and realizes that it supports multiplayer mode! So he calls five of his friends that also likes computer games and helps them download the game as well. He tells them to all get on MSN, a messenger service they usually use when communicating to each other over the Internet. He starts the game and tells his friends to do the same.

In the game he sees, that to start a multiplayer game, somebody has to be the host. As he was the one to find the game in the first place, he naturally chooses the role of host. He chooses an alias, configures the specific game settings and opens up a pre-game screen which his friends can join before actually starting the game. Since his friends are starting the game in client mode, he gives them his IP to connect to, which is showed on his current game screen. He can see the player slots filling up on his pre-game screen, and when everybody are connected and have pushed "Ready", he starts the new game session.

The game round is up and running and everybody are competing as opponents against each other. He notices that he has one village to control, and that his opponents seem to have one as well. He builds up his village with new buildings, choosing strategically which ones to build first. Exploring the map he sees that his opponents do not seem to have any military strength, so he starts producing soldiers. He then attacks his friends one by one, destroying their villages, making each of them lose the game when their village is lost. They are not kicked out of the game but can stay in "Observe Mode", seeing what takes place with the remaining players but not actually doing any useful. When he has conquered all of his friends' villages, the game ends, leaving him as the sole winner.

He then shuts down the game and turns off the computer, tired but with a wicked smile on his face.

4.3.2. Narrative B – Lasse comes home from work

Lasse, a 28-year old software developer with interests in computer games and strategic board games, is at home after a tedious and stressful day at work. The whole day he has been thinking about the best way to beat his friends in the strategic war-game they tried last week.

Having his computer already on as usual, he sits down at it and starts up the same strategic game. At the main menu he has to choose if he wants to play the game in single –or multiplayer mode. He doesn’t want to call his friends to start up a multiplayer game since he first wants to try out the new buildings tactic he has planned throughout the day. This is easier done because there are no opponents that can disturb you or any time constraints that limits the game length. Therefore, he chooses to play the game in “Single player – training mode”.

The game loads a map and starts the main interface. Lasse chooses a place to settle his village and can then start to play. Lasse learns tries out his new building tactics to see how much time it takes, how much his resource income is and how many troops he can have ready until a certain point he has estimated that his opponents might have a sizeable army to attack him with. He feels confident about his new tactic, looking at the size his own army he was able to produce during the span of time. He then quits the game by clicking the exit button first in the game and then in the game menu. He just can’t stop thinking about trying of the new tactic in a multiplayer game against his friends. He is confident it will rock their earth.

4.3.3. Narrative C – Peter and Lars

Peter receives a phone call from his friend Lars. Lars is very excited over a new strategy game he has found on the web. Lars wants Peter to join in to try the multiplayer functionality. Lars and his other friend Gunnar is at Lars place with their computers and are connected via LAN. Lars tells Peter to check is email, where he will find the download location of the game and Lars’ IP address to connect to when in the game. Peter who likes a strategical challenge every now and then decides to join in. He downloads the game and is at the same time informed that he needs the Java Runtime Environment which also downloads. He then starts the game choosing to join a multiplayer session. He enters a player alias and is then asked to enter the host IP which he had got from Lars. Doing that he finds himself in at the multiplayer game start up screen, where he chooses ready. As he hits the ready-button, the game starts right away, probably since Lars and Gunnar has been waiting already pressing their ready-buttons.

Peter finds himself with an overview screen with a map to the right. A text message from Lars says: “Put your village somewhere on the map”. Peter randomly moves the mouse pointer over the map area and presses the left mouse button where he finds an appropriate for his village. A millisecond later his little sister pulls the ADSL-modem wire out of the wall. Peter’s game client is terminated and his on going game round quits and the start-up screen is displayed. The multiplayer game continues without Peter’s participation and his village is destroyed.

4.4. Functional requirements

4.4.1. World specifics

4.4.1.1. Player controlled villages

There shall exist villages which are owned and controlled by a player, where he or she can construct buildings.
Rationale: A village will consist of a specific set of building slots. These will not be visible on the map; instead they will always be visible in a separate overview frame. A village will always occupy one square on the map.

4.4.1.2. One village per player

Each player shall own exactly one village.
Rationale: It’s much easier for the player to control and maintain only one village.

4.4.1.3. Resource

There shall be a resource that each player can gather. This resource will be used to build and expand the village and/or troops.
Rationale: Resource is equal to money in the real world. Each player will have a base income and can then upgrade their cash flow with a building that increases their income.

4.4.1.4. One type of resource

There shall be only one type of resource.
Rationale: Having one type of resource make it easier for the players to focus on the game play instead of which type of resource they need to gather.

4.4.1.5. Playable races

There shall be different playable races with unique attributes.
Rationale: Each race corresponds to a different social class. The race specific attributes will during the game round affect e.g. building construction times, cost of troops and strength of troops.

4.4.2. Building

4.4.2.1. Building slots

A building slot shall be able to hold zero or one building.

Rationale: Building slots are used for constructing buildings and can either be empty or contain one building.

4.4.2.2. Different types of buildings

There shall exist different types of buildings with different benefits.
Rationale: Different types of buildings will force the user to make choices of what buildings to build in what order.

4.4.2.3. Construct buildings

Each player shall be able to construct buildings using empty building slots.
Rationale: A player must be able to construct buildings in order to provide him/herself with the functionality and benefits each building gives. The buildings will not be visible on the map.

4.4.2.4. Upgrade buildings

Each constructed building shall be upgradeable a certain amount of times (referred to as “levels”). Each upgrade will improve the functionality of the building.
Rationale: When a player has constructed a building the player will have the option to upgrade the building until it reaches its maximum level.

4.4.3. Armies

4.4.3.1. Different types of troops

There shall exist different types of troops with different military strengths.
Rationale: In combat, all troops are strong against one type, and at the same time weak against another.

4.4.3.2. Train military troops

It shall be possible to train military troops.
Rationale: The player must be able to train troops in order to be able to attack other players or defend his own village. The player decides when he wants to start train a troop. Each troop has a certain time it takes before the training is complete. The troops are trained at specific buildings. When a troop has been trained it is automatically positioned on the player’s village location on the game map.

4.4.3.3. Armies consists of troops

An army consists of one or several troops positioned on the same square on the game map.
Rationale: The consequence of having troops in the same army is that they are controlled together and can not be moved individually. Also, - the army move with the movement speed of the slowest moving troop in the army.

