CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF INFORMATION TECHNOLOGY

ASSIGNMENT OF BACHELOR'S THESIS

Title: RPG game with augmented reality features - server part
Student: Jakub   ech
Supervisor: Ing. Miroslav Balík, Ph.D.
Study Programme: Informatics
Study Branch: Software Engineering
Department: Department of Software Engineering
Validity: Until the end of winter semester 2018/19

Instructions

The aim of the thesis is to specify, design, and implement a functional prototype of the server part of a RPG game with features of augmented reality (AR).
1. Create a story line and rules for the game. Consider intensive usage of geolocation and AR features.
2. Formalize the following requirements for the implementation of the server part:
  - the data layer consists of a database engine and a caching,
  - users can use their Google accounts to login and play,
  - the client part provides an API,
  - the communication between client and server parts of the application must be secure.
3. Design the server part of the game.
4. Design a suitable front-end part for server administration.
5. Discuss and choose a suitable implementation platform and related technologies (databases etc.).
6. Implement the functional prototype, document it, and perform suitable testing.
7. Tightly cooperate with Tomáš Zahálka who works on the client part.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.   prof. Ing. Pavel Tvrdík, CSc.
       Head of Department   Dean

Prague March 6, 2017

CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF INFORMATION TECHNOLOGY

DEPARTMENT OF SOFTWARE ENGINEERING

Bachelor's thesis

Location-based Role Playing Game

Jakub ech

Supervisor: Ing. Miroslav Balík, Ph.D.

June 3, 2017

Acknowledgements

I would like to thank myself for doing this. I am an awesome and humble person. With great power comes great responsibility and no one else is as good or worthy as I am to be thanked. Ave Kuba!

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on June 3, 2017   . . . .

Citation of this thesis

Čech, Jakub. *Location-based Role Playing Game.* Bachelor's thesis. Czech
Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Hvzdy jsou krásné, protoe je na nich kvtina, kterou není vidt. Pou je krásná práv tím, e nkde skrývá studnu. A je to dm, hvzdy nebo pou, to, co je dlá krásnými, je neviditelné!

Klíčová slova   #deep, #thoughtoftheday, #follow4follow

Abstract

Place the 2 cups of crushed ice into a cocktail shaker. Pour the rum, lime juice, and simple syrup over the ice, cover, and shake well. Remove the ice from your serving glass and strain the drink into it. Serve immediately.

Keywords   Daiquiri, Coctail, Rum, Cuba

Contents

List of Figures

Introduction

The world of mobile devices is quickly evolving. Smartphones and tablets are becoming more and more powerful and not only in terms of computational power and available memory. Mobile devices nowadays are packed with various sensors. It is possible to integrate data from GPS (Global Positioning System), accelerometer, gyroscope, magnetometer, and camera to quickly determine devices position and orientation. This opens us doors to augmented reality. In my bachelor thesis, I, in cooperation with Tomá Zahálka, will create a mobile game for Android which will utilize elements of augmented reality. The game is set in a fantasy world and its genre is RPG (Role Playing Game).

CHAPTER **1**

Project overview

1.1 Project timeline

1. **Analysis** – finished by the end of February

2. **Design** – finished by the end of March

3. **Implementation** – finished in the middle of May

4. **Testing** – begins with implementation and will be finished by the end of May

5. **Releases**

   a) **Pre-alpha** - 16. 4. 2017
   b) **Alpha** - 30. 4. 2017
   c) **Beta** - 14. 5. 2017
   d) **Final** - 1. 6. 2017

1.2 Specification and features

The content of this section is temporary.

1. Bare minimum

   — Zobrazení pohybu hráe po map
   — Generování monster
   — Generování budov
   — Interakce s herními objekty
   — Ukladaní postupu, stavu postavy
   — Inventá

— Statistiky

2. Should be there

   — Mise, úkoly

   — Multiplayer

   — Skill systém

   — Dialogy

3. Nice to have

   — Navigace k objectives

   — Trading

   — Tutorial

   — ástená offline podpora

CHAPTER **2**

Analysis

2.1 Similar solutions

2.1.1 Parallel Kingdom - Age of Ascension

This game was on market for 8 years (2008-2016). Parallel Kingdom is a closest solution to ours.

