

**Developing a Game Engine for the Creation
and Management of Alternate Reality
Games**

Sam Hogarth (071944278)

BSc Computing Science, May 2010

Supervisor: Dr. C. Kray

Word Count: 15,101

Abstract

This dissertation researches the effectiveness of introducing game engine technology into the development cycle for Alternate Reality Games, in order to reduce the significant workload placed upon developers.

A summary of the current research within this area is presented, and the development of an Alternate Reality Game Engine is documented. The author's hypothesis is tested by a user study comparing the developed system with current methods of Alternate Reality Game creation and management. The results and subsequent analysis are presented within this dissertation.

Declaration

“I declare that this dissertation represents my own work, except where otherwise stated.”

Signed: _____

Dated: _____

Acknowledgements

I would like to thank my supervisor, Dr. Christian Kray, for the advice and support provided during the course of this project.

Table of Contents

ABSTRACT	2
DECLARATION	3
ACKNOWLEDGEMENTS.....	4
CHAPTER 1: INTRODUCTION	8
1.1: INTRODUCTION.....	8
1.2: DISSERTATION STRUCTURE	8
1.3: DEFINITION OF PROBLEM	8
1.4: HYPOTHESIS.....	9
1.5: PROJECT AIMS.....	9
1.5.1 AIM:.....	10
1.5.2 OBJECTIVES.....	10
1.6: SYSTEM REQUIREMENTS	10
1.6.1 CHARACTER MANAGEMENT.....	10
1.6.2 STORYLINE MANAGEMENT	10
1.6.3 EVENT MANAGEMENT	11
1.6.4 CHARACTER INTERACTION HANDLING.....	11
1.7: PLAN.....	11
1.8: SUMMARY.....	11
CHAPTER 2: BACKGROUND RESEARCH	12
2.1: INTRODUCTION.....	12
2.2: DEFINING ALTERNATE REALITY GAMES.....	12
2.3: PROBLEMS WITH ALTERNATE REALITY GAMING	13
2.3.1 CREATING THE GAME STORYLINE	13
2.3.2 DETECTING PUZZLE SOLUTIONS	15
2.3.3 HANDLING CHARACTER INTERACTIONS	15
2.4: GAME ENGINES	16
2.4.1 CREATING THE GAME STORYLINE.....	16
2.4.2 DETECTING PUZZLE SOLUTIONS.....	16
2.4.3 HANDLING CHARACTER INTERACTIONS	16
2.5: SIMILAR SYSTEMS	17
2.5.1 THE ARGOSI FRAMEWORK.....	17
2.5.2 XENOPHILE REACTOR.....	17
2.6: SUMMARY.....	17
CHAPTER 3: SYSTEM DESIGN AND IMPLEMENTATION.....	19
3.1: INTRODUCTION.....	19
3.2: SYSTEM ARCHITECTURE	19
3.3: SOFTWARE DEVELOPMENT MODEL.....	19
3.4: TOOLS AND TECHNOLOGIES USED	20

3.4.1	PHP	20
3.4.2	ZEND FRAMEWORK	20
3.4.3	SMARTY	21
3.4.4	MYSQL	21
3.4.5	XAMPP	21
3.5:	DATABASE COMMUNICATION	22
3.6:	USER INTERFACE	22
3.6.1	LAYOUT.....	22
3.6.2	ITEM SELECTION	23
3.6.3	CHARACTER DESCRIPTION PAGE	24
3.6.4	ARC DESCRIPTION PAGE.....	25
3.6.5	EVENTS DESCRIPTION PAGE	26
3.6.6	TABLES.....	27
3.6.7	FORMS.....	27
3.6.8	NOTIFICATIONS	28
3.6.9	BUTTONS.....	28
3.7:	CHARACTER MANAGEMENT	29
3.8:	PLUG-INS	30
3.8.1	WEB SERVICES	31
3.8.1.1	Linking	32
3.8.1.2	Communicating	33
3.8.2	PUZZLES	34
3.9:	GAME CREATION	35
3.9.1	LINKING ACCOUNTS AND CHARACTERS	35
3.9.2	EVENT SCHEDULING	37
3.10:	GAME MANAGEMENT	42
3.10.1	POSTING EVENTS	43
3.10.2	SOLUTION DETECTION	45
3.11:	CHARACTER INTERACTION HANDLING.....	47
3.12:	SUMMARY.....	48
CHAPTER 4:	TESTING.....	49
4.1:	INTRODUCTION.....	49
4.2:	UNIT TESTING	49
4.2.1	DATABASE COMMUNICATION	49
4.2.2	CONFIGURATION FILE	49
4.2.3	PLUG-INS.....	49
4.2.4	GAME MANAGEMENT	50
4.2.5	CHARACTER MANAGEMENT.....	50
4.2.6	STORYLINE MANAGEMENT	50
4.2.7	RESULTS	50
4.3:	USER TESTING	50
4.3.1	TEST DESCRIPTION	50
4.3.2	QUESTIONNAIRE DESIGN.....	51
4.3.2.1	About ARGs	52
4.3.2.2	Creating ARGs.....	52
4.3.2.3	Overall comparison	53
4.3.3	RESULTS	53
4.4:	RESULTS ANALYSIS	59
4.5:	CHAPTER SUMMARY	65

CHAPTER 5: CONCLUSION AND PROJECT EVALUATION	66
5.1: INTRODUCTION.....	66
5.2: CONCLUSION.....	66
5.2.1 SATISFACTION OF THE AIM AND OBJECTIVES.....	66
5.2.1.1 Aim	67
5.2.2 OVERALL CONCLUSION	67
5.3: EVALUATION.....	67
5.3.1 TIME PLAN	67
5.3.2 WHAT HAS BEEN LEARNT.....	68
5.3.3 WHAT WENT WELL	68
5.3.4 WHAT COULD HAVE BEEN DONE BETTER	68
5.4: FURTHER WORK	69
5.5: SUMMARY.....	69
<u>REFERENCES.....</u>	<u>71</u>

Chapter 1: Introduction

1.1: Introduction

This chapter introduces the overall structure of this dissertation, the motivation behind this project by discussing the author's hypothesis, and how this provided the project aims and objective. Finally, the system requirements will be specified based upon satisfying the objectives.

1.2: Dissertation Structure

Chapter One details the motivations behind this project, identifying the main themes and current issues that led the author to form a hypothesis. The nature of how this dissertation will address this hypothesis is presented as an aim, several objectives and a set of system requirements.

Chapter Two provides a detailed review of the background research obtained by the author, exploring the themes and issues identified in Chapter One and their relevance to this project.

Chapter Three discusses how the design of the developed system was relevant in order to meet the system requirements. This will take the form of a review of the system design, a detailed view of the core system elements, the design and project management techniques used during the system development and also the technologies employed in order to realise the system.

Chapter Four outlines the nature of the testing carried out in order to address the hypothesis. A detailed analysis of the test results will identify the issues that the system has addressed and not addressed.

Chapter Five provides a conclusion based on the evidence gathered by analysis to provide an answer to the hypothesis. Furthermore, the project will be evaluated, in which the author will critically review the entire proceedings of this project to identify areas that are suitable for future work, and sections which could be improved.

1.3: Definition of Problem

Alternate Reality Games (ARGs) use the context of the real world to establish a narrative. Fictional entities exist within this narrative, such as characters, corporations and objects. Using the framework of the real world, players can interact with entities within the Alternate Reality. They are free to do so using any interaction media supported by the real world. For example, game characters may possess working email addresses, telephone numbers and websites.

The 'game' aspect is introduced by challenges being set to players via entities within the Alternate Reality. By completing the challenges, the storyline is advanced and new content is revealed.

ARGs vary in scope. Television production companies deploy high-budget, large-scale ARGs to fit within the narrative of an existing television show. There is also a 'grassroots' community who create smaller-scale ARGs. The development of ARGs can be categorised into two phases:

- Creation – Preparing the game by developing character profiles, storylines, content and puzzles.
- Management – Handling character interactions and managing the flow of the game whilst in progress.

These two distinct phases pose three main categories of issues, which may restrict the development of ARGs by imposing time constraints upon developers. Each of these issues will be covered in detail in Chapter Two.

- Creating the game storyline is a complicated procedure as it requires a vast amount of planning, preparation and modelling to develop an understanding of the narrative.
- Developers rigorously search for solutions to puzzles. Developers must also respond to puzzle solutions (by releasing new storyline content) with immediacy, so player interest in the game does not wane.
- Players are able to freely interact with the characters. This may be either relevant or irrelevant to the game. Depending on the nature of interaction, immediacy of response may be required. Developers, therefore, must also allocate some of their in-game time to replying to interactions.

Video games are typically built upon a game engine - a re-usable framework of components, which significantly reduces development times by providing developers with general-purpose implementations of commonly-used functionality. Typically this involves providing components responsible for graphics rendering, physics and level design.

1.4: Hypothesis

Based on the game engine mantra described above, it is the author's hypothesis that introducing a game engine with components suited for ARGs will increase the ease of creating and managing an ARG.

The main issues identified above restrict the ease at which ARGs can be developed. Providing a re-usable solution to these issues reduces the time developers have to spend overcoming such issues in practise. In terms of the financial cost of developing an ARG, providing a solution to these main issues constitutes a lower associated risk for investment within such projects. Additionally, providing an easier development method may encourage more people to try developing and ARG. For example, by providing a re-usable implementation of a storyline-planning component, developers would be able to quickly build a mock-up and conduct a test run, to identify potential issues with their narratives. The developer's in-game workload can be reduced by providing components to handle specific character interactions with a degree of automation; or to provide automatic solution-detection for puzzles.

1.5: Project Aims

In order to provide an answer to the hypothesis, this project will be based upon an aim. This is further deconstructed into specific objectives, directly relating to the requirements of the proposed system.

1.5.1 Aim:

The aim of this project is to develop a game engine to allow for the creation and management of Alternate Reality Games.

1.5.2 Objectives

In order to achieve the aim described above, the following objectives have been specified:

Objective One: Provide components within the system aimed at solving each of the main issues highlighted in Chapter Two.

The main issues regarding creating and managing a game act as the main restrictions for securing investment for developing ARGs, or for newcomers to the genre to begin their own development projects. By providing a solution to these main issues, the feasibility of developing ARGs is increased; as introducing a stable, re-usable framework reduces the risk associated with ARGs.

Objective Two: Develop a plug-in architecture so that new puzzles and web services can be integrated into the engine.

The engine must be scalable such that it can incorporate any web service or puzzle type. Due to time constraints, only a selection of web services and puzzle types will be implemented. Developing a plug-in environment provides the system with a means of achieving scalability.

Objective Three: Evaluate the system on accessibility, game development efficiency and effort required by inviting users to test the application.

In order for the system to be adopted in practise it must be demonstrated to be better than the current alternative methods. A user study comparing such methods with the system will be performed. Analysis of the results obtained from this study provides significant evidence towards confirming or denying the hypothesis.

1.6: System Requirements

In order to achieve the objectives outlined above, the following requirements will specify the deliverables of this project. These are split into various sections, relating to components of the system.

1.6.1 Character Management

1. The system must store game characters.
2. The system must allow for conceptual groupings of characters.

1.6.2 Storyline Management

3. Plot segments are revealed from the character's perspective.
4. The system must allow for conceptual grouping of plot segments.
5. Plot segments are represented as events, a combination of content to post and a puzzle.
6. Each conceptual grouping is represented by a flow chart.

7. When a game is in progress, completed events and the current event must be highlighted on the flow chart.

1.6.3 Event Management

8. The system must be able to communicate with any web service.

9. The system must be able to interact with any puzzle.

10. The system is responsible for automatically posting events when a game has begun.

11. The system is responsible for searching for solutions to the active puzzle.

12. When a solution is detected, the narrative is advanced and the next event is triggered.

1.6.4 Character Interaction Handling

13. The system must be able to interact with users via any medium.

14. When replies to events are detected, the system may form replies or pass the reply to the developers; as stipulated by the developers.

1.7: Plan

Figure 1 demonstrates the proposed time plan for this project.

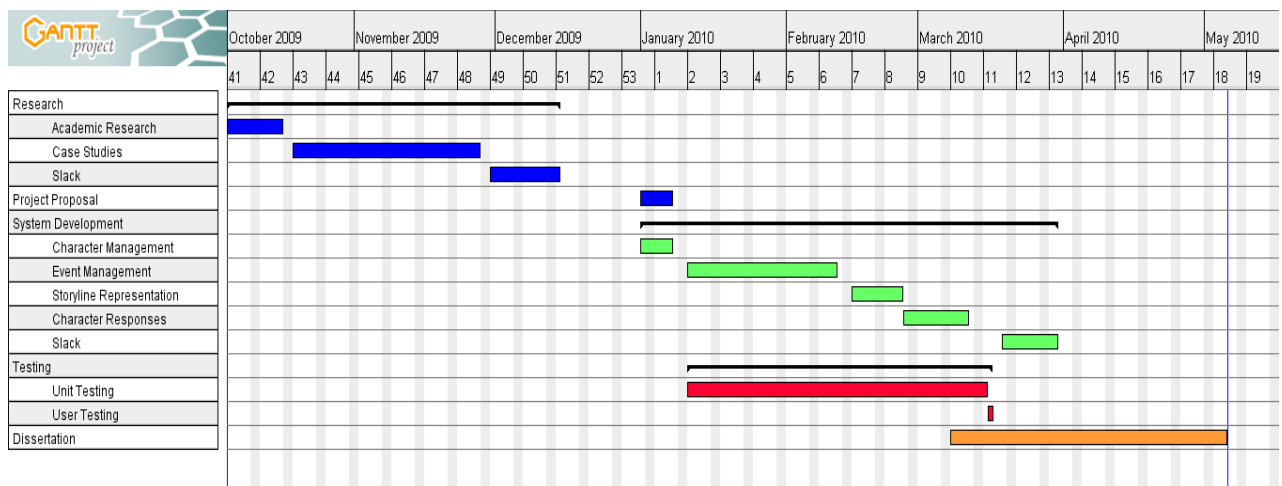


Figure 1: Project Timetable Gantt Chart

1.8: Summary

This chapter has introduced the motivation behind this project, and identified three main issues associated with developing ARGs:

- There is complexity associated with creating game content.
- A large workload is placed upon developers when in-game.
- Developers must constantly monitor incoming communications for puzzle solutions, and must reply to incoming character interactions.