4.4.3.4. Armies never separate

Armies shall not be able to be split into different armies once they have been formed.
Rationale: This restricts the player from merging and separating troops constantly and will affect strategical game play.

4.4.3.5. Attack villages with armies

It shall be possible to attack other players’ villages.

Rationale: If an army enters the same cell on the map as where a village is situated, a combat will resolved. This will either result in the army being defeated or the village being destroyed.

4.4.3.6. Conflict of armies

If two or more armies (from at least two different players) rendezvous at the same cell on the map a combat shall be invoked.
Rationale: Armies placed in the same cell will automatically fight each other in a combat. This will result in one army defeating the other one and becoming victorious.

4.4.3.7. Move armies around the map

Each player shall control their own armies and use them to move around the map.
Rationale: The troops can be moved around the map to secure areas around the map or check on the opponents what kind of troops they are using.

4.4.4. Map

4.4.4.1. Game map

The game shall have a map that shows the game field.
Rationale: The map shall be a grid-based, square map of cells.

4.4.4.2. Map cells

There shall be different types of map cells with different environment types.
Rationale: Some types are not passable by armies while certain can be moved over. Each army can only belong to exactly one cell at a time.

4.4.4.3. Fog of war

The map shall be totally unrevealed when the game starts. Each village and each army will reveal parts of the map around them.
Rationale: Each player can use their armies to move around the map, thus revealing the map piece by piece. When an army is killed or moved to another place the fog of war will take place again and hide the map that was revealed.

4.4.5. Multiplayer mode

4.4.5.1. Player name

Each player shall have a unique player name.
Rationale: It will make it easier for the other player to know who they are playing against.

4.4.5.2. Connect to a multiplayer game

The players shall connect to the game host via IP address.
Rationale: Using IP addresses makes it easy for the network to talk to each other, since all computers have a unique address.

4.4.5.3. Multiplayer Game Settings

The host shall be able to set specific game settings for an upcoming multiplayer game before it is started.
Rationale: Being able to change some settings will increase the variety between different games and make each game round even more unique.

4.4.5.4. Observer mode

When a player is killed during a multiplayer game he shall be able to continue viewing the game as an observer. He will not be able to actively participate in the game any more.
Rationale: The possibility of being able to follow the last events of the game is needed to make the lower skilled players learn from more skilled ones. It also reduces the boredom when waiting for a new game round. Of course the player will also be able to quit the game round without becoming an observer.

4.4.5.5. Simple game chat

Each player shall be able to send text messages that will be broadcasted to every other player during the game.
Rationale: The game will provide a simple chat feature where each player can write and send messages that every other player will see. This will provide a communication interface for every player.

4.4.5.6. Player specific colours

Each player shall have a specific colour.
Rationale: This will distinguish the armies and villages on the map so that each player can see who's controlling what.

4.5. Non-functional requirements

4.5.1. Usability requirements

4.5.1.1. 2D graphical view of the game

The game shall provide a simple 2D graphical view.
Rationale: The game will run smoother and will be playable on slower machines. It will also provide a better game overview.

4.5.1.2. Game overview

The game window shall provide an overview of the game where the player can see his/hers village, the map and view showing the currently selected item (building, army, etc).
Rationale: The game window will be fixed and provide a constant view showing the map and the player's village. There will be a view showing whatever is currently selected. If a building is selected it will show the upgrade options and other functionality assigned with that building. Above the map the player will see its resources, the current time and other important things. Below the map the chat will be located.

4.5.1.3. Keyboard & mouse control

Keyboard control will be used to choose name, communicate via the game chat and to use keyboard short cuts. Mouse control will control other interaction.
Rationale: All computers have a mouse and a keyboard and it is the best control method in strategic gaming.

4.5.1.4. Training mode

The game shall provide a training mode where a player can play the game as the only team on the map. This will be the only single player mode in the game.
Rationale: The training mode will provide a way for the player to play the game without connecting to another player. The player will be alone on the map and will not be able to

defeat anyone or anything. The sole purpose of the training mode is to provide a way for the player to train and master his techniques in the game, i.e. build and expand the village in the best way.

4.5.2. Efficiency requirements

4.5.2.1. Real Time

The game shall run in real-time which mean that all players play simultaneous (To be compared with turn-based where each player does their moves on at a time).
Rationales: Players won't have to wait on each other to do their moves and real-time can also contribute to stressful and funny situations during the game.

4.5.2.2. Average game length

Average game length should be between 10-30 minutes.
Rationale: Game length is an important factor since people will likely play the game during breaks when it is advantageous to know in advance approximately how long a game takes. This gets even more important since the game lacks pause function. Shorter game round also makes waiting times shorter for players that have been eliminated during the game round.

4.5.2.3. Multiplayer

A multiplayer game shall be playable for two to six players.
Rationale: The limitations of participants are set to a fixed number to enable fast game rounds.

4.5.2.4. Hard drive space

The game will require only a small amount of hard drive space.
Rationale: The game will not have i.e. extensive graphical textures, because of the focus on the strategical level of the game why the required hard drive space will be small.

4.6. Use Cases

4.6.1. Launch the game application

Primary actor: User
Stakeholders and Interests:
- User: Wants to start the game
Preconditions: The user has downloaded the game on his computer and has Java Runtime Environment 1.5 installed.
Success guarantee (Post conditions): The game application is running
Minimal guarantee: -

Main Success Scenario (or Basic Flow):
1. The user looks up the game executable path.
2. The user clicks on the game icon which starts the games executable file
3. The game file is launched.

Extensions (or Alternative Flows):
2a. The user can't find the executable file

    1. The user searches the hard drive for the games executable file
3a. An execution error occurs
    1. The computer returns to the state before trying to launch the application

4.6.2. Start a training mode game

Primary actor: User
Stakeholders and Interests:
- User: Wants to increase his gaming skills
Preconditions: The user has **launched the game application** launched.
Success guarantee (Post conditions): The user is in training mode.
Minimal guarantees: The computer and the operative system do not crash.

Main Success Scenario (or Basic Flow):
1. The user chooses to play Training mode.
2. The user chooses the game map settings.
3. The user chooses to start the game.
4. A training mode session starts.

Extensions (or Alternative Flows):
4a. User regrets his choice of the game setting.
    1. User chooses to cancel training mode
    2. User returns to Step 2 in main scenario.

