Defining and Understanding the Problem

# Defining the Problem

Overall, the problem is that a turn-based strategy game is to be developed. Broken down, the problems needing to be solved include:

* The game has a top-view/plan view of a world map. The map is broken up into a grid.
* There are two players, each representing a country. The players are able to name their country before starting the game.
* The main objective is to become the most civilised country by either having the most points or taking over the other player’s country.
* More gameplay rule requirements in the Rules.docx document.
* Educate people to think strategically before making decisions through the playing of the game.

In the development of this software solution, the Structured Approach will be used in conjunction with the Java 7 Standard Edition programming language. This is for the following reasons:

* The requirements of the software solution are very well defined and are not expected to change at all during the development process.
* The client requires that the software solution be of the highest possible quality due to its planned widespread use as an educational tool in educational institutions.
* Due to planned widespread use, it is important that the program can be run on any operating system. Java has a widespread reputation for being able to achieve this (in fact, Java’s goal is “write once, run anyway”), and hence it is the perfect programming language to utilise to achieve the goal of easy widespread use in educational institutions.

However, some elements from the prototyping approach will be used in order to maximise communication and feedback from the client to ensure that the software solution is built to the client’s exact requirements and expectations. These will include:

* Regular compiling of the software solution for testing by both the development team and the client.
* Maximising accessibility for the client to oversee the development process.
* Regular communication with the client, both formal informal.

To do this, the development team will employ the use of various CASE (computer-aided software engineering) tools to achieve the goal of creating the best possible software solution. These include:

* **Netbeans IDE 8.0** – An integrated development environment that streamlines the process of software development process using various tools, such as colour coding specific keywords, management of files, syntax checking and checking of the code to minimise the risk of logic and runtime errors.
* **Adobe Creative Cloud Suite** – A collection of industry-leading creative tools. In the development of this software solution, the programs *Adobe After Effects CC* and *Adobe Photoshop CC* will be used to create assets for the game. This will be done to ensure that the assets are of the highest quality, and to ensure that there is no copyright infringement occurring that may cause legal issues when the solution is released to the public.
* **Dropbox** – The industry-leading cloud-storage service. This will be employed to make regular backup of the software solution assets and code to ensure that any catastrophic loss of data will not create a significant setback to the software development process.
* **GitHub** – A Git repository services that provides powerful version control. This will be used to create regular versions of the program and allow for public review by any software developer. In addition, it will be used as a communication tool with the client, where the client can easily check on the progress of the software development. It also acts similarly to Dropbox in that it allows the cloud-storage of the software solution’s assets and code. This creates a double redundancy to the software development process, effectively eliminating any chance of permanent loss of data. The software development project will be hosted here: <https://github.com/WhalesAreDelicious/Conquest2>.

# Design Specifications

## Data Dictionary

See Data Dictionary Excel spreadsheet.

## Criteria

In the development of the software solution, the perspectives of both the software development team and the client must be considered.

The general requirements of the software solution are:

* **Graphical**
  + Users must be able to use all controls using the GUI
  + GUI must be well organised and constant.
  + GUI elements must function as intended.
* **Gameplay**
  + **Map:**
    - The map must be subdivided into a two-dimensional array of locations, similar to a grid.
  + **Unit Soldiers:**
    - Unit soldiers must be able to move in any unoccupied area.
    - Unit soldiers must not be able to move into spaces occupied by other units, or blocked areas.
    - Unit soldiers must not be able to exit the battlefield.
    - Unit soldiers must have a power/health system.
    - Unit soldiers must be able to attack nearby enemy unit soldiers (1 square grid away).
    - Unit soldiers must be able to capture control points if they are on it.
    - Moving and upgrading units will cost 1 move each.
  + **General:**
    - Moves can only be gained by “rolling a dice” that returns a random integer between 1 and 6.
  + **Control points**
    - If all control points are controlled by one player, the game ends and that player wins the game.
    - There will be 3 or 5 control points (will be decided later).
    - In order to capture a control point, a unit must be standing in it and click on the capture button. If the player click on capture button if the selected unit is not in a control point, no capture will be done.

### User’s perspective

As the users will be group that will be primarily running the software solution, it is important to put the needs of the users first before the needs of the development team. The following design specifications have been made to ensure that the needs of the users will be met:

* **Interface design** – The graphical user interface will be the primary method of interaction between the user and the software. The user interface will be designed to be as easy to use as possible through the use of consistent screen design. In order to maximise utility of screen space, the actual game will comprise of two windows: the control window and the map window. This will ensure the user will have a full, uncluttered view of the map when needed.
* **Appropriate messages** – The user will need to be able to understand what the software will be displaying. Because of this, all dialog boxes will be kept to a minimum and only used when it is absolutely necessary. In addition, information displayed will be simple and easy to understand, so that any user with a basic understanding of the English language will be able to use the software.
* **Appropriate icons** – The use of icons will be used to complement the need for appropriate messages to ensure that the user will be able to easily understand all aspects of the software during usage.
* **Relevant data formats for display** – Care must be taken to ensure that the user is able to read the data given to him/her during the use of the program. This will be done by ensuring that information is displayed in a simple format in a consistent location.
* **Ergonomic issues** – The user will need to be able to comfortably utilise the program. This is to ensure maximum satisfaction and minimise any physical or psychological injuries or stress. This will be done by placing buttons in consistent locations, the ability to exit the program at any point in time by clicking on the “X” button at the top right corner of the screen in Windows and its equivalent counterparts in Mac and Linux; and placing the buttons in close proximity to each other to minimise physical and psychological stress.
* **Relevance to the user’s environment and computer configuration** – Due to the planned widespread use in many educational institutions as an educational tool, care must be taken to ensure that this software solution is able to be run on a wide range of computer configurations and environments. This is one of the main reasons for the use of the Java programming language, which allows programs developed in the language to be run on any operating system with the Java interpreter. This means the software is able to be run on all Windows, Mac/OS X, Linux and Solaris operating systems, which are used in over 99% of computers in the world. In addition, the nature of the software solution developed means that any basic computer with the ability to render images, utilise mouse input have at least 256MB of RAM, 15MB of storage space, a single-core 1GHz CPU, run either Windows, Mac/OS X, Linux or Solaris, and a 800x600 monitor display will be able to run the software.
* **Social and ethical issues** – Care must be taken to minimise the chance that any user will be affected by social and ethical issues, such as sanctions by governments, restrictions imposed by religion, impaired use impaired usage due to a disability, and natural disasters as a result of the use of the software solution. This will be done by making the software solution as open as possible through the ability for any member of the public to review the source code and assets of the software solution on GitHub. If a potential social and ethical issue is found, the reviewer is able to easily contact the development team through GitHub to suggest and recommend changes to the software to accommodate that issue. In addition, users affected by disabilities will be able to use the software through the employment of various techniques, such as relatively high contrast between the letters and the words in the control window, and the use of colourblind-safe colours in distinguishing units between players.

### Developer’s Perspective

The development team will need to ensure that the software solution is well optimised and easy to maintain during the development of the solution. In other words, the developer needs to ensure that the program works as intended “behind-the-scenes”. As a result, the following design specifications have been made to ensure this:

* **Data types** – Appropriate data types must be used to minimise excessive system resource requirements and to maximise speed in running the software. For example, a variable that only requires relatively small, whole numbers for its successful use should use the 16-bit integer data type.
* **Data structures** – Some data can be grouped together for better storage and management. This should be done whenever possible to maximise maintainability and expandability. For example, an x-y coordinate system should be stored as a two-dimensional array.
* **Algorithms** – To ensure maximum efficiency of code, all algorithms should be written beforehand, desk-checked, and then individually tested on the computer before being implemented into the software solution. This will be done in the Planning the Solution phase of the software development process.
* **Variables** – To ensure that the development team is able to quickly find the role that each variable has in the software solution during implementation, testing and maintenance, a naming convention will be used. For example, variables with the string data type will have the prefix “string”, followed by its name (e.g. stringPlayer1Name). In addition, a data dictionary will be created and maintained for easy reference during maintenance and testing.
* **Software design approach** – The development process will require the use of the Structured approach, with some elements of the Prototyping approach in order to achieve and meet the needs of the client. This is further explained on page 1 in the section “Defining the Problem”.
* **Quality Assurance** – To ensure the maximum quality of the software solution, all aspects of the software is planned before-hand and checked before implementation. This will be done through the creation of a refined design specification, screen design, algorithm desk-checking etc.
* **Modelling the system** – To ensure that both the client and the developer understand the scope of the program and what will happen “behind-the-scenes”, various developer tools such as diagrams will be used to model the system. This includes flowcharts, IPO (input-output) charts and data-flow diagrams etc.
* **Documentation** – It is in the best interests of the software development team, future maintainers of the software solution and the client to ensure that the development of this software solution is well documented to ensure that the software is as easy to maintain as possible. This is the reason for the creation of this document.

# Communication

To ensure that the needs of both the client, users and developer are met, formal and informal communication methods will need to be implemented.

## User/Client

Users will be able to contact the software development team of any issues they may have with the software solution through various methods. These include sending messages through the internet via the software development team’s website (<https://whalesaredelicious.github.io>) and calling the software development team through the dedicated hotline (1800 696 969).

The client is also able to contact the software development team formally through regular telephone calls weekly. In addition, the client will be able to organise more frequent formal communication if reasonable. Informal communication with the client is done through the instant-messaging system Skype between the client and members of the software development team, as well as being able to review the code and assets themselves through GitHub. A change log will be available here: <https://github.com/WhalesAreDelicious/Conquest2/commits/master>.