The concept of a game engine was introduced, and its potential application with regards to solving the issues described above was discussed.

Chapter 2: Background Research

2.1: Introduction

This chapter introduces the fundamental concepts relating to this project. A formal definition of ARGs is provided. Following this, the current issues regarding ARGs are discussed, and a case is presented suggesting how these could be solved through the use of game engine technology. Finally, systems which are similar in nature to this project are introduced and reviewed.

2.2: Defining Alternate Reality Games

Salen et.al (2003) defines a game as “*a system in which players engage in artificial conflict, defined by rules that results in a quantifiable outcome*”.⁽¹⁾ This system is referred to as a magic circle – a well-defined region in which a game takes place.

ARGs place a large focus upon a narrative, which is broken down into plot segments. These are published to players via services associated with fictional characters, such as email or weblogs. The ‘hook’ to the game is provided by puzzles, such as riddles and treasure hunts, which are set by the characters. Solving the puzzles reveals new plot segments.⁽²⁾ Puzzles “*pace the player’s progress through the game*”⁽³⁾, and players gain a “*feeling of involvement with and agency in the game*”⁽³⁾ due to this.

Unlike a board game or a video game, which have a predefined means of interaction, ARGs permit the use of any interaction method permissible in real-life. Thus, it is not possible to establish a well-defined magic circle of belief as the boundary between the real world and the virtual world becomes blurred.⁽⁴⁾ Montola (2005) argues that a magic circle can be extended to include the scenario presented by ARGs by undergoing three forms of expansion:⁽⁵⁾

- Spatial expansion means that the actual location in which a game is played is either unknown or unlimited. The ARG ‘I Love Bees’ used telephone boxes across multiple cities to deliver plot content. The ARG ‘The Beast’ similarly used multiple websites to deliver its content. In these situations it becomes unclear where the games are located, as they can be played anywhere.
- Temporal expansion “*offers opportunities for interlacing games with everyday life*”.⁽⁵⁾ This behaviour is typical of ARGs, as the primary interaction method involves using tools players use on an everyday basis. There are no “*explicit game sessions*”⁽⁵⁾; instead interactions become part of daily activity.
- Social expansion blurs the boundary between a player and a non-player.⁽⁵⁾ If someone were to read an in-game character’s blog, they may be unaware that the content is fictional or part of a game. Additionally, someone may communicate with an in-game character but not participate in the game beyond these interactions. In either case, regardless of whether they intend to play the game or not, they have entered the alternate reality in some form, and social expansion has occurred.

Miligram et.al (1994) states that the composition of an environment can be described as a continuum, known as the Reality-Virtuality Continuum⁽⁶⁾, shown in Figure 2.



Figure 2: The Reality-Virtuality Continuum

At the left extreme is the real environment, where every object is physical and can be interacted with according to the laws of physics. At the right extreme is an entirely virtual environment, where every object has no physical representation but is instead represented digitally. The space between these two extremes is defined as mixed reality, where “*real world and virtual world objects are presented together*”. ⁽⁶⁾ This separates into two categories:

- Augmented Reality represents a real environment, to which virtual objects have been added. An example of augmented reality is a head mounted display that adds digital labels to objects the user is looking at.
- Augmented Virtuality represents a virtual environment, to which real objects have been added. An example of augmented virtuality is a user interacting with a 3D environment displayed via a monitor. The real-life interactions of the user are mirrored in the virtual environment by some interaction device, such as motion tracking via a Nintendo Wii remote.

It is not possible to place ARGs into a distinct category due to their varying nature; one game may focus more on an augmented reality, whereas another may focus more on augmented virtuality. However, some form of augmentation occurs, thus ARGs can be placed within the mixed reality range. Dawson (2008) shows that ARGs are formed by “*applying a narrative context*” ⁽⁷⁾ to a mixed reality.

2.3: Problems with Alternate Reality Gaming

A typical ARG will span multiple websites, multiple web services and may also include offline content. This situation raises some logistical issues that must be solved by the development team, which can be categorised as:

- Creating the game storyline.
- Detecting puzzle solutions.
- Handling character interactions.

2.3.1 Creating the Game Storyline

Szulborski (2005) states that the narrative and content of an ARG should be developed prior to the game starting in order to reduce the workload during game uptime ⁽³⁾. A typical ARG is played over a period of months and has hundreds of plot segments. If this content was to be developed simultaneously to the game being played, it could lead to team “*burnout*” ⁽⁸⁾, as experienced by the development team of ‘Perplex City’.

Creating a narrative up-front is a complicated procedure due to the vast amount of content to develop, and the need to represent the large narrative in a simplistic manner.

Traditionally the narrative has been broken down into arcs, which “*delineate specific phases in the story*” ⁽⁸⁾. This form of categorisation was used by the ‘Perplex City’, ‘The Lost

Experience' and 'The Beast' development teams as a way of simplifying the narrative structure. However, this approach does not go far enough to allow the developers to gain insight into how the narrative will be executed. Thus, each arc is also represented as a flow chart of distinct events, as shown in Figure 3. Entries in the flow chart represent plot segments, and the link between events either represents the passage of time or a puzzle. By tracking the player's progression through the narrative, developers can identify what should be published next; or identify if the game has completed.

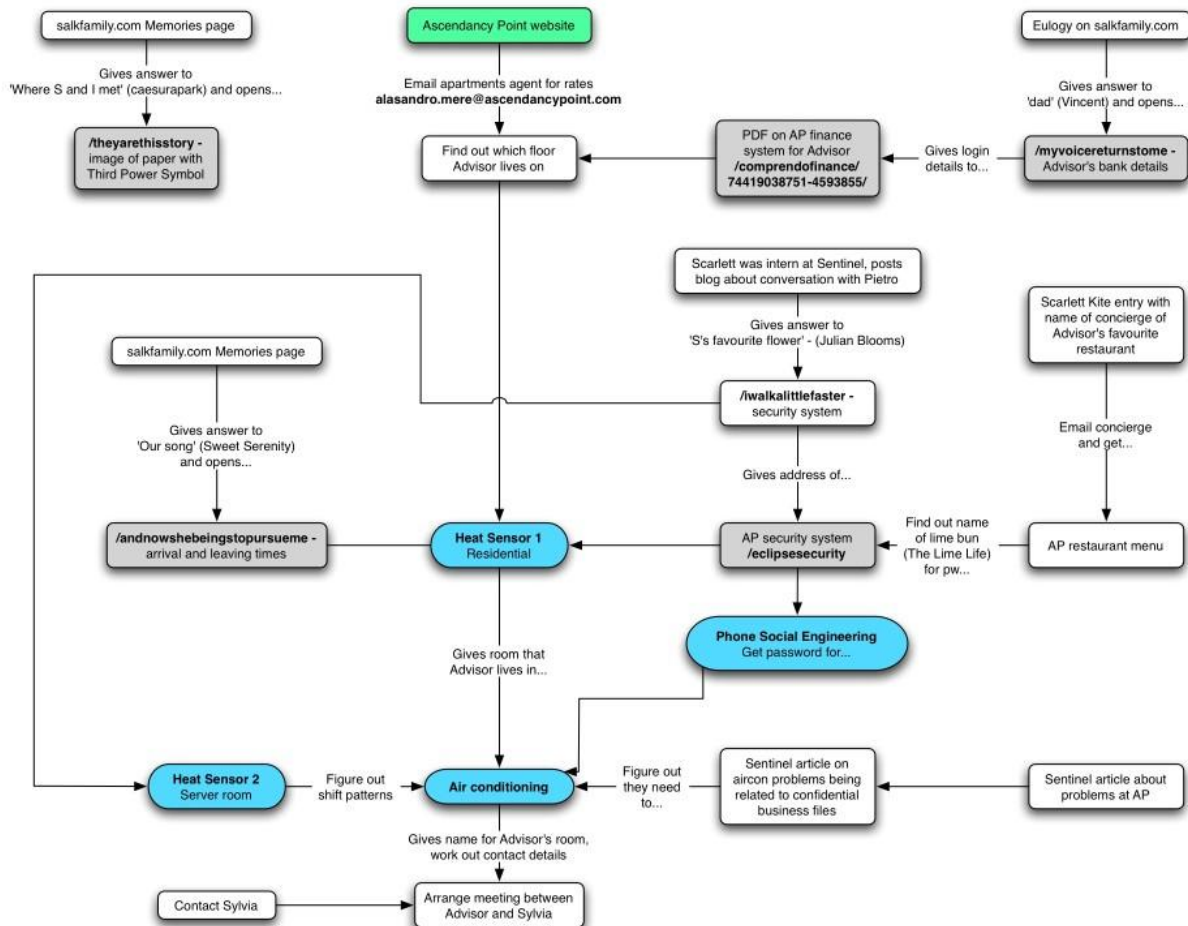


Figure 3: Perplex City Ascendancy Point Flowchart ⁽⁹⁾

Flow charts provide an easily understood method of representing how players will 'flow' through the narrative, and highlight the complexity of a narrative. Figure 3 represents an early arc in the 'Perplex City' narrative, which has multiple plot branches. Szulborski (2005) notes that narratives with multiple-branching plots are undesirable in ARGs as only one of these branches will be visited, meaning time has been wasted creating the alternate timelines ⁽³⁾. Later arcs in 'Perplex City' adopted linear structures due to this. Hon (2007) also noted that this made the game both "*easier to play*" ⁽⁸⁾ and narratives easier to manage.

There are two main issues regarding the use of flow charts to represent narratives. Flow charts can take a large amount of time to create, as they must be tracked manually.

2.3.2 Detecting Puzzle Solutions

Kim et.al (2008) suggests that in order for puzzles to act as a regulatory system for the game's narrative, "*the designers must react immediately*", and that "*the storyline must be updated in real time*".⁽²⁾ If these criteria are not met the player's sense of involvement is lost. The ARG 'Majestic' terminated early due to this loss of player involvement.⁽³⁾ The game had a 'window' in which all updates and character interactions took place. Outside of this window the game laid dormant, causing players to feel their level of involvement was not worthwhile.

'The Beast' and 'I Love Bees' employed a "*round-the-clock monitoring*" system⁽²⁾ to solve this problem. The developers continuously monitored character's inboxes, voicemails, and blog comments for solutions to puzzles. However, this technique quickly becomes *tedious*"⁽³⁾, presenting a major time management issue for large scale games, which can amass hundreds of thousands of forum posts.⁽²⁾⁽¹⁰⁾

It is possible to recruit team members who have the sole role of searching for puzzle solutions.⁽²⁾ This approach can alleviate the time commitments of the developers, so more time can be focused on creating new content and handling character interactions. However, this approach has the added cost of recruiting extra personnel to carry this out, which may not be feasible for low-budget ARGs.

Automation has been introduced in order to reduce the dimensionality of this problem. 'Perplex City' used Flash files on some websites to represent puzzles.⁽⁸⁾ Email notifications were sent to the developers whenever solutions were detected. Similarly, 'The Lost Experience' included a challenge requiring users to submit images, and new content would be released once a target upload count was reached.⁽¹¹⁾ Automation offers to lessen the burden placed upon developers during the game's uptime if used on a larger scale.

2.3.3 Handling Character Interactions

As with puzzle solving, handling character interactions requires a large proportion of the developer's time. When players communicate with the characters in an ARG they wish to receive personalised responses,⁽³⁾ adding to the level of immersion into the game world.

An issue arises when considering the potential scope of an ARG. 'I Love Bees' generated an average of 33,000 lines of daily chat⁽²⁾, drastically increasing the development team's workload.

Large-scale ARGs such as 'The Beast' hired actors to act as characters, who were responsible for dealing with all communications for a particular character. This reduces the responsibilities of the development team. Whilst the cost of hiring actors may not be feasible for lower-budget games, it is possible to compromise and assign each member of the development team particular characters to be responsible for.

'The Beast' developers also included automation to reduce the number of interactions requiring personalised, human-typed responses. This was achieved through a heavily scripted version of ELIZA⁽¹⁰⁾ – a program which uses natural language processing to return relevant responses to input. Pre-scripted responses were assigned to particular keywords; allowing for communications that are either mundane or relevant to the game. However, the main issue regarding automated interaction handling is that responses will become less personal. The developers of 'Perplex City' did not use any automated services to handle character interactions for this reason.⁽⁸⁾

2.4: Game Engines

A game engine is a set of reusable components which can aid game development teams realise games in a cost-effective, efficient manner. Typical game engines such as the ‘Unreal’ engine consist of multiple components, including graphics rendering, physics calculations and multiplayer networking.⁽¹²⁾

Game engines aid the development of games by providing a framework for commonly-used components. Development teams do not need to continuously build games from the ground-up, which is time-consuming and risks introducing errors into basic components unnecessarily. Game engines commonly feature components required for console-based game development. However, this is not necessary in order to classify a game engine as such. By introducing a set of reusable components relevant to developing ARGs, some of the current issues described above can be alleviated.