4.6.3. Host a multiplayer game

Primary actor: User
Stakeholders and Interests:
- User: Wants to create a multiplayer game with other players involved
- Other users: Wants to participate in a multiplayer game
Preconditions: The game application has been launched
Success guarantee (Post conditions): A multiplayer game is created and launched
Minimal guarantees: The operating system remains intact

Main Success Scenario (or Basic Flow):
1. User chooses to play a multiplayer game from the main menu.
2. User chooses to host a game.
3. User enters an alias that he will use during the whole game round.
4. User selects the game settings.
5. User selects the maximum limit of players that can join this game round.
6. User chooses to create the game.
7. User is marked as ready and is waiting for other clients to connect.
8. When all players are connected and ready the game starts.

Extensions (or Alternative Flows):
8a. User regrets his choice of hosting a game
    1. User cancels the game hosting.
    2. User returns to the main menu.

8b. User regrets his choice of the game setting.
  1. User chooses to cancel hosting the game
  2. User returns to Step 4 in main scenario.

8c. No user connects to the host
  1. The user cannot start the game.

4.6.4. Join a multiplayer game

Primary actor: User
Stakeholders and Interests:
- User: Wants to play a multiplayer game
- The game host: Wants player to get connected to start a multiplayer game

Preconditions: The user has received an IP address to connect to.
Success guarantee (Post conditions): A multiplayer game is started
Minimal guarantees: The computer and the operative system do not crash.

Main Success Scenario (or Basic Flow):
1. The user chooses to play a multiplayer game.
2. The user chooses to join a game.
3. The user chooses a player name
4. The user types in the IP address of the multiplayer host
5. The user chooses to connect to the entered IP address
6. A connection between the user and the host is established
7. The user enters the multiplayer setup mode.
8. The user chooses to make himself ready to start the game
9. The game is started by the game host.

Extensions (or Alternative Flows):

3a. The user types the wrong name
  1. User changes his name

5a. The wrong IP address was entered
  1. The system puts the user back to choose name and IP for the connection.

5b. The host is not responding
  1. The system puts the user back to choose name and IP for the connection.

6a. No connection is established
  1. The system puts the user back to choose name and IP for the connection.

4.6.5. Win a multiplayer game

Primary Actor: User A.
Stakeholders and Interests:
- User A: Wants to win the game
- User B: Wants to play the game
- User C: Wants to play game

Preconditions: There is an ongoing multiplayer game.
Success Guarantee (Post conditions): User A controls the only remaining village and therefore wins the game.
Minimal Guarantee: The reason why a player did not win the game is stated.

Main Success Scenario (or Basic Flow):
1. User A builds new soldier-producing buildings in his village.
2. User A trains a couple of troops.
3. User A sends all his troops against User B's village.
4. User A's troops arrives at User B's village.
5. User B has no troops in his village why no battle between troops commences.
6. User A destroys User B's village.
7. User A is the only player who controls a village.
8. User A is proclaimed the winner.
9. The multiplayer game exits.
10. Exit scenario, Success.

Extensions (or Alternative Flows):
4a. User B has troops in his village.
  1. User A has no troops in his village.
  2. User A wins the battle and kills all of User B's troops in the village.
  3. Continue at Step 6 in main scenario.

4b. User B sends troops and attacks User A's village.
  1. User A has no troops in his village.
  2. User A loses his village which is destroyed by User B's troops.
  3. User A is proclaimed to have lost the game.
  4. User A can no longer interact with the ongoing game.
  5. Exit scenario, Failure.

7a. There are still other users except User A that controls a village.
  1. User A continues attacking other players.
  2. User A changes attacking focus to attacking User C
  3. Continue at Step 2 in main scenario replacing User B with User C.

4.6.6. Leave a multiplayer game round

Primary actor: User A
Stakeholders and Interests:
- User A: Wants to leave the current game round.
- Other users: Wants to continue playing the game.

Preconditions: A multiplayer game with a set of users has been initialized. User A is not the host of the game.
Success guarantees (Post conditions): User A has left the game round.
Minimal guarantee: User A’s game round has been terminated.

Main Success Scenario (or Basic Flow):
1. User A chooses to give up game round
2. User A confirms his choice to leave the multiplayer game.
3. User A gets back to the main menu.

Extensions (or Alternative Flows):
2a. User A wants to keep watching the remainder of the game.
  1. User A chooses to stay in Observer mode

2b. User wants to exit the application

1. User A chooses to leave game round by exiting directly to the operating system.

5. System architecture

This section describes the system architecture, both as an overview and as modular decomposition of the application.

5.1. Overview

When playing a multi player game, the application utilizes a client-server model.

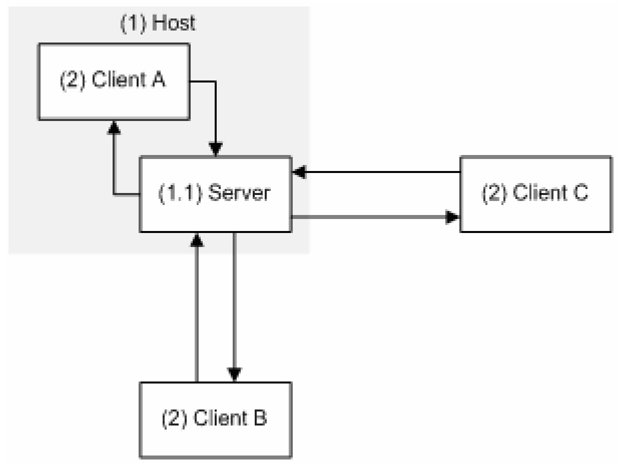

Figure 1. Application network overview.

5.1.1. Host

The host is the computer hosting a multiplayer game and is decomposed into a server and a client. The server is started as a standalone part of the host and the client of the host connects to the server as any other client. This way the hosting client doesn't have to be distinguished from the other clients.

5.1.1.1. Functions in this module

- Starting a server
- Closing a server

5.1.2. Server

The server handles all connected clients, incoming connections and other events associated with the network layer of the game. All communication among the clients are sent through, handled by, and forwarded from the server.

5.1.2.1. Functions in this module

- Accepting/Refusing connections
- Message checking and forwarding
- Handling network events (unexpected disconnects and such)

5.1.3. Client

The client is the game application running on every connected computer.