Parallel Kingdom is a mobile, location based, massively multiplayer game that uses GPS location and Google Maps to place users in a virtual world. Parallel Kingdom is the first location based RPG for the iOS and Android platforms. The game is set in a virtual world or Parallel Kingdom where users claim their territories based on their GPS location or by making friends who invite them to travel to new places. Parallel Kingdom is a freemium game and utilizes a virtual goods revenue model.

2.1.2 Ingress

Developed by Niantic, which was then part of Google, this game was released in 2013 for Android and in 2014 for iOS.[4] It is a location based, massively multiplayer game. A player have to choose one of the two factions, Enlightened or Resistance, and then as a part of his team capture regions of the game map. A faith of each faction relies on players cooperation. Thanks to that players meet in real life and coordinate their actions.

Ingress was the first very successful augmented reality game with more than 10 000 000 installs.

2.1.3 Pokémon GO

After its success with Ingress, Niantic started working on a new game Pokemon GO. Once released, the game became incredible hit. Even though the game faced many problems during its launch, mainly caused by the unexpected success and more active users than Pokémon GO was able to handle, in the

first 80 days Pokémon GO reached about 550 millions downloads and earned about \$470 million.

The game is very similar to Ingress and uses the same crowd-sourced geographical data.
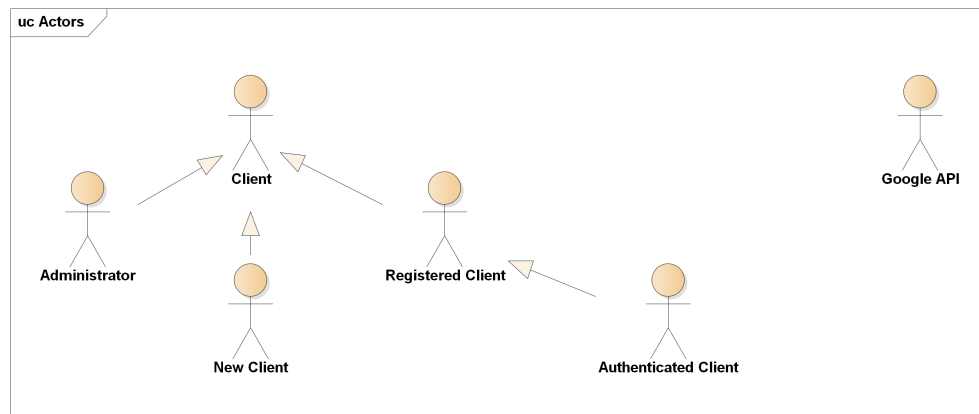
2.2   Use Cases

2.2.1   Actors

Figure 2.1: Use Case Diagram – Actors

Actor is a role played by a user or other system that interacts with the server. The most general role is *Client* and anyone who accesses the server through API is considered to play either this role or any of its children. From now on, the terms client and user will be used interchangeably. Refer to Figure 2.1 for the role hierarchy.

Administrator is a client who has privilege to create and maintain the functionality of the game. *New Client* is a user who is not yet registered and probably accesses the game for the first time. *Registered Client* is the default role for a user who already has a valid account but is not logged in. Lastly, *Authenticated Client* has all the required privileges to play the game. This client will be refereed to as *player*. The last main actor is Google API which provides the access to Google services.