2.4.1 Creating the Game Storyline

A game engine component could automatically construct a flow chart based on the current timeline of events. This would save development time by removing the necessity to manually create a visual representation of the storyline, and could introduce large-scale time benefits if plot segments could be interactively rearranged.

The risk of developers not responding according to the player’s pace could be solved by automatically publishing new plot segments as needed. The engine would then know the current position in the narrative, which could be automatically mapped onto the relevant flow chart. This eliminates the need to manually track the progression of the storyline.

2.4.2 Detecting Puzzle Solutions

A component of a game engine could be responsible for automatically tracking for solutions to puzzles and acting appropriately when a solution has been detected. This would satisfy the concern raised by Kim et al (2008), that “*the storyline must be updated in real time*”.⁽²⁾

Multiple puzzle components could be combined in order to track the solutions of different puzzle types. For example, a keyword-scanner component could check for puzzles which require textual solutions. Alternatively, a hit counter could be placed onto a hidden web page to monitor the unique visitor count. When a certain threshold is reached, the page is logged as ‘found’.

2.4.3 Handling Character Interactions

Different approaches to character interactions could be combined through the use of multiple components in a game engine. For example, a keyword-scanner component could automatically scan incoming communications, adding a pre-scripted response to a reply. An ELIZA-based component could be combined with the above to add personalisation to a message. A queue-based component provides developers with the opportunity to add their own material to replies, or to moderate and edit the content generated by the other components. This system would allow developers to ‘pick-and-choose’ the style of interaction handlers required for their game, whilst reducing the time spent manually handling interactions.

2.5: Similar Systems

No system exists which could be defined as a true game engine for ARGs. However, there are similar systems in academia and in industry which incorporate features that are desirable to an ARG engine.

2.5.1 The ARGOSI Framework

The ARGOSI framework was developed for creating and managing educational-based ARGs⁽¹³⁾. This is a web-based system which acts as a centralised location for players to view and solve challenges. Whilst it is not strictly a game engine, it does provide components which could act as part of a game engine. The framework provides a puzzle management system which allows puzzles to be created by developers, and solved by players. Puzzles can be organised into challenge sets, a concept similar to arcs.

The limitation of the ARGOSI Framework is that players know from the onset that they are playing an ARG, by being required to sign up to the framework website. Puzzles are solved by entering the solution directly into the framework⁽¹³⁾, which risks losing the player's immersion into the alternate reality. ARGs tend to be community-based games; but the ARGOSI Framework is designed to act as a unique game for each user. Therefore, this system is not scalable to the more popular, community-based ARGs.

The ARGOSI Framework records the puzzles that players have solved. The only immediacy of response that is presented by the framework is a live leaderboard update, and that the next puzzles will become 'active'⁽¹³⁾.

2.5.2 Xenophile Reactor

The Xenophile Reactor system, developed by Xenophile Media in 2007,⁽¹⁴⁾ acts as a web-based content management system, which tracks players progress through the game narrative. Content is published via various platforms and web services. The system also includes automated character interaction handlers, and additionally the automatic release of scheduled content.⁽¹⁴⁾

This system is closer to a game engine than the ARGOSI Framework, and is a more well-rounded approach to creating and managing an ARG, as the components incorporated in the system allow for a more realistic ARG experience. Puzzle solutions can be detected with regards to their original context, rather than having to be directly entered into the system. The use of scheduling plot segments incorporates the real-time aspect of ARGs. Time considerations of the development team are accounted for by including automated character interaction handling.

However, Xenophile Reactor suffers the same issue as the ARGOSI Framework. The system requires players to sign up, and puzzles are tracked on a player-by-player basis. This removes the possibility for community collaborations such as 'The Cloudmakers' to occur; something which large-scale ARGs such as 'The Beast' relied upon⁽¹⁵⁾. Thus, the system is not scalable in its present form to become a general purpose engine.

2.6: Summary

In this chapter the concept of Alternate Reality Gaming was defined. The main issues described in Chapter One were discussed in detail, using three case studies:

Alternate Reality Game Engine

- The Beast.
- The Lost Experience
- Perplex City

The concept of a game engine was discussed in detail, with its relevance to Alternate Reality Gaming introduced. The benefits of introducing a game engine for ARG development were discussed in detail. In summary:

- An ARG engine can automate frequently-used activities such as storyline management and puzzle solution detection.
- Game engines can be extended with additional components, thus are scalable to cope with the changing demand of ARGs.
- Developer workload is reduced, allowing more time for game content development.

Two similar systems, the ARGOSI Framework and Xenophile Reactor, were introduced. Their relevance to the project and their similarities to the concept of an ARG Engine were discussed. From this review it was established that current attempts at creating a game engine have not placed a large focus upon community level multi-player games – the style of play normally adopted by ARGs.

Chapter 3: System Design and Implementation

3.1: Introduction

This chapter provides a detailed description of the design process undertaken in order to develop a system. Based upon the requirements specified in Chapter One, the proposed design will identify which of these have been met. The design decisions taken will be justified in context of their advantages and disadvantages. The implementation of the system is also discussed; describing the approach taken for development, the tools and technologies used, and how the system components are translated from design to implementation.

3.2: System Architecture

Figure 4 represents a high-level view of the system architecture, identifying four major aspects, each of which will be discussed in this chapter:

- A means of data storage, and communication with this storage.
- A user interface.
- A means of delivering game content and receiving updates to the game state via web services.
- A game controller to manage a game once it is started.

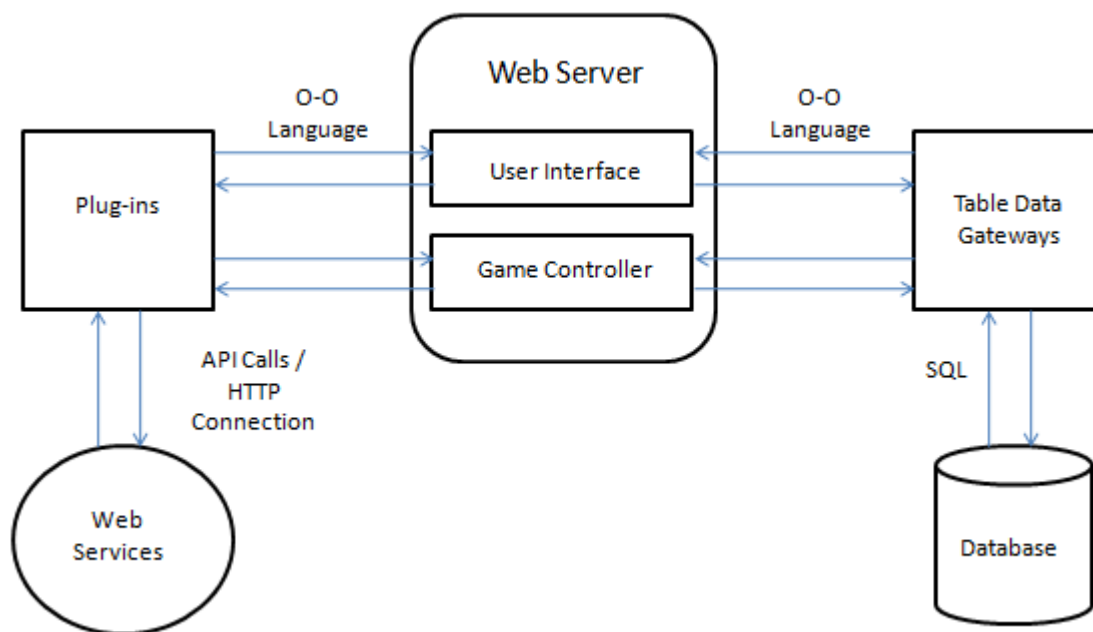


Figure 4: System Architecture Design

3.3: Software Development Model

The development of the system consisted of five phases:

- System design.
- Plug-in development.

- Character management development.
- Event scheduling development.
- Game management development.

This correlated to the waterfall development model, but this was not used for development as it assumes each phase is completed perfectly before moving onwards, leaving no need to return to a phase. ⁽¹⁶⁾ Whilst it is possible to perform unit testing and integration testing upon the completion of a phase, this gives no guarantee the phase will never be returned to due to later developments.

Instead, the development of the system complied with the modified waterfall model, in which earlier phases can be returned to in order to solve issues related to the design, the requirements, or errors discovered through testing ⁽¹⁶⁾.

3.4: Tools and Technologies Used

3.4.1 PHP

PHP is an “*HTML-embedded scripting language*”⁽¹⁷⁾; executed on a web-server in response to a client request. Output is returned to the client.

PHP provides a library of commonly-used functions without requiring any configuration. A large focus is placed upon Internet-based functionality, with built-in support for sending emails, communicating with a MySQL database. Support is provided for creating and parsing XML files using the `DOMDocument` and `SimpleXML` extensions. The PHP library is capable of being extended by including frameworks.

PHP was chosen to implement the system design for the following reasons:

- PHP is well-placed in order to supply the system to multiple developers due to its server-side execution.
- PHP supports processes to be forked onto a server for a near-continuous uptime. This satisfies the need for the system to continuously monitor the game state without requiring a constant connection from a user.
- Integration with MySQL provides an efficient method of communicating with a database management system.
- PHP’s library reduces development times by encouraging code re-use.
- PHP’s object-oriented nature simplifies code due to abstraction and encapsulation. Functionality can be defined in interfaces and implemented by multiple sources, which is useful to establish a plug-in architecture.
- Exception handling provides a mechanism for the interruption of program execution should an error be detected, and to take corrective action.

3.4.2 Zend Framework

The Zend Framework extends the PHP library to provide a wider, more advanced set of components. Zend is an object-oriented framework with little dependencies between the various components, thus classes in the framework can be included using a “*use at will*”⁽¹⁸⁾ approach. Developers are not constrained to developing in conformance to a particular design pattern, enabling the framework to be scalable. The framework is robust, and has been strictly unit tested, so developers can use it with confidence ⁽¹⁸⁾.

The Zend Framework provides some components which are especially useful with regards to this system:

- Communication with web services is provided by multiple packages. Web services operated by Google, such as Blogger, are included in `Zend_Gdata`. Other web services including Twitter are provided in `Zend_Service`.
- `Zend_Mail` extends the PHP library to provide efficient email sending via multiple transports and enables emails to be read via POP3 and IMAP.
- `Zend_Config` provides functionality for storing, reading and writing INI configuration files, which can be used to manage the game state.
- `Zend_Log` enables system logging and supports multiple error levels to increase the clarity of logs, which is beneficial for unit and user testing.
- `Zend_Date` provides a simplified means to convert dates and times into one common representation.

3.4.3 Smarty

Smarty is a template engine for PHP, designed to separate application logic from presentation logic through the use of template files.⁽¹⁹⁾ These files contain presentation logic, which will include mark-up such as HTML. The main advantage with using Smarty is that the two concerns of application and presentation are separated, facilitating independent development with a low risk of introducing errors in other layers. Separating the layers additionally increases code readability - PHP files contain only application logic, and TPL files contain only presentation logic. Smarty template files are compiled and optimized such that using the Smarty template engine introduces no significant performance penalties.⁽¹⁹⁾

3.4.4 MYSQL

MYSQL is a relational database management system, which was chosen for implementation of the database backend of the system. MYSQL is built upon the relational model, in which entries in tables are related through unique identifiers. It supports foreign key mapping; the concept that entries in tables can relate to entries of other tables by storing unique identifiers. This is advantageous over storing redundant data as referential integrity can be ensured. All data relating to an entry can be deleted or updated accordingly with changes to that entry. The database stores the user-created game, including event content, puzzle solutions and organisational data.

3.4.5 XAMPP

A common deployment pattern for a general-purpose web server is to use a LAMP stack⁽²⁰⁾. This involves using a Linux system, an Apache HTTP server, a MYSQL database and PHP. XAMPP provides an implementation of a LAMP stack, which is bundled as a single release to permit fast deployment with minimal need for configuration⁽²⁰⁾. XAMPP provides out-of-the-box support for all tools mentioned above with the exception of Smarty, which can be easily included without affecting the XAMPP configurations. In addition to the components mentioned above, XAMPP provides support for Perl and FileZilla FTP, which are not used for the implementation of this project, but may prove beneficial for later extensions to the system. For example, FileZilla FTP could be used to deploy updates to character web pages.

A web server deployment is advantageous for the development of this system as they do not require users to establish a permanent connection to the system. Web servers have high up-time rates, which, when combined with PHP's ability to be scheduled as described in Section 3.4.1, provides a suitable environment with which to satisfy the system requirements.

3.5: Database Communication

Communication with the database is overseen by the Table Data Gateway pattern. Interaction with each table is handled by a class which encapsulates all necessary SQL commands; providing a high-level interface for the system. This also minimises the chances of errors regarding syntactically-invalid database commands ⁽²¹⁾.

Table Data Gateways do not provide error correction functionality. Instead, upon the detection of an error, an exception is returned to the caller. This method permits error correction to be performed using multiple tables. A `NoDataFoundException` is thrown when a query returns no data. File uploading failure is denoted as an `UploadFailedException`. Invalid data is represented as a `ValidationException`. A `DatabaseConnectionException` is thrown if a connection to the database fails.

Figure 5 shows this pattern being used to obtain a list of all the stored characters.