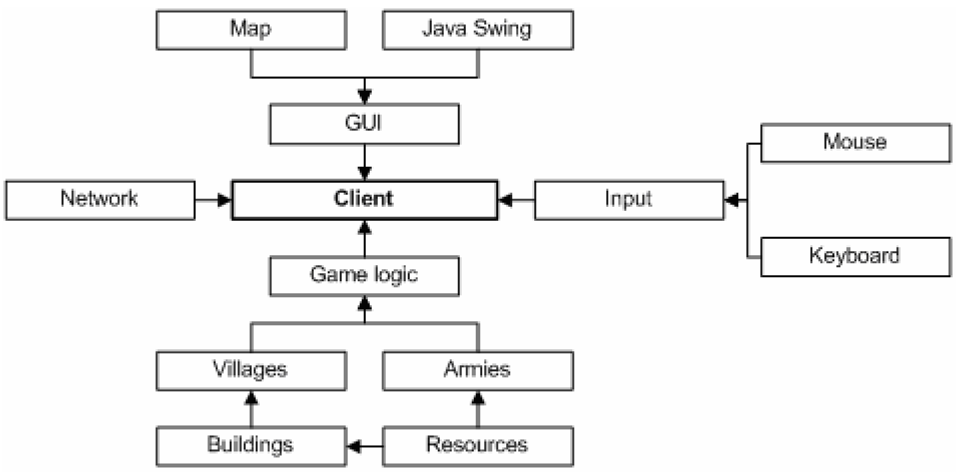

Figure 2. Client modular decomposition.

5.1.3.1. Functions in this module

- Creating a multiplayer game
- Joining a multiplayer game
- Connecting/Disconnecting to a server
- Handling input
- Handling game logic
- Visualizing current game state
- Reflecting game state to server through network
- Handling incoming data from server

6. System requirements

6.1. Functional Requirements

6.1.1. World specifics

6.1.1.1. Establish village control in multiplayer mode

Function:	Establish village control: A player needs a village to participate
Traceback:	User requirements: - Player controlled villages
Description:	Establishes a village for a player as the player manually selects a place for his village to situate on the map during the initial phase of the game round.
Inputs:	The player identity (playerId), the mouse location (mouseLoc), the map coordinates (mapCord)
Source:	The player identity assigned by the game host. The map specifications of the spot of village establishment.
Outputs:	playerVillageId – The village now obtained by a specific player.
Destination:	Game main interrupt target

Action:	All players that enter the game are already in a possession of a village but needs to choose a spot where the village shall established. Each player therefore has to pick place on the map and establish the village. As all players enter the game at the same time there is an initial competition for the best situated spots. As a player finds a desired spot he can choose to establish his village on this spot, unless another player has already established his village on the exact same spot.
Requires:	Reading of the player identity and available map spots.
Pre-condition:	The player has not already established a village.
Post-condition:	Village specifics are assigned to the player's village.
Side-effects:	There is one less available spot on the map where a village could be placed.

6.1.1.2. A player shall have one village

Function:	Assign a village to player: A player needs a village to participate
Traceback:	User requirements: - One village per player
Description:	As a player enters a game round the player will posses a village to place on the map**.**
Inputs:	The player identity (playerId),
Source:	The player identity assigned by the game host.
Outputs:	playerVillage – The village attribute now specified for the player.
Destination:	Game main interrupt target
Action:	To play and participate in the game you need a village. As the game starts up the system assigns a village the each player.
Requires:	Reading of the player identity.
Pre-condition:	The player identity has not got a village.
Post-condition:	A village is assigned to the player's identity.
Side-effects:	None.

6.1.1.3. One collectable resource

Function:	One collectable resource: The game contains a resource that enables player village development.
Traceback:	User requirements: One type of resource
Description:	The game will contain one type resource that a player could exploit in order to increase the player's resource level to be able to further construct buildings and troops within the game round**.** Every player has a basic income at the beginning of the game round. The resource is not specific to the player race but something that every player will need to collect in order to be successful in the game. A player could collect more resources by creating and upgrading a resource collection building. This building is dedicated to collecting resources. The resource income level is also depending one the quotas of collection population, the part of population that is dedicated to build and collect resources.
Inputs:	None.
Source:	None
Outputs:	resourceLevel – The basic level of resource income of the village, resourceEarningsLevel – The level of resource increment.
Destination:	Player village specification.
Action:	None.

Requires:	Village identity.
Pre-condition:	None
Post-condition:	Every village has a level of resource income.
Side-effects:	None.

6.1.1.4. Resource collection

Function:	Resource collection: A player needs resources to be competitive in the game.
Traceback:	User requirements: Resource
Description:	A player could increase its village level of resource earnings. This could be done by building a resource collection building. This building will increase the current level of resource earnings. Further upgrading of the resource collection building will further increase the earnings. The resource income level is also depending one the quotas of collection population, the part of population that is dedicated to build buildings and collect resources.
Inputs:	Player Identity – playerId, the village identity – villageId, The current level of earnings – resourceEarningsLevel, number of worker – numberOfWorkers, workers/warriors ratio – buildingRatio, Level of resource collection building - levelOfRCB
Source:	The player identity, village identity assigned by the game host, village specification.
Outputs:	resourceEarningsLevel – the changed level of resource income.
Destination:	Game main interrupt target
Action:	The system calculates the resource collection level by adding numberOfWorks*buildingRatio*levelOfRCB to the resourceEarningsLevel for the playerId and VillageId.
Requires:	Reading of the player identity, village identity, resourceEarningsLevel, numberOfWorkers, buildingRatio, levelOfRCB
Pre-condition:	The player has established village
Post-condition:	A village has a new resource collection level
Side-effects:	None.

6.1.1.5. Choosing a player race

Function:	Choosing a player race: A player needs a defined player race
Traceback:	User requirements: - Playable races
Description:	Before a game can be initialized each participating player needs to specify his desired player race. The player race defines the race specific capabilities that the player will benefit from during the game round**.** The race specific attributes will affect e.g. construction times, cost of troops and strength of troops.
Inputs:	The player identity (playerId), the player race choice (raceId)
Source:	The player identity assigned by the game host, the game races
Outputs:	playerRace – The race attribute now specified for the player.
Destination:	The main game loop.
Action:	To play and participate in the game you need to choose a player race. During the initialization of game the system assigns a race to each player.
Requires:	Reading of the player identity.
Pre-condition:	The player identity has not got a race.
Post-condition:	A race is specified and assigned to the player’s identity.
Side-effects:	None.