2.2.2   Authentication

Authentication use cases user performs to get promoted to more privileged roles. The basic transition flow is *New Client → Registered Client → Authenticated Client*. See Figure 2.2.
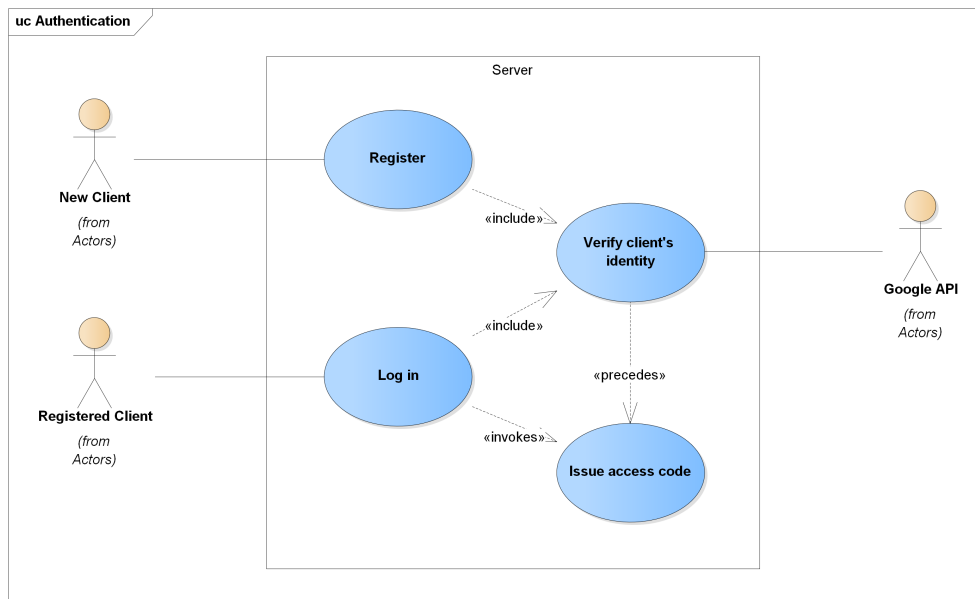
Figure 2.2: Use Case Diagram – Client's authentication

— **Register**
  The only choice *New Client* has is to register a new account. He uses provides his Google identity and chooses an username. After his identity is verified, the server creates a new profile and logs the user in.

— **Log in**
  The *Registered Client* must log in before he can access any of the game features. He provides his Google identity which is then verified and matched to an account. An access code is issued for the future identification within the session.

— **Verify user's identity**
  Proper verification is needed when the game server receives a Google identity of a user. The server contacts the *Google API* which responds with user's personal information if the identity is valid.

— **Issue access code**
  This use case is invoked during login process. Server issues a unique access code to the user. Only *Authenticated Client* has such code.

2.2.3 Actions

Action is an event triggered by player's interaction with a game object. See the diagram in Figure 2.3.
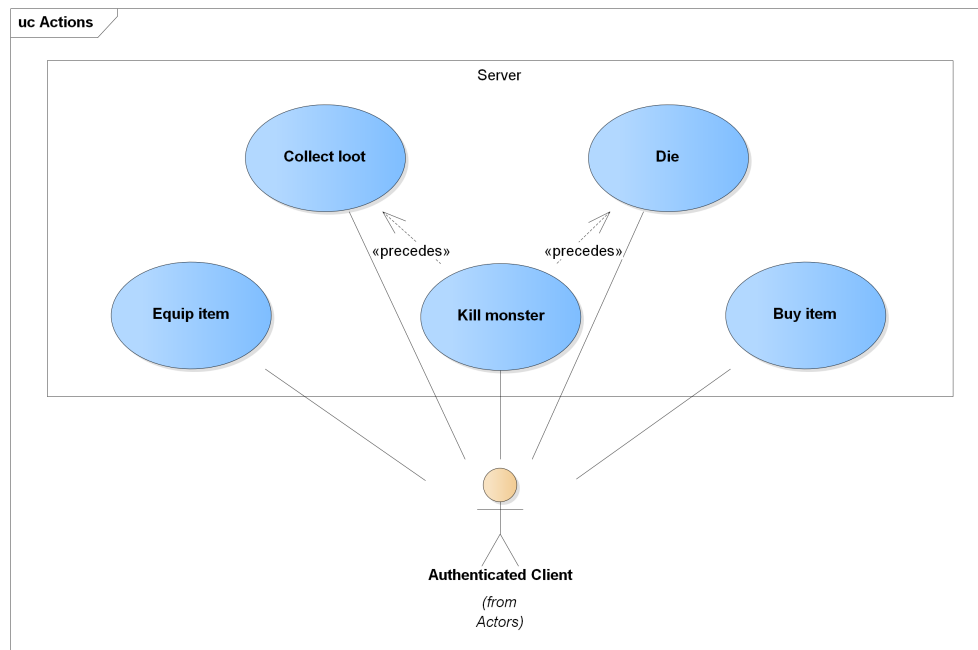
Figure 2.3: Use Case Diagram – Player's actions

— **Equip item**
A player wants to equip an item from his inventory. The item will be assigned to a specific slot. For example the player equips a sword to his right hand.

— **Buy item**
A player wants to exchange gold in a shop for an item he chooses. When the purchase is finished, the player receives the item to his inventory.