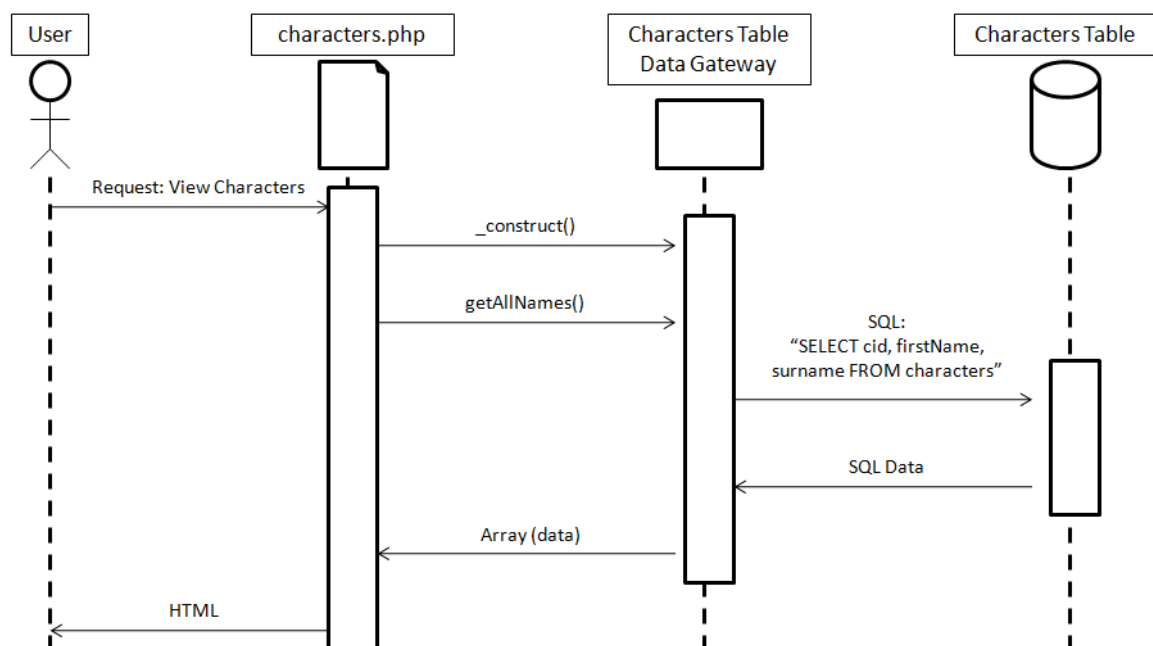


Figure 5: Table Data Gateway Pattern – Sequence Diagram

3.6: User Interface

3.6.1 Layout

A consistent layout is presented for every web page in order to introduce familiarity when interacting with the system. This layout, shown in Figure 6, is split into two sections. The header displays a horizontal navigation menu of hyperlinks to the main system sections. Horizontal menus show a navigation menu without the need for scrolling. This also allocates a larger horizontal space for the main content to occupy. Whitespace can be used to increase the readability by acting as a mechanism to structure, group and separate items.

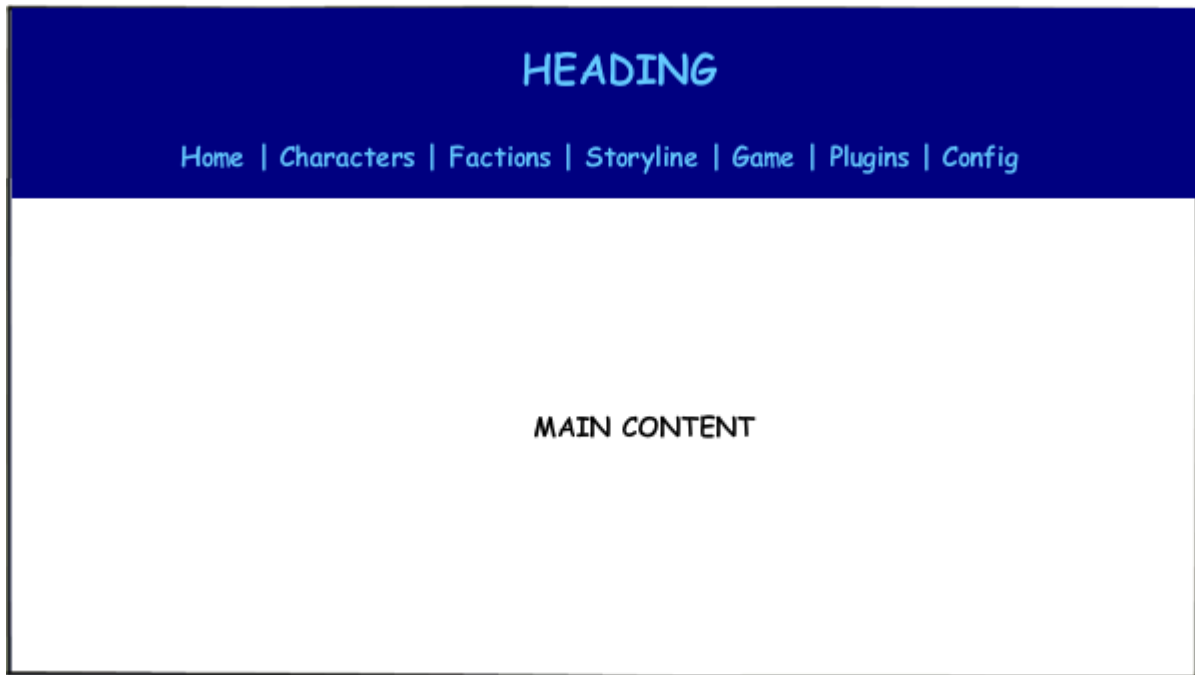


Figure 6: User Interface - Layout

The navigation menu does not expose the full functionality of the system, which can potentially cause confusion when a user searches for a specific action. In order to alleviate this, each link name categorises the actions performed by that section. For example, the ‘Characters’ link indicates that the functionality provided in this section acts upon characters. The presence of the navigation menu on every page provides a “*go back to a safe place*”⁽¹⁶⁾ method, aiding users in understanding the system structure.

The header background contrasts with the main content background to visually distinguish the two sections. The header background is complemented by the lighter-shaded links to demonstrate a grouping whilst maintaining visibility. Links change colour upon mouse rollover to a contrasting pink, to show a user’s position within the link list.

3.6.2 Item Selection

A consistent interface is provided for item selection. Each item is allocated a box, in which an icon and a textual identifier are presented, as shown in Figure 7. The image associated with an item acts as a visual cue to the user⁽¹⁶⁾, providing a recognisable identifier to aid users when selecting from large lists. This interface is used to represent characters, factions and arcs. The interaction rules with this interface can therefore be generalised and applied in all three situations due to familiarity; contributing to an increased “*learnability*”⁽¹⁶⁾ of the system.

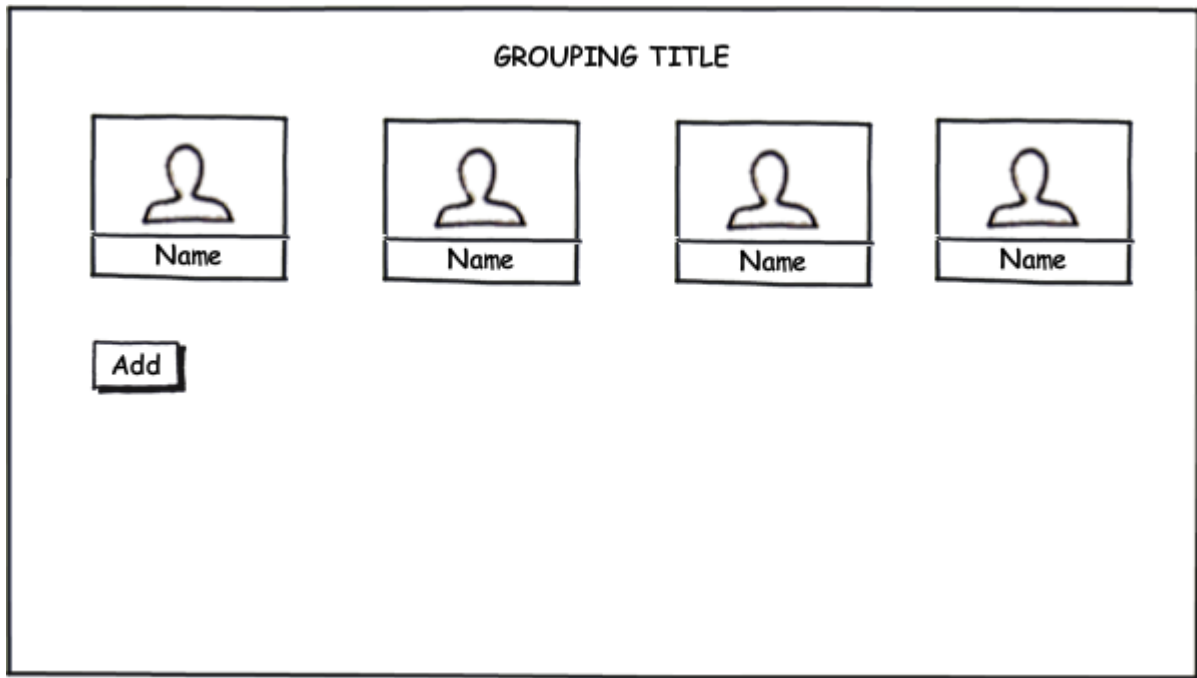


Figure 7: User Interface - Item Groupings


3.6.3 Character Description Page

This page, shown by Figure 8, displays a character's profile. A character's image and description are displayed, as well as a tabulated representation of all linked web service accounts for a character. Each entry displays an icon representing the type of service, and a web button provides functionality to delete a linked account.

The purpose of this page is to present an encapsulated view of the full extent of a character's involvement in the game. Actions relating to a character are located here to provide a logical grouping.

Character deletion is the most unlikely function to be performed, thus this action is placed at the bottom of the page, minimising the chance of accidental activation. For this reason, this style is also adopted for the deletion of arcs and events.

Editing Character: CHARACTER NAME



Description

Character Description

Services

Type	Description	Functions
BLOGGER	My Weblog	DELETE LINK

Link A Web Service

Delete Character

Figure 8: User Interface - Character Description Page

3.6.4 Arc Description Page

This page displays a flow chart of all scheduled events for the selected arc, shown in Figure 9. A flow chart representation of events directly relates to the sequential timeline formed when scheduling events, and correlates to the typical pattern of storyline management used by ARG developers. The use of this representation satisfies Requirement 6.

The title of each entry represents the linked account used for publishing. This is clarified using an icon of its type of service, consistent with how web service accounts are represented on the character and event description pages. The summary of an event is displayed in the main section of the entry to contextualise the event and its position within the flow chart.

The colour of an entry represents its state when a game is in progress. A white background indicates the event has not been posted, thus is inactive. A green background indicates an event is completed. A yellow background indicates the current event. In addition, the state is displayed textually to compensate for users who are unable to fully perceive colour. A legend representing the colours and their meanings is displayed in order to reduce confusion with the colouring system. This, when coupled with the current game state, satisfies Requirement 7.

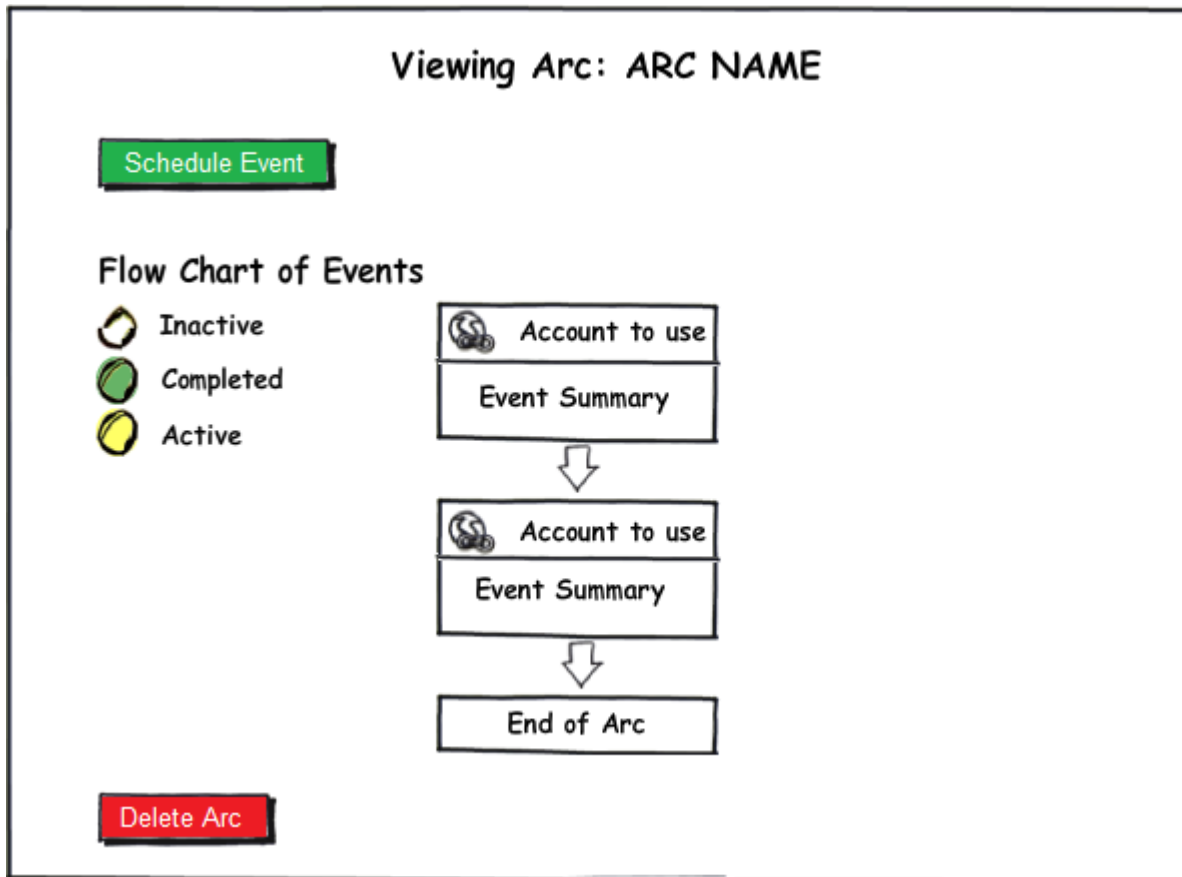


Figure 9: User Interface - Arc Description Page

3.6.5 Events Description Page

Similar to the Character Description page, this page encapsulates all information relating to a selected event. Figure 10 describes the layout of the Events Description page. The event is contextualised by displaying the related arc, character, puzzle and linked web service account; each of which has an associated icon in order to provide a more recognisable link. The event status is also displayed when a game is in progress, according to the standards established on the Arc Description Page.