6.1.2. Building

6.1.2.1. Building slots

Function:	Create building slots in a village: A village needs building slots in which you can place a building.
Traceback:	User requirements: - Building slots
Description:	A village will contain building slots in which you can choose to build buildings.
Inputs:	A village (villageID)
Source:	The village specifications
Outputs:	building slots – The village now contains building slots.
Destination:	Game main interrupt target
Action:	When a village is created by the system it does not have any building slots. This function will create building slots.
Requires:	A village without building slots.
Pre-condition:	A village has been created.
Post-condition:	A village with building slots.
Side-effects:	None

6.1.2.2. Construct different kinds of buildings

Function:	Creates a building in an empty building slot. : Players need different kind of buildings to provide services as for example produce troops.
Traceback:	User requirements: Different types of buildings (1.2.2) and Construct buildings (1.2.3)
Description:	Players shall be able to build different kinds of buildings in their villages empty building slots. Different kind of buildings can provide different kind of services to the user. E.g. A troop-training-building can create military troops or a village-wall can add defence bonus to troops located in the city. A building takes a specific amount of time and resource to build witch depends on the type of building.
Inputs:	A village (villageID), an empty building slot (slotID)
Source:	building specification
Outputs:	a building (buldingID)
Destination:	Game main interrupt target
Action:	When a village is created it doesn't have any buildings. This function creates different kinds of buildings in empty building slots (One building per slot.)
Requires:	An empty building slot and the amount of resource that type of building costs.
Pre-condition:	A village with an empty building slot.
Post-condition:	A building exists in the building slot.
Side-effects:	One empty building slot less in the village

6.1.2.3. Upgrade buildings

Function:	Upgrades a building: An upgrade will improve the functionality of the building.
Traceback:	User requirements: - Upgrade buildings (1.2.4)
Description:	Each constructed building shall be upgradeable a certain amount of times (referred to as “levels”). Each upgrade will improve the functionality of the building.
Inputs:	A building (buildingID)
Source:	none
Outputs:	a building (buildingID) – The building is now one level higher than before.
Destination:	the building object
Action:	When a building is created it has level 0, this function will upgrade the level of the building by one.
Requires:	A building which haven’t reached its maximum upgradeable level.
Pre-condition:	A building exist witch haven’t reached its maximum upgradeable level.
Post-condition:	A building which is one level higher than before.
Side-effects:	The building will be possible to upgrade one time less.

6.1.3. Armies

6.1.3.1. Different types of troops

Function:	Different types of troops: A player needs troops to attack other players. Different kind of troops creates the possibility of strategies.
Traceback:	User requirements: different types of troops (1.3.1)
Description:	There shall be different kind of troop. Those will have troop-type specific attack strength and defence strength and speed (how fast they moves). Troops can also be especially strong or weak against another type of troop regardless of their individual attack and defence strengths.
Inputs:	The player race (raceID)
Source:	The troop specification
Outputs:	A set of race specific troops. (setOfTroops)
Destination:	Game main interrupt target
Action:	Creates a set of troops for each playable race.
Requires:	Specification of the troops abilities.
Pre-condition:	A game is to be started.
Post-condition:	A game is running and a set of different types of troops have been created.
Side-effects:	None

6.1.3.2. Train military troops

Function:	Training military troops. A player needs troops to be able to defeat other players.
Traceback:	User requirements: - Train military troops
Description:	When playing a game round, all players will need to train troops in order to get strong and to be able to defeat other armies and players. There are different types of troops that are specialized in defeating certain types of other troops.

Inputs:	The player race (raceId), troop type (troopType).
Source:	The train troops building
Outputs:	troop – The troop trained
Destination:	The main game loop.
Action:	If the player owns enough resources the troop/troops will be appended to the current training queue. If/When no other troop is in the queue, the troop will be trained.
Requires:	Enough resources and a building where troops can be trained.
Pre-condition:	The player has built a building where troops can be trained.
Post-condition:	A new troop is being trained or a new troop has been appended to the training queue. Resources have been decreased by the cost of the troop.
Side-effects:	None.

6.1.3.3. Armies consist of troops / Armies never separate

Function:	Troops are considered armies when they are out on the playfield. Armies can be merged but never split.
Traceback:	User requirements: - Armies consist of troops / Armies never separate
Description:	Every troop that is located in the same cell on the map is considered to be an army. If an army is moved from a cell to another and another troop enters the old cell, they will not be of the same army. If two armies (from the same player) move to the same location, they will be merged into one army and can never split again.
Inputs:	Armies/Troops.
Source:	Armies on the map
Outputs:	army – The army that was merged or created
Destination:	The main game loop.
Action:	Armies are merged if they are owned by the same player and moved to the same location on the map.
Requires:	Troops and/or armies
Pre-condition:	At least one army
Post-condition:	At least one army (same as before) or a new army (if two armies have been merged)
Side-effects:	None.

6.1.3.4. Attack villages with armies

Function:	Players use their troops to attack other players' villages.
Traceback:	User requirements: - Attack villages with armies
Description:	When a player has trained enough troops and armies they can be used to attack another player's village. If the attacking army is enough big the village is destroyed.
Inputs:	An army owned by a player (army, playerId1) and a village owned by another player (village2).
Source:	The map
Outputs:	army – the army remains on the playfield if the village is destroyed OR village – the village remains on the playfield if the army is defeated
Destination:	The main game loop.

Action:	If an army attacks a village the village is destroyed if the army’s size is large enough. If the army’s size is not large enough, the village remains on the playfield and the army is defeated.
Requires:	An army owned by a player and a village owned by another player.
Pre-condition:	The player has trained an army. Another player is in the game and has a village.
Post-condition:	A village is destroyed or the army is defeated.
Side-effects:	The army is defeated because it was too small to defeat a village.

6.1.3.5. Conflict of armies

Function:	Two or more armies, owned by different players, located in the same cell on the map will result in a battle (conflict).
Traceback:	User requirements: - Conflict of armies
Description:	When an army from a player enters the same cell as an army from a different player a conflict will arise and the armies will start to battle each other. This will result in one army being victorious and another one being defeated.
Inputs:	Two different armies (army1, army2).
Source:	The map
Outputs:	army1 or army2 depending on who won the conflict
Destination:	The main game loop.
Action:	Two armies start a battle. The stronger army will defeat the other but will also loose troops depending of the size and strength of the other army.
Requires:	At least two different armies (one from a player and another from another player) located in the same cell on the map.
Pre-condition:	Two armies.
Post-condition:	One surviving army.
Side-effects:	None.