— **Kill monster**
A player wants to kill monsters to progress in the game. If he successfully kills the monster, he is rewarded with gold and experience

— **Collect loot**
This action must be preceded by the *Kill monster* use case. A player can choose to collect loot from the monster he killed.

— **Die**
This action must be preceded by the *Kill monster* use case. A player who lost his fight against a monster dies and is punished with some penalty.
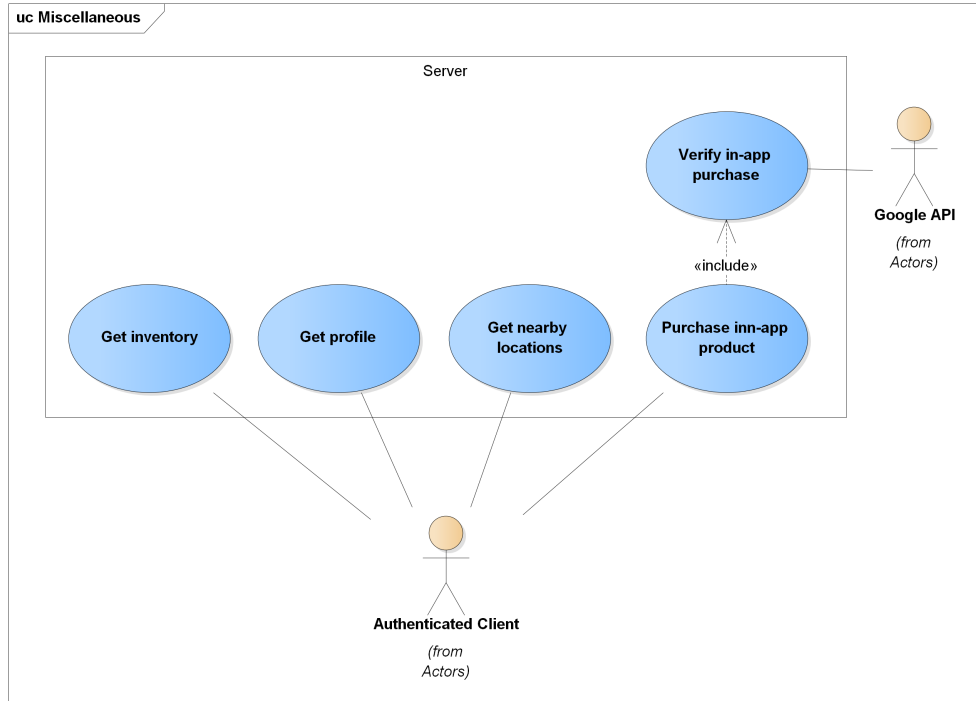
2.2.4 Miscellaneous

Figure 2.4: Use Cases: Miscellaneous

The following use cases mainly cover requests made by client application. See the diagram in Figure 2.4.

— **Get inventory**
The application requests all items the player owns. It is also possible to retrieve all the player's equipment with each item assigned to some slot.

— **Get profile**
A client may need to synchronize its internal state of the players profile with the server. The application is provided with complete state of the player's profile including attributes like health, gold, and experience.

— **Get nearby locations**
This is a critical functionality of the server. A client asks for game locations near his coordinates. The server provides such locations along with their associated game objects.

— **Purchase in-app product**
The application supports micro-transactions. Since the purchase is made

client-side, the application has to notify the server about the purchase
and give the bought product to the player.