This page has two sections populated by plug-ins. The 'Event Content' section describes what is to be posted in relation to this event. The 'Puzzle Information' section describes the solution to the associated puzzle. These are displayed as lists, in order to provide a structured, but variable, means of displaying such information.

Event Details For: EVENT NAME

Taking Place In:
ARC NAME

Posted Using:

Account to use

Posted By:

Character

Puzzle Type:

Puzzle Type

Event Content

Title: TITLE

Subject: SUBJECT

Content: CONTENT

Puzzle Information

SOLUTION 1

SOLUTION 2

SOLUTION 3

Delete Event

Figure 10: User Interface - Events Description Page

3.6.6 Tables

Tabulated data is displayed such that the background colour of a row alternates between contrasting colours, as shown in Figure 11. Row clarity is increased by using colour as a grouping mechanism, distinct to the rows above and below.

Heading	Heading	Heading
Entry	Entry	Delete
Entry	Entry	Delete

Figure 11: User Interface - Tables

3.6.7 Forms

Web forms are the primary method of input to the system. A common, consistent form layout is adopted throughout the system, shown in Figure 12 to improve the learnability of the system. Forms are divided into groups to conceptually relate form entries. Each grouping is identified by a legend, which acts as the grouping title.

Each form entry has its own label to provide contextual information. Labels and form entries are vertically aligned to separate the description from user input. Each label and form entry pair is related through horizontal alignment.

Legend

Form field ComboBox ▼

Form field ☐ Value 1 ☒ Value 2

Form field

Some text
A second line of text

Submit

Figure 12: User Interface - Forms

3.6.8 Notifications

Notifications identify when a user has successfully completed a task, introducing “closure”⁽¹⁶⁾, or when errors requiring the user’s action occur. Notification types are distinguished visually using colour; errors are presented in a red notification, and success statuses are presented in a green notification, demonstrated in Figure 13. The information within the notification delineates the reason for the notification, including any actions to take. Enforcing this internal consistency establishes a common meaning within the system regardless of the cultural implications of the colours chosen.⁽¹⁶⁾

An error has occurred.
Error details

Success!
Details

Figure 13: User Interface - Notifications

3.6.9 Buttons

Web buttons represent a link to begin a procedure of actions within the system. Buttons provide a larger clickable area than standard text links, increasing visibility of functions.⁽¹⁶⁾

The background colour of a button distinguishes the action it performs, similarly to notifications. Green buttons represent creative actions, such as adding a new entry. Red buttons represent deletion actions, as shown in Figure 14.



Figure 14: User Interface - Buttons

3.7: Character Management

Characters are stored in the `characters` table, shown in Figure 15. Users can store character information and an image to represent a character's profile, providing a solution to Requirement 1.

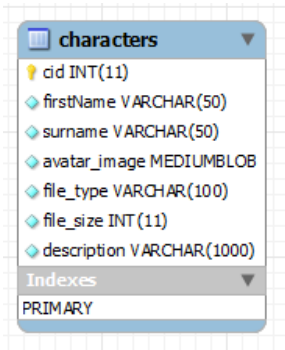


Figure 15: The 'characters' Table

Requirement 2 is satisfied by allowing characters to be grouped into factions. Factions are a scalable method of organising large numbers of characters, and are stored in the `factions` table. The `factionmembers` table references the `factions` and `characters` tables to represent faction members. This method also allows characters to join multiple factions. The relational view of these tables is described in Figure 16.

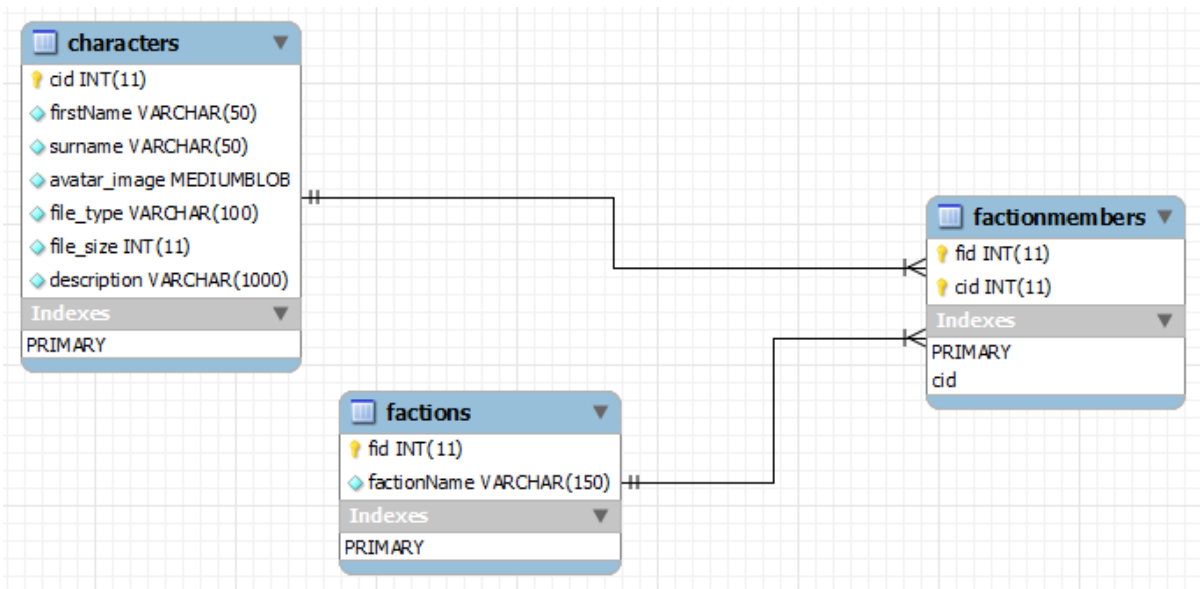


Figure 16: Factions Entity-Relationship Diagram

3.8: Plug-ins

A plug-in architecture is used to provide the system with a means of sending, receiving and processing game-related data. There are two types of plug-in used by the system, to satisfy Requirements 8 and 9, which both share a common architecture:

- Web service plug-ins.
- Puzzle plug-ins.

The plug-in architecture is based upon the Separated Interface and Plug-in patterns⁽²¹⁾. The Separated Interface pattern observes that interfaces form no dependencies to their implementations, thus implementations can exist within standalone packages. Interfaces are therefore used to define plug-in behaviour, such that plug-ins can provide their own implementation.

The Plug-in pattern specifies that these implementations are dynamically linked at run-time only when required. This is achieved by a factory class, which instantiates a plug-in and subsequently acts as a mediator by mapping function calls to the plug-in. A consistent interface is achieved by ensuring the factory implements the same interface as the plug-in classes it links with.

This forms the general dependency structure shown in Figure 17.

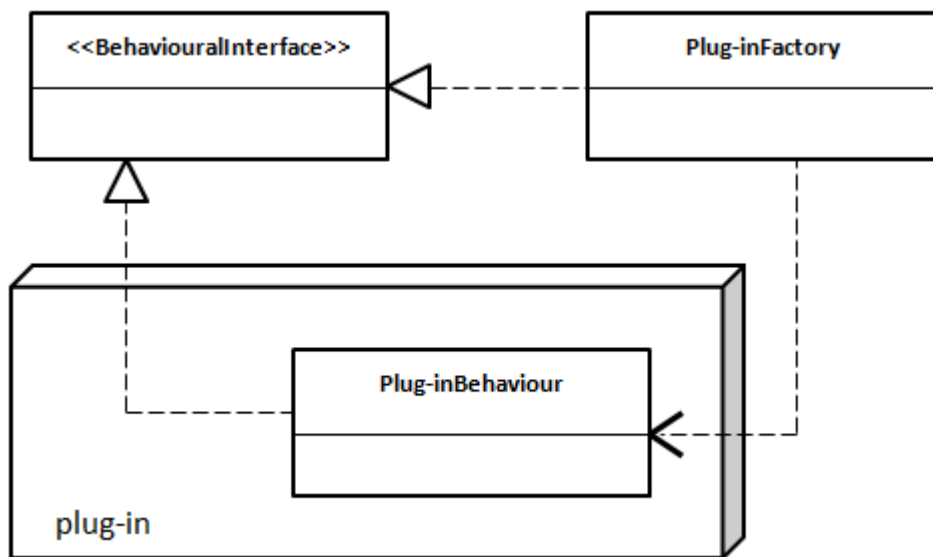


Figure 17: Plug-in Architecture UML Diagram

Factory classes establish a link to a required plug-in dynamically. This is achieved using Reflection, provided as part of the PHP library; allowing for a program to be modified at run-time. Fowler (2003) identifies Reflection as an effective implementation of the Plug-in pattern⁽²¹⁾.

Plug-ins are contained in a folder, identified by the plug-in name. Similarly, every class is prefixed by the identifier. Given the plug-in identifier (supplied by the system to the factory), and the behaviour defined by the factory, the class name can be constructed by String concatenation, and then instantiated. Upon instantiation, the system interacts with the factory, which mirrors this interaction to the instantiated plug-in, and returns any resulting data. This interaction is demonstrated by Figure 18.

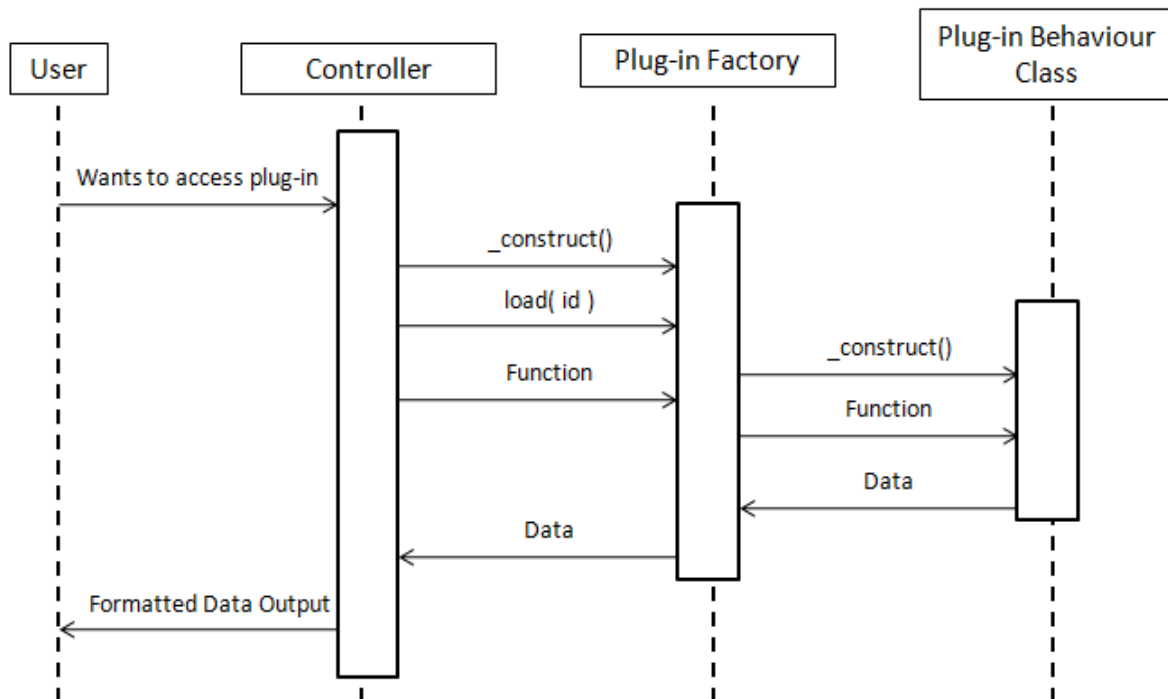


Figure 18: Plug-in Interaction Sequence Diagram

Error handling is not provided by a factory. Instead, errors will be returned to the caller as exceptions. If a plug-in cannot be linked, a `ServiceNotFoundException` is thrown. If a factory function is called prior to a plug-in link being established, a `NoServiceException` is thrown.

Three plug-ins were implemented to demonstrate the plug-in architecture of the system; consisting of two web service plug-ins, and one puzzle plug-in, to reflect the most commonly-used components in ARGs:

- A plug-in for Google's Blogger service.
- A plug-in for sending and receiving emails.
- A plug-in for scanning data for a set of keywords.

The particular implementation details of each plug-in will be discussed in later sections, where relevant.

3.8.1 Web Services

Web service plug-ins are used for communication between players and characters. Via this medium, characters post storyline content and set puzzles. Player responses represent solutions to these puzzles.

To achieve this behaviour, a web service plug-in must provide two categories of functionality:

- Linking a web service account to a character.
- Communicating with a web service.