6.1.3.6. Perform movement of players armies

Function:	Perform movement of armies: A player must be able to move his armies
Traceback:	User requirements: - Move armies around the map
Description:	Moves a player’s army around the map as the player selects an army and a destination.
Inputs:	The player identity (playerId), the army (chosenArmy), the destination (destination)
Source:	The player identity assigned by the game host in multiplayer or assigned by system itself if single player. The army selected earlier by the players left mouse click. The destination chosen at the start of movement request by the player right mouse clicks.
Outputs:	A path for the army to start moving by.
Destination:	Game main interrupt target
Action:	A player can always have one army selected at a time by mouse left-clicking on the cell populated by the army on the map view. The player can then issue a movement command by mouse right-clicking on another cell on the map. The system calculates what cells the army should cross to reach its destination whilst not colliding with blocked cells.
Requires:	Reading of the players identity and path finding around blocked cells on the map.

Pre-condition: The player has an army selected and has issued a movement command.
Post-condition: A path for the army has been found and movement initialized.
Side-effects:

6.1.4. Map

6.1.4.1. Provide interactive game map

Function: Provide a game map: The players must be able to see where armies and villages are located. The armies and villages must be able to exist on unique cells.
Traceback: User requirements: - Game map
Description: Provide a game map that is grid-based. It is a square map with many cells.
Inputs: Size of map (mapSize).
Source: Host before start up of the game at the game map settings.
Outputs: A game map with chosen size.
Destination: An initialized game.
Action: A map is generated upon game start and can be used by the players to select/move armies, see opponents and their own village's locations.
Requires: A map generator.
Pre-condition: A game has been initialized.
Post-condition: A map is generated and can be used by the players.
Side-effects: None

6.1.4.2. Constructs a map with different kinds of map cells

Function: Constructs the map with different kind of map cells: The game world will be more realistic than if there were only empty map cells.
Traceback: User requirements: - Map cells
Description: There are different kind of map cells. These are used when the system constructs the game map. Some are not passable by armies while others can be moved over. Others might give an environmental bonus for the army/village placed upon the cell on the map.
Inputs: Host has chosen what environment types are allowed in the game.
Source: Chosen by host before start up of the game, in the game settings menu.
Outputs: Different cells on the game map.
Destination: Game map in an initialized game.
Action: The system will use different cells when generating the game map for an upcoming game.
Requires: Map generator making use of different map cells.
Pre-condition: Different cells have been defined
Post-condition: A map being able to contain different kinds of cells.
Side-effects: Some parts of the map are strategically better to place one's village in.

6.1.4.3. Cover the map in a fog of war

Function: Cover the map in a fog of war: Will make the game more interesting
Traceback: User requirements: - Fog of war
Description: The map is always covered in a fog of war, making it only possible to see parts of the map where a player's village and own troops are located.

Inputs:	The player identity (playerId), his armies location(armies[].location), his village location (villageName.location).
Source:	The player identity (playerId) assigned by the game host. The map specifications of the player's village and armies' locations.
Outputs:	A grey clouded area over those cells that are not close to inputted locations.
Destination:	Game map.
Action:	The map is constantly covered in the fog of war for each player if they do not have a close by army/village. If armies are moved around the map, new parts of the map will be revealed while parts at the army's old location will be covered in the fog of war.
Requires:	Reading of the players identity and fog-Calculator checking what cells should be covered
Pre-condition:	A game has been initialized.
Post-condition:	Some parts of the map are covered in a fog of war.
Side-effects:	The players have to use armies to see what their opponents are up to.

6.1.5. Multiplayer mode

6.1.5.1. Assign each player a unique player name

Function:	Assign player names: A player needs a unique name to be identified by.
Traceback:	User requirements: - Player name
Description:	Assigns unique names to the players as they join the multiplayer game setup mode. If the name is equal to a name of the other player, modify the name of the latest player joined.
Inputs:	Latest joined player's name (playerName). The names of the others players in the game (playerNames[]).
Source:	The players have chosen a name of their choice prior to joining/hosting a multiplayer game. The names of all players in the current game are accumulated as more and more players join the game.
Outputs:	A unique name for each player.
Destination:	Multiplayer game setup mode
Action:	Each player has earlier chosen a player name. When they join the multiplayer game setup mode they will be assigned this name as long as they are not a duplicate of an already joined client's name. In this case the name will be altered so that it is unique.
Requires:	Name-checks on players that join the game.
Pre-condition:	A multiplayer game is hosted by a client but not started.
Post-condition:	All player names are unique.
Side-effects:	Some names may be altered slightly.

6.1.5.2. Connect to a multiplayer game

Function:	Connect to a multiplayer game: The players need to know who the data should be sent to.
Traceback:	User requirements: -Connect to a multiplayer game
Description:	Give the connected player an address to the game host so data can be transferred.
Inputs:	The connected players game name and game host IP address.
Source:	The player who wants to connect enters the IP and alias in a text label.

Outputs: A connection to the game host.
Destination: Before the game start.
Action: All players that want to play need a connection to the game server to be able to play. Each player need to know the game host IP address and connect to him/her before the game start. The connection is needed to be able to send data to each other otherwise we wouldn't have a game at all. If a player lost connection he/she is out of the game and has to connect to another game.
Requires: Game host IP address and alias from the player.
Pre-condition: The player has written his/her alias and game host IP.
Post-condition: A connection between the player and server.
Side-effects: One of the players might have a bad connection to internet and it will lead to disconnection between the server and the player.

6.1.5.3. Choose Multiplayer Game Settings

Function: Chose multiplayer game settings: Different options in the game.
Traceback: User requirements: - Multiplayer game settings.
Description: The game host is able to choose different options to play with. He can chose how many players that's going to play and how they are going to pick there villages (random or manual).
Inputs: Game settings.
Source: The game host selects his/her options in text labels and buttons.
Outputs: Different game settings.
Destination: Before a game is hosted.
Action: The game host has to enter some game settings to be able to start a game. He can always leave it as it is and have standard settings or change maximum players that's allowed to join the game and how they will pick there villages (random or manual).
Requires: The maximum players should be less then 8 and greater then 1.
Pre-condition: A player has chosen to be host.
Post-condition: The game starts with the settings that the player elected.
Side-effects: None.

6.1.5.4. Observer mode

Function: Observer mode: Let the player stay in the game even if he lost
Traceback: User requirements: -Observer mode
Description: If a player is defeated by his opponents he will not be able to play anymore. He can then choose to stay in observer mode or leave the game. If he choose to stay in observer mode he should be able to se all the players on the map all the time with no fog of war.
Inputs: The player choose observer mode or leave game.
Source: Button to choose either observer mode or leave game.
Outputs: Depending on what button is pressed, different readings.
Action: When the player is defeated he can choose to stay in observer mode or leave the game.
Requires: The player must know if he want to stay in game or leave right now.
Pre-condition: A player is defeated.
Post-condition: The player is in observer mode or left the game.
Side-effects: None.