— **Verify in-app purchase**
  During the previous use case (*Purchase in-app product*), the server veri-
  fies if the purchase is valid and not fake, canceled, or already accounted.
  The verification process contacts *Google API* which confirms the valid-
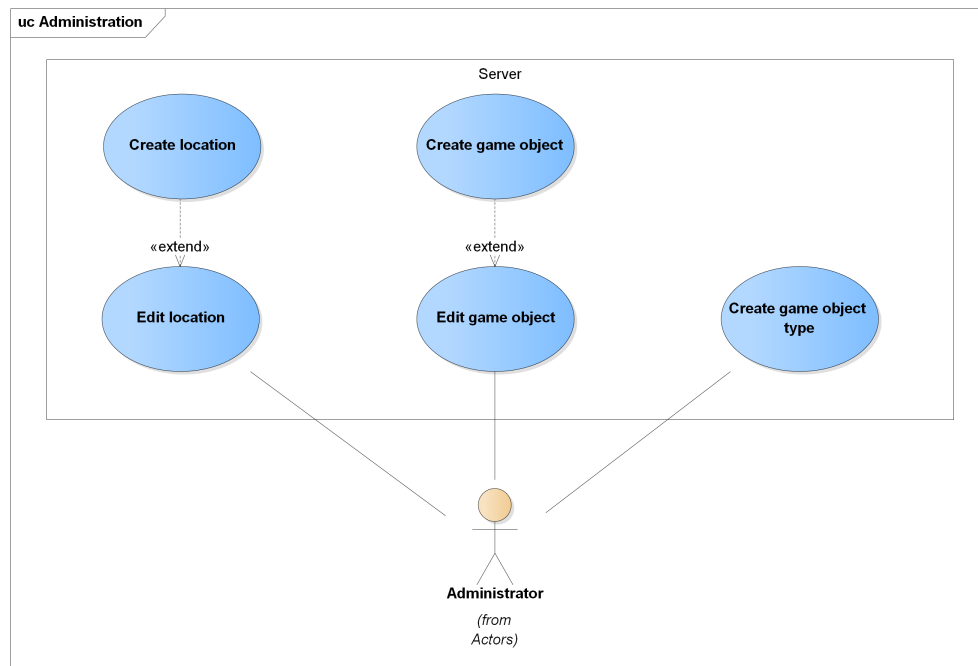 ity.

2.2.5   Administration

Figure 2.5: Use Cases: Administration

Since the game is not static during its lifetime, people in charge of the changes
needs an easy way to add new locations and maintain game objects. These
use cases are described in Figure 2.5.

2.3 Requirements

2.3.1 Functional Requirements

Rules

1. **The server must provide API to clients**
   The key requirement for the server is to allow receiving HTTP(S) requests. When processed, the server responds in JSON format.

2. **The player's character has attributes**
   The character has a set of attributes, including health, experience, level, and owned gold. Maximum health increases with level. The experience is rewarded after certain actions, e.g. after killing a monster. The gold is primary in-game currency.

3. **A player can own items**
   A player has an inventory which can contain various types of items. The item can be for example sword, potion, armor etc.

4. **A game object has a type and inherits all its properties**
   The type of game object specifies allowed actions, its attributes, default name and description.

5. **A game object can be a monster**
   The monster can be killed but can also inflict damage to the player. It has its own inventory and there's a reward for killing the monster in a form of gold and experience.

6. **A game object can be a shop**
   The shop can contain several items with specified price.

7. **A game object can be an item**
   The item can be one of the many objects useful to a player. Examples of the items are health potion, sword, armor, necklace and similar.

8. **Each game object has its own inventory**
   The inventory contains other game objects. Example of this requirement is a monster with a potion and a sword in its inventory; both will be given to the player who kills the monster.

9. **The server stores a list of predefined locations**
   Real geographic locations for the game objects are stored on the server to ensure every player has the same location-object pair.

10. **A game object can independently exist at many locations**
    This requirement aims to help maintain the game objects efficiently by administrators. It allows creating small set of abstract game objects with predefined inventories and other attributes.

11. **If a player kills a monster at a location, the monster will be hidden for a period**
To prevent the player from killing the same monster continuously without a need of moving somewhere else, the location should be hidden for a certain period after the kill.

12. **The server should provide API for administration**
Such API will be used to manage locations, create and edit game objects or to assign a game object to some locations. It is necessary to protect the administration endpoints from unauthorized access.

13. **The server must persist players profile between sessions**
All the player's attributes, his inventory and equipment must be stored between sessions. Player will continue from the state in which he ended.

Features

1. **A player registers and logs in the game using Google account**
For the player's convenience, a Google account is required to play. The server does not have to store or handle any password. Most of the authentication process is delegated to Google servers.

2. **A client can get nearby game objects based on his location**
The major feature of this application is being location-aware. Server must provide a method to retrieve game locations near the requested latitude and longitude. The "near area" should be circular, defined by its radius; the size have to be carefully chosen so it's big enough to cover client's maps but also small to limit the response size and the spatial search overhead.

3. **A player can kill a monster**
When the player wins the fight, he will be rewarded by experience and gold.