Consequently, two interfaces define this behaviour and two factories provide access to any implementations. Both factories implement a common interface as a means of introducing consistency when handling with web services, shown in Figure 19:

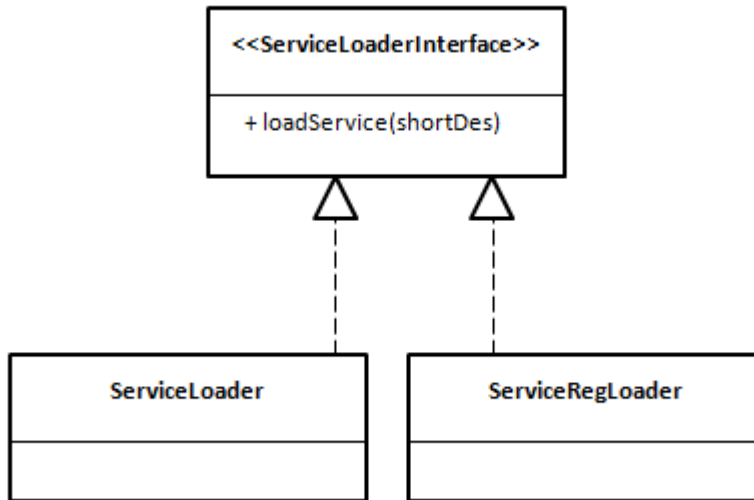


Figure 19: Web Service Factory UML Diagram

These plug-ins are tracked in the `servicelist` table, demonstrated by Figure 20. A plug-in's unique name, enforced by `ServicelistGateway`, is stored so the system can link a plug-in at run-time. A unique identifier is also used so plug-ins can be linked to events using foreign key mapping.

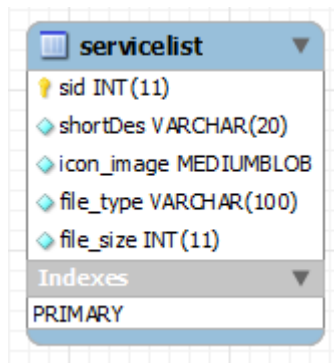


Figure 20: 'servicelist' Table

3.8.1.1 Linking

The act of linking a web service account to a character identifies the possible channels of communication a game can utilise, relating to Requirement 3. The system does not facilitate creating web service accounts, as APIs do not provide such functionality. This is due to necessary deployment of security, which is provided by a web service's own registration procedures.

Registration defines the act of linking an account, and `ServiceRegLoader` acts as a factory, linking to a class in a plug-in packaged suffixed by 'Register', as shown in Figure 21.

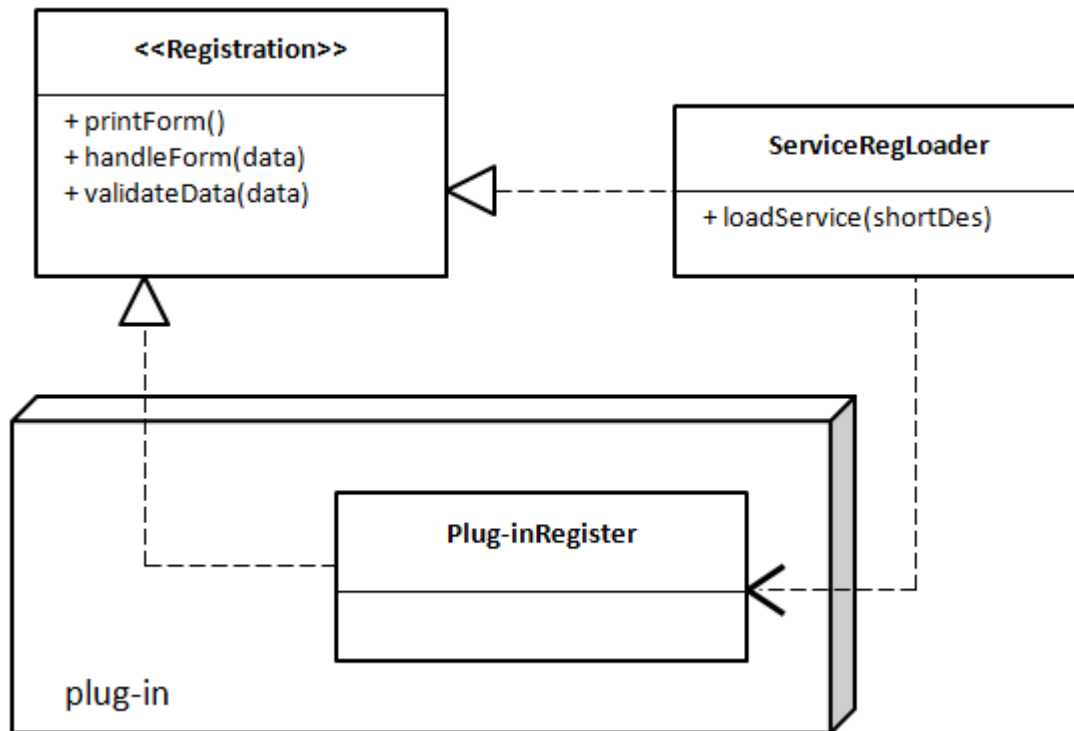


Figure 21: Web Service Registration UML Diagram

Additionally, each web service is allocated its own table within the database in order to store any configuration information required to establish a connection. The email table in Figure 22 demonstrates an application of this.

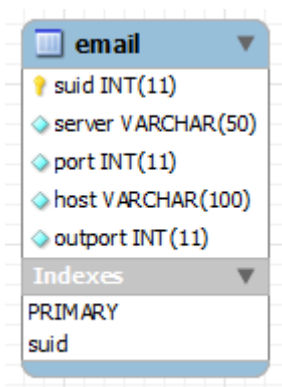


Figure 22 'email' Table

Section 3.9.1 describes the use of this behaviour, discussing the design decisions regarding this plug-in.

3.8.1.2 Communicating

Communicating with a web service provides a basis in which Requirements 10 and 11 can be satisfied. Plot segments are represented as events, containing content and an associated puzzle. Content is published via a linked account to represent the storyline. Subsequent replies represent proposed solutions to the puzzle.

`Client` defines this behaviour, and `ServiceLoader` acts as a factory, linking to a class in a plug-in packaged suffixed by 'Client', as shown in Figure 23.

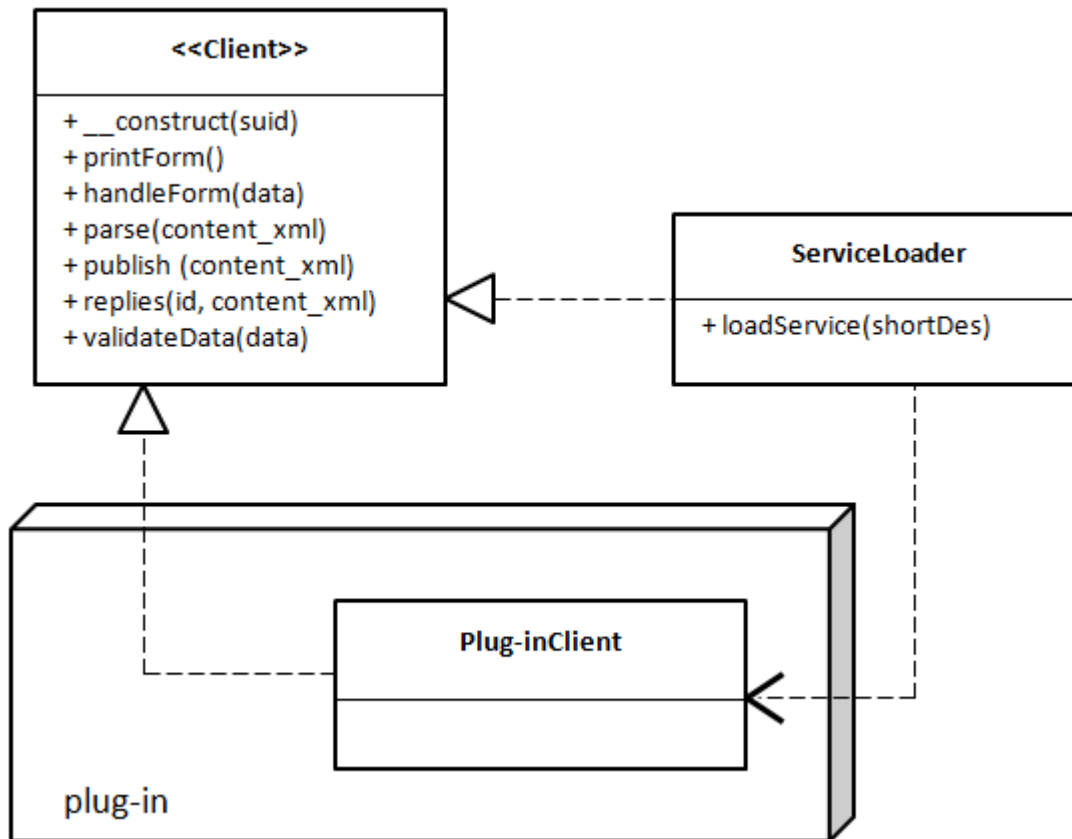


Figure 23: Web Service Communication UML Diagram

This behaviour is used for event scheduling and for posting and obtaining replies to events. Thus, the design of this plug-in type will be discussed in Sections 3.9.2 and 3.10: where appropriate.

3.8.2 Puzzles

Puzzle plug-ins represent challenges that players must solve. They define how a puzzle is solved, and how to check for such solutions. The `puzzlelist` table, demonstrated in Figure 24, tracks puzzle plug-ins in a similar manner to the `servicelist` table.

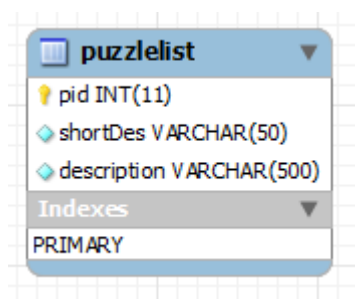


Figure 24: 'puzzlelist' Table

Puzzle plug-in behaviour is defined by `Puzzle`. `PuzzleLoader` links to a class in the plug-in package suffixed by 'Client', as demonstrated by Figure 25.

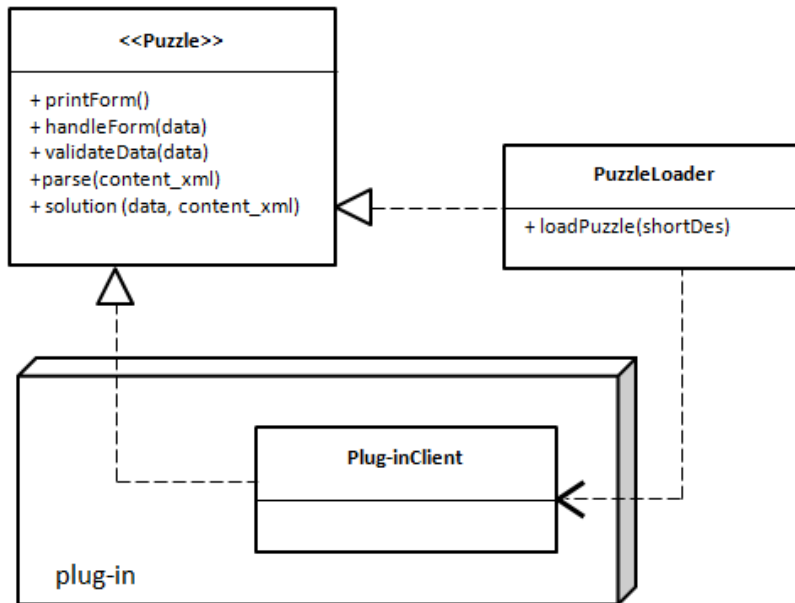


Figure 25: Puzzle Plug-in UML Diagram

Puzzle plug-ins are used during event scheduling to specify the solution to a puzzle associated with an event. This solution is then used to check against incoming replies. As such, the design of this plug-in will be discussed in Sections 3.9.2 and 3.10.2 respectively.

3.9: Game Creation

3.9.1 Linking Accounts and Characters

The linking procedure consists of three steps:

- Select a character.

The ‘Character Description’ page shown in Figure 8 displays all linked accounts for a given character; providing a natural entry point to this procedure and implicitly satisfying this step.

- Select a web service type.

The `shortDes` field for every entry in the `servicelist` table is displayed as a drop-down menu.

- Supply authorisation criteria.

This form includes two entries common to all web services – a username and password. The relevant plug-in is linked by passing its unique name to `ServiceRegLoader`. The `printForm` function defined in `Register` is used to populate this form with any additional configuration requirements a plug-in requires.

Upon submission, the `handleForm` function is used to store any configuration details in the plug-in’s assigned table. The `validateData` function is used to ensure the validity of supplied user data, by attempting to establish a connection to the web service.

The `users` table stores the username and password. The `servicesused` table represents linked accounts, referencing the `users` table for authentication, the `characters` table to identify the

owner of an account, and the `servicelist` table to identify a plug-in to handle interactions. The relational view of this arrangement is described in Figure 26.