6.1.5.5. Simple game chat

Functional:	Simple game chat: Players are able to talk to each other
Traceback:	User requirements: -Simple game chat
Description:	Simple game chat is a very simple chat that players can use to send messages to each other.
Inputs:	Messages in form of strings.
Source:	Chat window with a text label to write the message in and another label that show the history of the chat. There is also a button to send the message with.
Outputs:	Sending message to each other in the game chat.
Action:	When players write and send something in the chat they send the message to everyone connected to the game chat.
Requires:	A started game.
Pre-condition:	A game has started.
Post-condition:	The game is still going on.
Side-effects:	None

6.1.5.6. Player specific colours

Functional:	Player specific colours: Unique colour for all the players
Traceback:	User requirements: -Player specific colours
Description:	Each player should have a unique colour to know who’s controlling what.
Inputs:	None
Source:	Each armies and the village a player control should be of the unique colour
Outputs:	None
Action:	When you se an army or village you look at the colour to se who’s controlling them
Requires:	More then one player
Pre-condition:	A started game with more then one player.
Post-condition:	The game is still going.
Side-effects:	None

6.2. Non functional requirements

6.2.1. Usability requirements

6.2.1.1. 2D graphical view of the game

Description:	The visual perspective of the game shall be a 2D graphical view, a bird’s eye perspective. The simple graphics will enable the game to run smoother on a large variety of computer machines. A two dimensional game overview will increase and enhance the strategical possibilities in the game. The system will divide the computer display in a set of frames. These frames will remain the same during the whole game and are not changeable in size. The larger frame on the right hand side will display the game map. On the left hand side

smaller frames will display the player village and player statistics like resources, levels of income and troops etc.

System goal: The game shall be playable on any computer not depending on individual computer hardware. The game shall be easy to overview.

Verification: The visual overview of the game shall interpretable and understandable to a person that has never been playing the game before but who is familiar with computer gaming in general.

6.2.1.2. Game overview

Description: Game overview is the things that will be shown when you play the game. Map, village and a current target window will be showed.

System goal: To make it easier for the players to play the game.

Verification: There are 3 different windows that contain the map, village and the selected entity.

6.2.1.3. Keyboard and mouse control

Description: Keyboard and mouse will be used to interact between the game and the players.

System goal: It should be possible to do something in the game, and then you need something to interact with.

Verification: Both the mouse and keyboard will be used to play the game.

6.2.1.4. Training mode

Description: The game shall provide a training mode where a player can play without opponents.

System goal: The game shall be able to initialize and run a single player training mode game.

Verification: The training mode shall be able to be started by any player from the main menu, without the need of an Internet connection or friends to play with.

6.2.2. Efficiency requirements

6.2.2.1. Real Time

Description: The game shall run in real-time which mean that all players play simultaneous (To be compared with turn-based where each player does their moves on at a time). The system will have a speed attribute which will be fixed during a game and be the same for all games. The speed attribute will set the pace of the game. If you for example would double it everything will happen twice as fast (troop movements, etc.). The amount of time all different operation in the game takes will be expressed in this attribute. E.g. A troop can move one square on '5t' where 't' is the system speed attribute which could be one second. In that way you can adjust the speed of the entire game.

System goal: The game shall run smooth: Different players will actually be able to play simultaneous and in real-time and not have to wait because of computer processing.

Verification:	By test playing the game with maximum number of players and large amount of operations going on at the same time (troop movements, creating buildings, etc.).

6.2.2.2. Average game length

Description:	The average length of a game round will be around 10 to 30 minutes.
System goal:	To make the game quick and playable during a short break, for example during lunch.
Verification:	A game round will not last for hours.

6.2.2.3. Multiplayer

Description:	The game shall provide multiplayer games for two up to six players.
System goal:	The game shall be able to initialize and run a multiplayer game
Verification:	A multiplayer game shall be able to be initialized by a group of players, connecting to the same host and starting to play within the same multiplayer game.

6.3. Use Cases

6.3.1. Launch the game application

Primary actor: User
Stakeholders and Interests:
- User: Wants to start the game
Preconditions: The user has downloaded the executable game file on his computer and has installed Java Runtime Environment 1.5.
Success guarantee (Post conditions): The game application is running
Minimal guarantee: -

Main Success Scenario (or Basic Flow):
1. The user looks up the game executable path.
2. The user clicks on the game icon which starts the games executable file
3. The system launches the game file.
4. The system displays the game main menu.
5. The system waits for the player input.

Extensions (or Alternative Flows):
2a. The user can’t find the executable file
    1. The user searches the hard drive for the games executable file
    2. The user launches the now found executable file.
2b. The system can’t find the executable file.
    1. The user gives up trying to execute the game.
3a. An execution error occurs
    1. The computer returns to the state before trying to launch the application

6.3.2. Start a training mode game

Primary actor: User

Stakeholders and Interests: - User: Wants to increase his gaming skills
Preconditions: The game application has been launched.
Success guarantee (Post conditions): The user is in training mode.
Minimal guarantees: The computer and the operative system do not crash.
Main Success Scenario (or Basic Flow):
1. The user chooses to play Training mode from a menu the system presents.
2. The system presents the game map options to the user.
3. The user chooses the game map settings.
4. The user confirms he wants to start a training mode game.
5. The system generates a game map and creates a player from the chosen game map settings.
6. The system starts a training mode game.

Extensions (or Alternative Flows):
1a. User regrets playing at all
    1. The User chooses to quit the application.
3a. User regrets his choice of the game setting.
    1. User chooses to cancel training mode and return to the main menu
    2. User returns to the previous step.
4a. Same as 3a

6.3.3. Host a multiplayer game

Primary actor: User
Stakeholders and Interests:
- User: Wants to create a multiplayer game with other players involved
- Other users: Wants to participate in a multiplayer game

Preconditions: The game application has been launched
Success guarantee (Post conditions): A multiplayer game is created and launched
Minimal guarantees: The operating system remains intact

Main Success Scenario (or Basic Flow):
1. User chooses to play a multiplayer game from the main menu.
2. User chooses to host a game.
3. User enters an alias that he will use during the whole game round.
4. User selects the game settings.
5. User selects the maximum limit of players that can join this game round.
6. User requests to create the game.
7. The system confirms the settings.
8. The system creates the game.
9. The system establishes a lobby where players can connect.
10. The system connects the hosting player to the lobby and marks the player as ready.
11. The system waits for other players to connect.