4. **A player can be killed be a monster**
The player can lose health during the fight with a monster. If the health reaches zero, the player dies and loses an amount of gold based on his level.

5. **A player can collect items from the monster he killed**
When the player wins the fight, he's offered to collect items from the monster's inventory. He can chose any subset of these items.

6. **A player can equip an item**
Many items in the game can be equipped. These items have predefined equipment slot, for example a sword have to be held in hand, an armor worn on chest, shoes put on feet and so on.

7. **A player can buy object from a shop**
   Gold can be exchanged for various items in shops.

8. **A player can use an item from his inventory**
   Some items in the game are consumables. When used, an action defined by the item is executed. For example a health potion heals the player.

9. **A player can purchase in-app product**
   The application allow a user to exchange real-life currency for the in-game one. The server should verify such purchase and add the currency to his profile.

2.3.2 Non-functional

1. **The data layer consists of a database engine and a caching**
   Game Server never accesses database directly. Data are retrieved from the database upon request and then cached. Upper layers communicate exclusively with cache.

2. **The server provides an API for client**
   The API support at least following:

   a) Retrieve nearby game locations

   b) Create player

   c) Update player's data (inventory, experience, resources, quests progress etc.)

   d) Provide information about a quest

3. **The communication between client and server parts of the application must be secure**
   All data sent from and to a client has to be encrypted.

4. **Client can only connect to a Connection Server**
   Several Connection Servers exist to prevent a bottle-neck. Client selects the Connection Server by an algorithm. Client does not have an access to any other part of the server.

2.3.3 System and Interface

1. **System uses Java 8 SE as an execution environment**

2. **Operating system for the server is Debian 8**

3. **Database engine is MySQL**

4. **Cache engine is Redis**

2.4 Technology

2.4.1 Frameworks

Design

3.1 Server-Client communication

How the data are exchanged between client and server. Will contain format, technologies and protocol specification.

3.2 Security

Basic methods for verification of legitimacy of the data. How the server API will be secured.

3.3 Architecture

Diagrams and overall architecture of the server belongs here.

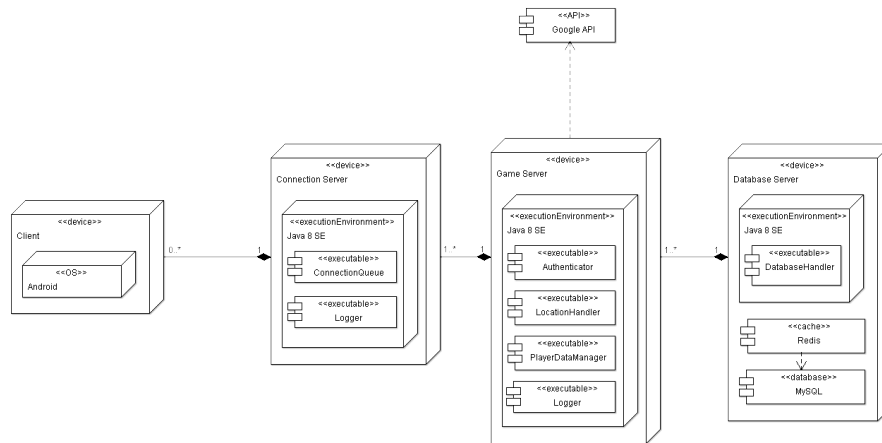## 3.4 Deployment Diagram

3.5 Database Model

Figure 3.1: Deployment diagram for the game

Figure 3.2: Diagram of the database structure implemented on the server

Geo-data mining

Description of how the geographical data for the monsters and building will be obtained.

4.1   Methodology

4.2   Output

Implementation

Implementation-specific aspects of the game. Will include a description interesting parts of the server.

Deployment

Testing

7.1 Unit testing

7.2 Pre-alpha version

7.3 Alpha version

7.4 Beta version

7.5 Release

7.6 User feedback

Results

APPENDIX **A**

Acronyms

GUI Graphical user interface

XML Extensible markup language

Contents of enclosed CD

```
readme.txt........................the file with CD contents description
exe ................................... the directory with executables
src.......................................the directory of source codes
    wbdcm......................................implementation sources
    thesis.............the directory of LaTeX source codes of the thesis
text ....................................... the thesis text directory
    thesis.pdf...........................the thesis text in PDF format
    thesis.ps............................the thesis text in PS format
```