Extensions (or Alternative Flows):
8a. User regrets his choice of hosting a game
    1. User cancels the game hosting.
2. User returns to the main menu.
    8b. User regrets his choice of the game setting.
    1. User chooses to cancel hosting the game
    2. User returns to Step 4 in main scenario.
8c. No user connects to the host

    1. The user cannot start the game.

6.3.4. A player joins a multiplayer game

Primary actor: User
Stakeholders and Interests:
- User: Wants to play a multiplayer game
- The game host: Wants player to get connected to start a multiplayer game

Preconditions: The user has received an IP address to connect to.
Success guarantee (Post conditions): A multiplayer game is started
Minimal guarantees: The computer and the operative system do not crash.
Main Success Scenario (or Basic Flow):
1. The user requests to the system to initialize a multiplayer game.
2. The system receives a request from the user to initialize a multiplayer game.
3. The system sends a request to the user about in which way a multiplayer game should be initialized.
4. The user requests to the system to join a multiplayer game.
5. The system receives a request that the user wants to join a multiplayer game.
6. The system sends a request to the user to choose a player name.
7. The user types in a player name.
8. The user sends information to the system of his chosen player name.
9. The system receives information about the chosen player name.
10. The system sends a request to the user to type in the host's IP address.
11. The user types in the host's IP address.
12. The user chooses to connect to the written IP.
13. The user sends a request to the system to connect to a written IP.
14. The system establishes a connection between the user and the host.
15. The system displays the multiplayer setup mode.
16. The user sends information to the system to mark him as ready to start.
17. The system receives information that the user is ready to start.
18. The system adds the user to the quantity of ready players.
19. The host sends a request to the system to start the game.
20. The system starts a multiplayer game for the host and all clients.

Extensions (or Alternative Flows):

7a. The user types the wrong name.
    1. User changes his name.

13a. The wrong IP address was entered.
    1. The system sends information to the user that an incorrect IP address was entered.
    2. The system puts the user back one step to write a new host IP.

13b. The host is not responding
    1. The system sends information to the user that an incorrect IP address was entered.
    2. The system puts the user back one step to write a new host IP.

6.3.5. Win a multiplayer game

Primary Actor: User A.
Stakeholders and Interests:
- User A: Wants to win the game
- User B: Wants to play the game

- User C: Wants to play the game
Preconditions: There is an ongoing multiplayer game.
Success Guarantee (Post conditions): User A controls the only remaining village and therefore wins the game.
Minimal Guarantee: The reason why a player did not win the game is stated.

Main Success Scenario (or Basic Flow):
1. User A builds a new soldier-producing building in his village.
2. The system receives a request from user A to build a building.
3. The system builds a solder-producing building in user A's village.
4. User A trains a couple of troops.
5. The system receives a request from user A to build troops.
6. The system starts to train these troops.
7. User A sends all his troops against User B's village.
8. The system receives a request from user A to move his army.
9. The system move user A's army to the chosen position.
10. User A's troops arrives at User B's village.
11. The system calculate User A as the winner of the battle and destroy user B's village.
12. User A is the only player who controls a village.
13. The system present User A is as the winner.

Extensions (or Alternative Flows):

12a. There are still other users except User A that controls a village.
   1. User A continues attacking other players.
   2. User A changes attacking focus to attacking User C
   3. Continue at Step 7 in main scenario replacing User B with User C.

6.3.6. A player leaves a multiplayer game round

Primary actor: User
Stakeholders and Interests:
- User: Wants to leave the current game round.
- Other users: Wants to continue playing the game.
Preconditions: A multiplayer game with a set of users has been initialized. The user is not the host of the game.
Success guarantees (Post conditions): The user has left the game round.
Minimal guarantee: The user's game round has been terminated.
Main Success Scenario (or Basic Flow):
1. The user requests to the system to leave a multiplayer game.
2. The system receives a request from the user to abort the game.
3. The system sends a request of confirmation of the user chooses to abort.
4. The system receives a confirmation on the abortion from the player.
5. The system removes the player's village from the game map.
6. The system asks the user if he wants to watch the rest of the game round or quit to main menu.
7. The user chooses to exit to main menu.
8. The system shows the main menu.
9. User A gets back to the main menu.

Extensions (or Alternative Flows):
6a. The user wants to keep watching the remainder of the game.
1. The user chooses to stay in Observer mode
2. The system displays an overview of the game map.
3. The system shows the other players statistics and results of battles.

6b. The user wants to exit the application
1. The user chooses to leave game round by exiting directly to the operating system.

7. System evolution

7.1. Fundamental assumptions on which the system is based

1. The game will work the same on all computers with JRE 1.5 and windows XP. In other words, the game won't be tested on a variety of different manufactures and setups.
2. The users have internet connection.
3. The users either have Java Run Time 1.5 (JRE) installed or know how to retrieve it.
4. The user can find other users to play against on their own. (The game will not provide any functionality to find other players)
5. The game will *not* be accompanied by any written instructions (manual).

7.2. Anticipated changes

Hardware evolution
Computer games are primarily affected by hardware evolution in the way that people want their games to support and give them benefit of e.g. their new graphics card. This won't affect this game since it's completely without focus on issues that rely on hardware performance. However one important factor can still be identified:

Screen resolution – If users screen resolutions getting higher it will make the game interface look very small since the game will run in the operating systems fixed resolution.

Changing user need
The changes in user needs in computer game in general is that users want more playable options as new characters, new levels and so on. In many commercial games changes in user needs are not considered during the life-time of the game. In other word you don't upgrade or maintains it (except errors etc.) because it will oppose the possibility to release a sequel, e.g. SAD 2. However there are some exceptions, for example online games where you pay per month and never actually buy the game. This is not the case for this game but that might change in the future and some anticipated changes in user need are:

1. Being able to play multiplayer games with more than 6 players.
2. More variety in playable options; resources, troops, buildings, etc.
3. A larger game map (playing field)
4. Users want to play the game in the web browser instead of downloading it.

Software Evolution

If the JRE version 1.5 is not available any longer due to a later versions and that version isn’t backward compatible it will demand a new version of the game that rely on the either the new JRE version or that is completely independent of JRE.

Since the game depends on JRE any future incompatibility between JRE and Windows or any other operating would demand a version of the game that runs independently of JRE in order to work on those operating systems.

8. Index