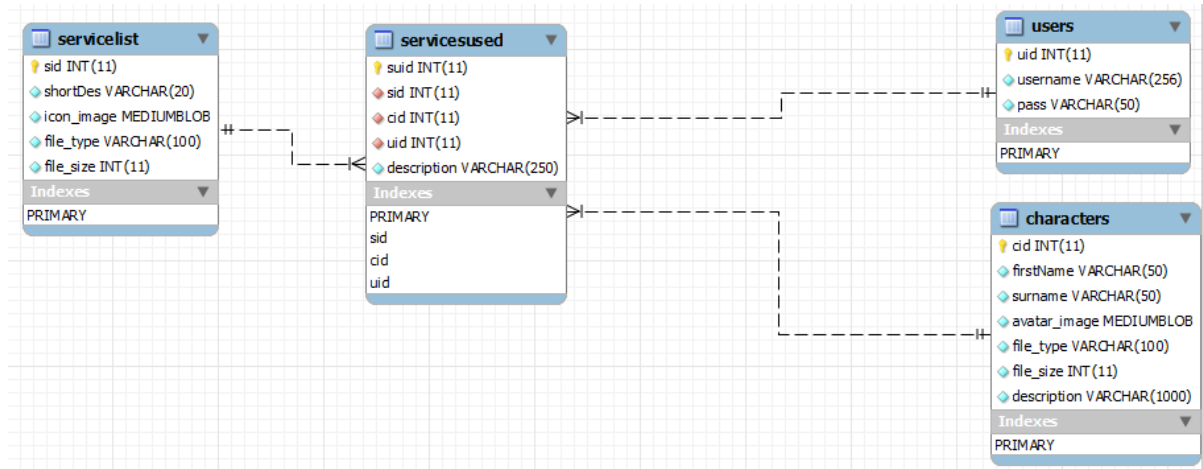


Figure 26: Linking an Account: Enhanced Entity-Relationship Diagram

This system implements this feature as described by Figure 27:

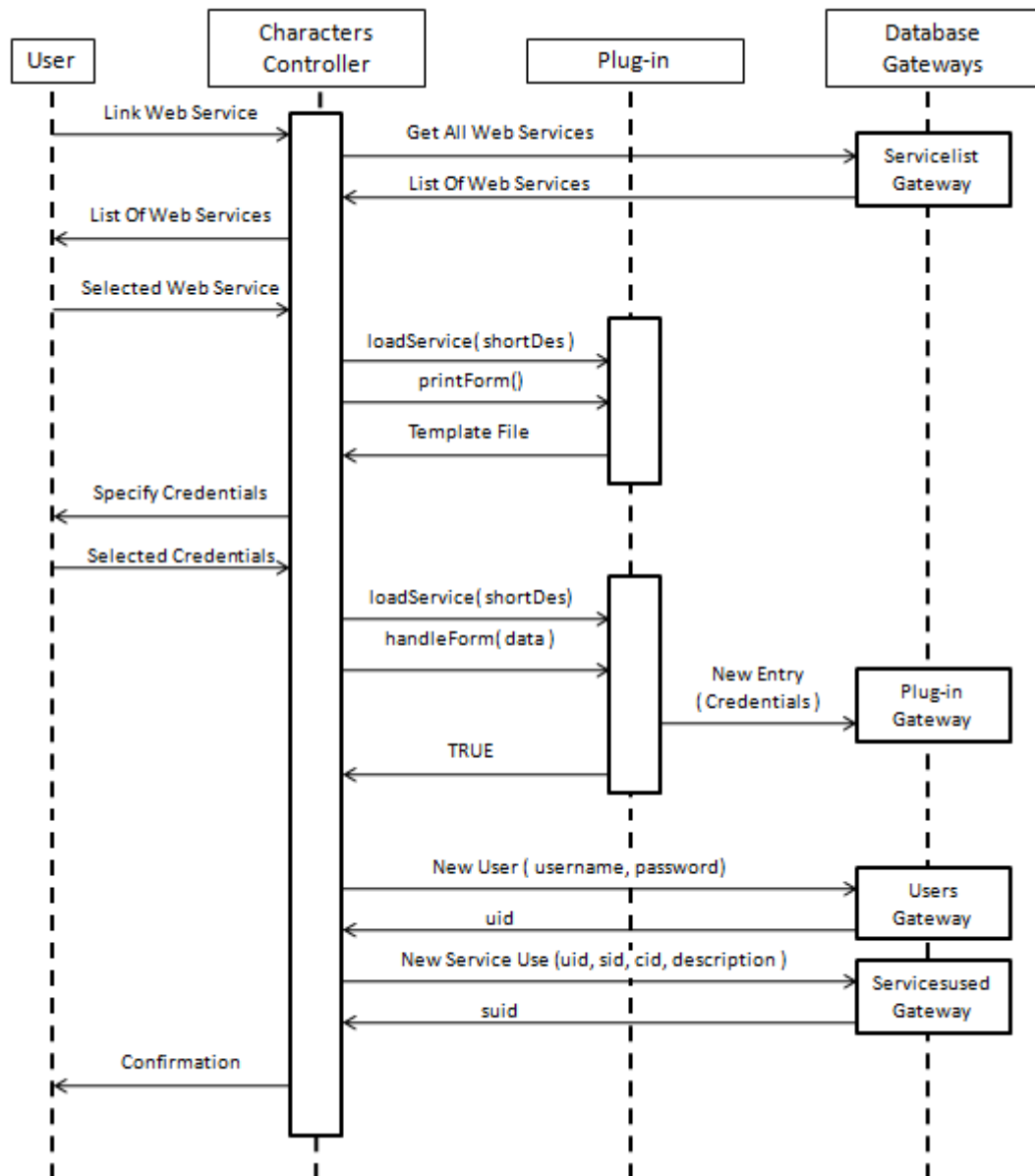
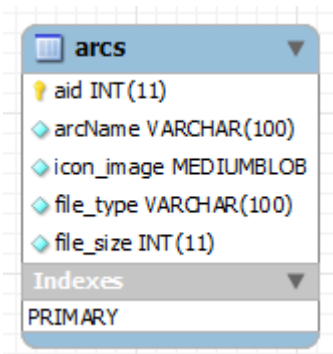


Figure 27: Linking a Web Service Sequence Diagram

3.9.2 Event Scheduling

Events are grouped into arcs to satisfy Requirement 4. Arcs provide developers with a scalable means of organising large numbers of events, and are stored within the `arcs` table shown in Figure 28.



The screenshot shows a database table named 'arcs'. It has five columns: 'aid' (INT(11)), 'arcName' (VARCHAR(100)), 'icon_image' (MEDIUMBLOB), 'file_type' (VARCHAR(100)), and 'file_size' (INT(11)). Below the columns, there is an 'Indexes' section showing a 'PRIMARY' index on the 'aid' column.

Column	Type
aid	INT(11)
arcName	VARCHAR(100)
icon_image	MEDIUMBLOB
file_type	VARCHAR(100)
file_size	INT(11)

Index
PRIMARY

Figure 28: The 'arcs' Table

The event scheduling procedure consists of four steps:

- Select an arc.

The 'Arc Description' page displays all scheduled events for a given arc; providing a natural entry point to this procedure whilst implicitly satisfying this step and introducing consistency with the linking procedure, demonstrated in Section 3.9.1.

- Select a character.

All names in the `characters` table are displayed to represent that an event is published from a character's perspective. This step satisfies Requirement 3.

- Specify event type.

This form is split into two sections, each consisting of one entry. The first section displays a list of all linked accounts for the selected character, obtained by filtering the entries of the `servicesused` table using the specified character identifier. The second section displays a list of all puzzle types stored in the `puzzlelist` table. These are placed within the same step to conceptualise the event structure as a whole unit.

- Specify event details.

The form structure of this step mirrors step three. Based upon the selected account, the `ServiceLoader` class loads the 'Client' class of the relevant plug-in. The `printForm` function, defined in `Client` is used to populate this form with the posting requirements a plug-in requires. The second section uses the `printForm` function defined in `Puzzle` in the same manner; displaying a form representing the criteria to solve the associated puzzle.

Upon submission the user data is passed to the `handleForm` functions defined by `Client` and `Puzzle` respectively, which both use their defined `validateData` functions for validation to check that there are no empty fields, or contain invalid data.

The main concern with a plug-in based architecture is that each web service and puzzle will require different amounts and types of data. Such data can be encapsulated into an XML file to establish a common interface for plug-in interaction. The XML file is generated by the `handleForm` functions of the web service and puzzle plug-in 'Client' class upon validation, using the `DOMDocument` extension of PHP. This is provided as part of the library with no configuration required. Consistency between plug-ins is enforced by requiring event XML to use the schema in Figure 29, and puzzle XML to use the schema of Figure 30. The system uses two schemas to distinguish XML types, such that error-checking can be performed.

```

<event>

  <textual identifier>

    <!-- Content goes here -->

  </textual identifier>

</event>

```

Figure 29: Event XML Schema

```

<puzzle>

  <textual identifier>

    <!--Content goes here -->

  </textual identifier>

</puzzle>

```

Figure 30: Puzzle XML Schema

The `events` table stores the XML data for scheduled events, referencing the `arcs` and `servicesused` table to establish the position within the narrative, and with which linked account to post content to. The `arcPos` field represents the ordering of events in an arc, and is unique to an event within the arc it occurs within, and is maintained by `EventsGateway` rather than the database. The relational view of these tables is shown in Figure 31.

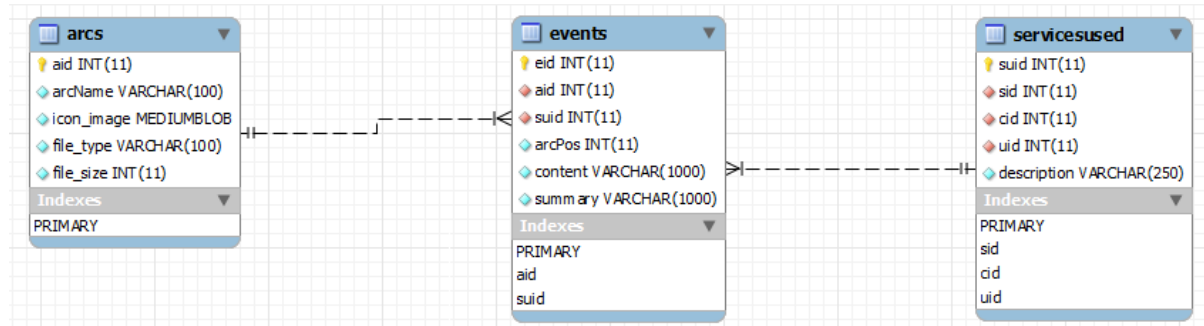


Figure 31: Event Scheduling: Enhanced Entity-Relationship Diagram

Puzzle XML is stored in the `puzzles` table; which references the `puzzlelist` table to identify the plug-in responsible for handling each entry, and the `events` table to identify the associated event. A value indicates if a puzzle has been solved. The relational view of these tables is shown in Figure 32, and demonstrates the implementation of Requirement 5.

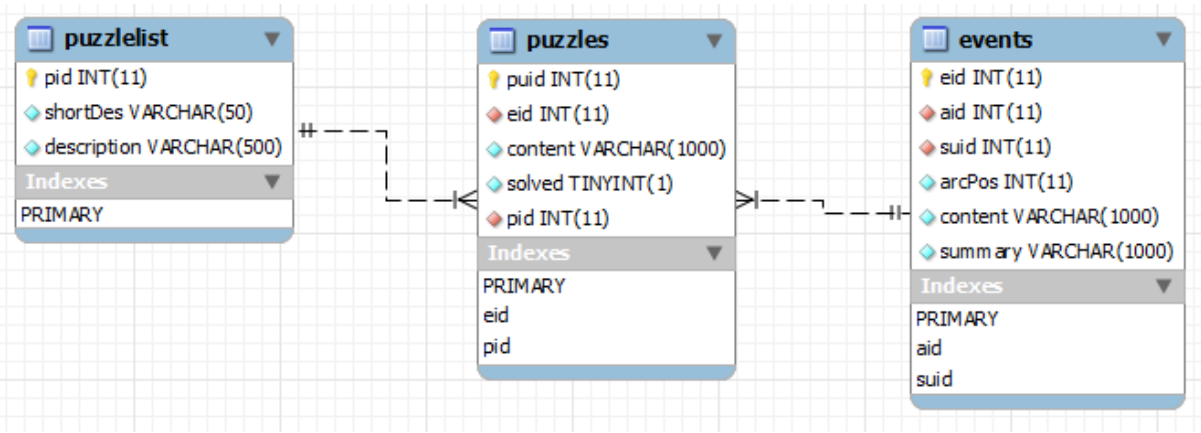


Figure 32: Puzzles: Enhanced Entity-Relationship Diagram

The system implements this feature as described by Figure 33:

Alternate Reality Game Engine

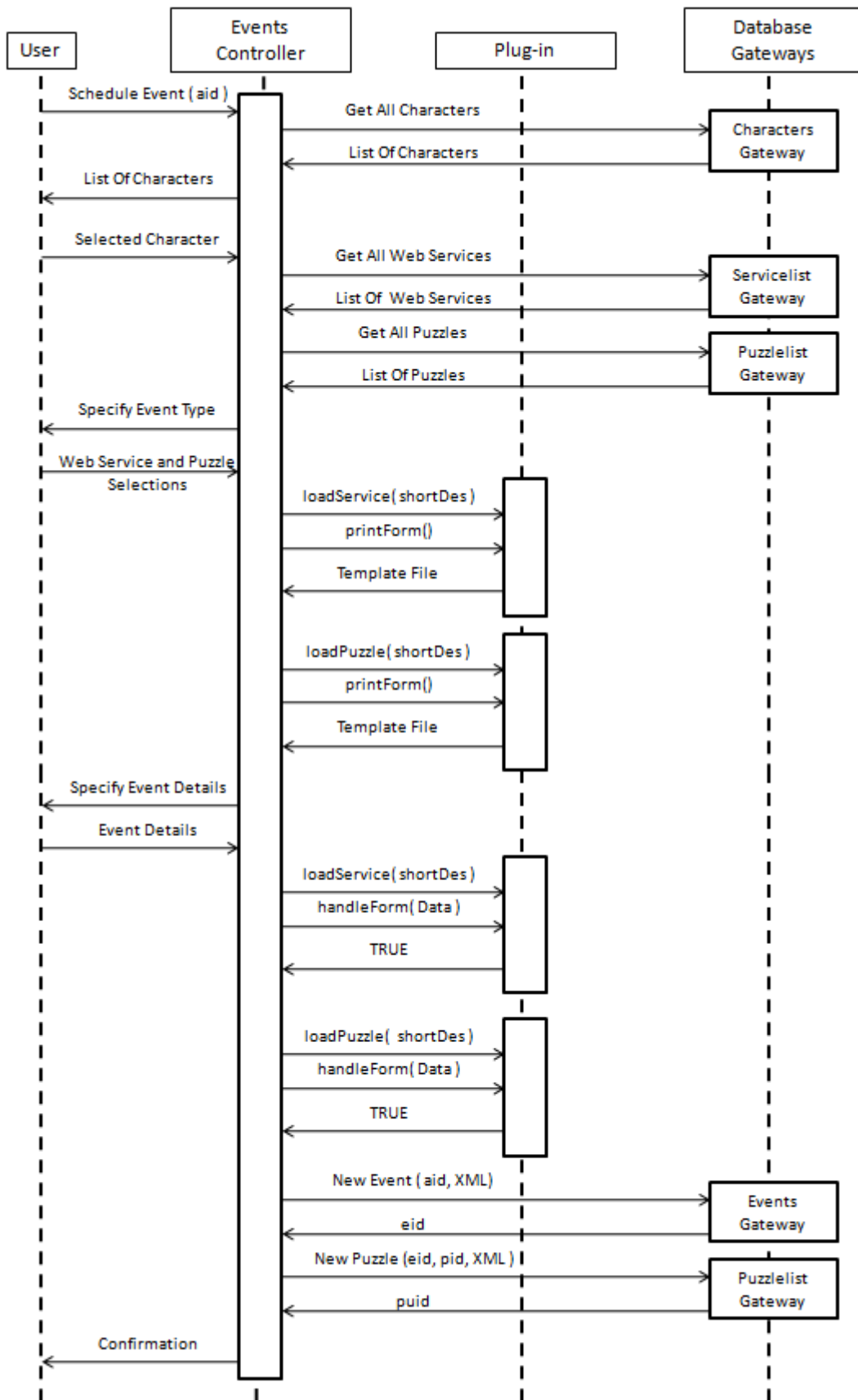


Figure 33: Scheduling An Event Sequence Diagram

3.10: Game Management

This component of the system is responsible for managing a game when it is in progress. There are two types of activities to be performed by the system in this situation:

- Posting event content according to the current position in the narrative. (Requirement 10).
- Detecting puzzle solutions and updating the game state accordingly. (Requirements 11 and 12).

A configuration file stores the overall game state for a game. `Config` acts similarly to the Table Data Gateway pattern to interact with this file, in order to achieve the same benefits as discussed in Section 3.5:. The configuration file is represented in INI format, and `ConfigGateway` is used for data transfer. `Zend_Config_Ini` is used to read INI data, and `Zend_Config_Writer_Ini` is used to write INI data. Both classes represent INI variables in an object-oriented manner for simple reading and writing. Upon calling `write`, any changes are updated to the INI file. Before writing data, `ConfigGateway` performs data validation, such as checking an event id exists within the events table, and returns a `ValidationException` if an error is detected.

The game state can be represented by three fields:

- A Boolean variable represents if the game has been started (`started`).
- An integer identifies the current narrative position (`event`).
- A Boolean variable represents if the current event has been posted (`posted`).

A user starts a game session via a button on the ‘Game’ page. When this button is pressed, the configuration file is modified to represent a game has begun. Actions relating to organising the game data, such as creating or deleting characters, factions, arcs and events, are disabled in order to maintain data integrity. However, the user is permitted to view this data whilst in-game, so the game state can be observed.

The ‘Game Controller’ script is designed with the intention of operating independently to the user interface, such that it can manage a game without user input. As such, the algorithm in Figure 34 was designed to identify the next action to perform, given the state variables in the configuration file.

```

Variables: started - Bool, posted - Bool, solution - Bool, event - Int

BEGIN
IF started
    IF !posted
        PostEvent
        posted = true
    ELSE
        solution = CheckSolution
        IF solution
            event = nextEvent
            posted = false
END

```

Figure 34: Game Controller Algorithm Pseudocode

The algorithm identifies that the current event needs to be posted when `posted` equates to `False`. When `posted` equates to `True`, the current event is currently awaiting a solution, and therefore the controller should search for a solution. The `cron` command schedules an action to be performed on a given periodic basis. In this situation 15 minutes was selected. This allows the game controller to operate on the web-server without user contribution to monitor the game state. A periodic approach was implemented rather than a persistent approach to reduce the server time allocated to game management and limit requests to interact with web services, which may impose limitations of the rate of interactions.

3.10.1 Posting Events

The `postEvent` function of the game controller defines behaviour to post an event. The event identifier specified in the configuration file is accepted as an argument. Using this it is possible to obtain the XML representation of the content, and a reference to a linked web service account from the `events` table. This provides access to authorisation details in the `users` table.

`ServiceLoader` is used to link to the appropriate plug-in. When supplied with a linked account identifier, the plug-in establishes a connection using the relevant authorisation details. Connection to a web service is not guaranteed. Authorisation details are validated upon linking, but if these are altered then the connection will fail. Additionally, if a web service is undergoing maintenance a connection will be denied. These error situations are represented by the `ValidationException` and `NoServiceConnectionException` classes respectively. Upon detection of such an error, the game controller will terminate this execution cycle, and retry upon its next iteration.

Upon successful connection the game controller calls `publish`, passing the XML file. This calls `parse` to translate the XML file into usable data; achieved using the `SimpleXML` extension provided in the PHP library. This extension was used instead of `DOMDocument` as it provides a simpler means of extracting XML data.

Using the connection established upon instantiation of the plug-in, the content is published. An identifier to the post is returned from `publish`, which is unique to that web service only. This is used in order to obtain replies to a post during solution detection, and is stored within the `postedEvents` table, shown in Figure 35. The `posted` variable in the configuration file is set to `True`.

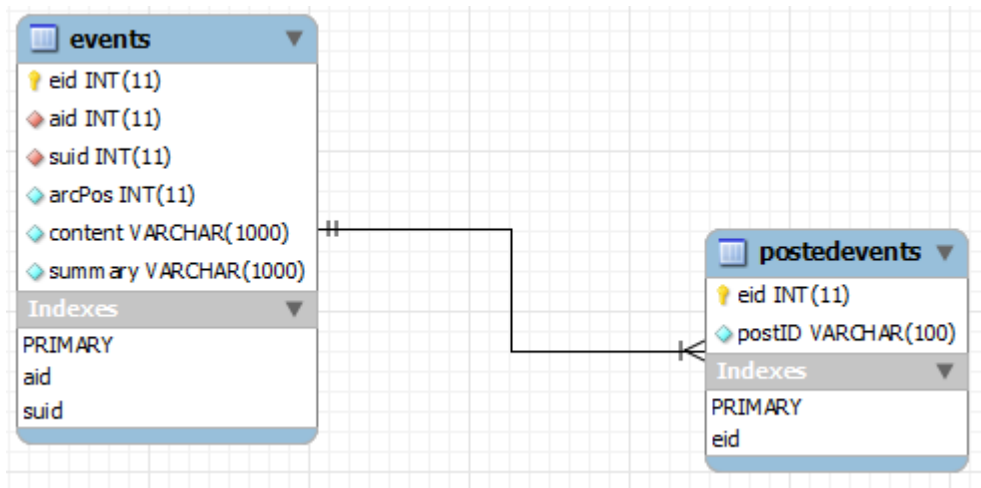


Figure 35: Posted Events Entity-Relationship Diagram

The ‘blogger’ plug-in uses the `Blogger` class of `Zend_Gdata` to publish content. The `publish` function returns an integer identifier to the post. The ‘email’ plug-in uses `Zend_Mail` to construct and send an email. This is used instead of PHP’s built-in `mail` function as `Zend_Mail` permits the use of one connection to post multiple emails. The date-timestamp is used as an identifier to prevent the SMTP `post-id` header field from being altered. This is represented as a `Zend_Date` instance to establish a common representation of time.

Figure 36 demonstrates the use case of this procedure.

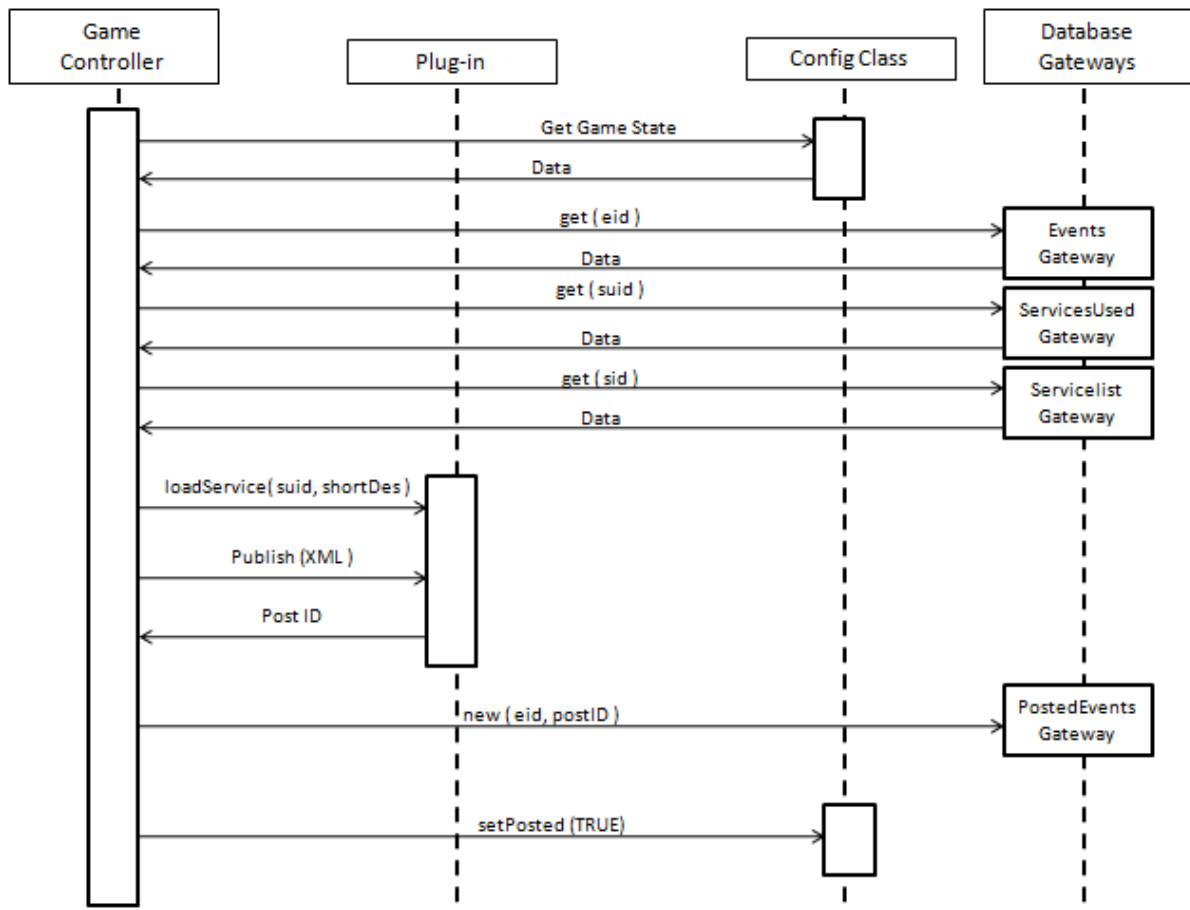


Figure 36: Posting Event Content Sequence Diagram

3.10.2 Solution Detection

If there is no event content to publish then the game controller searches for a solution to the puzzle associated with the current event. This is defined in XML format within the `puzzles` table, which is accessible by providing the event identifier from the configuration file.

To obtain a set of proposed solutions, the game controller requests for a set of replies to the event post, from the web service account used to post it. This behaviour is defined in the 'Client' class of a plug-in. A plug-in is linked using `ServiceLoader` as mentioned above. The `replies` function accepts a post identifier from the `postedEvents` table as an argument, and returns an associative array of replies; thus satisfying Requirement 11.

Puzzle data is loaded from the `puzzles` table by supplying the event identifier. This returns the solution in XML format, and an identifier to the appropriate plug-in to be instantiated using `PuzzleLoader`. The set of replies and the XML representation of the solution are passed to the `solution` function of a puzzle plug-in. Similarly to web service plug-ins, this function calls `parse` to translate the XML file into a workable set of variables. For every reply, `solution` checks if the puzzle has been solved based upon the data in the XML file. If a solution is detected, then the function returns `True`.

If `solution` returns `True`, then the next event within the timeline is triggered. The `nextEvent` function in `EventsGateway` returns the event identifier of the next event in the sequence,

taking into account if a transition between arcs is made. If the game has finished, then `null` is returned. The event identifier is stored within the configuration file, and `posted` is set to `False` to notify the game controller of the appropriate action to take; satisfying Requirement 12.

The ‘blogger’ plug-in obtains a set of replies using the appropriate API call within `Zend_Gdata_Blogger`. The ‘email’ plug-in treats all incoming messages after the date-timestamp as a reply. Whilst this may result in data being incorrectly classified as a reply, it provides a mechanism to accept emails sent in reply which do not contain the `in-reply-to` SMTP header field.

The ‘keyword’ plug-in performs a String-search for a set of user-defined keywords. This is accomplished using PHP’s `strcmp` function.

Figure 37 demonstrates the use case of checking a set of replies for a solution.

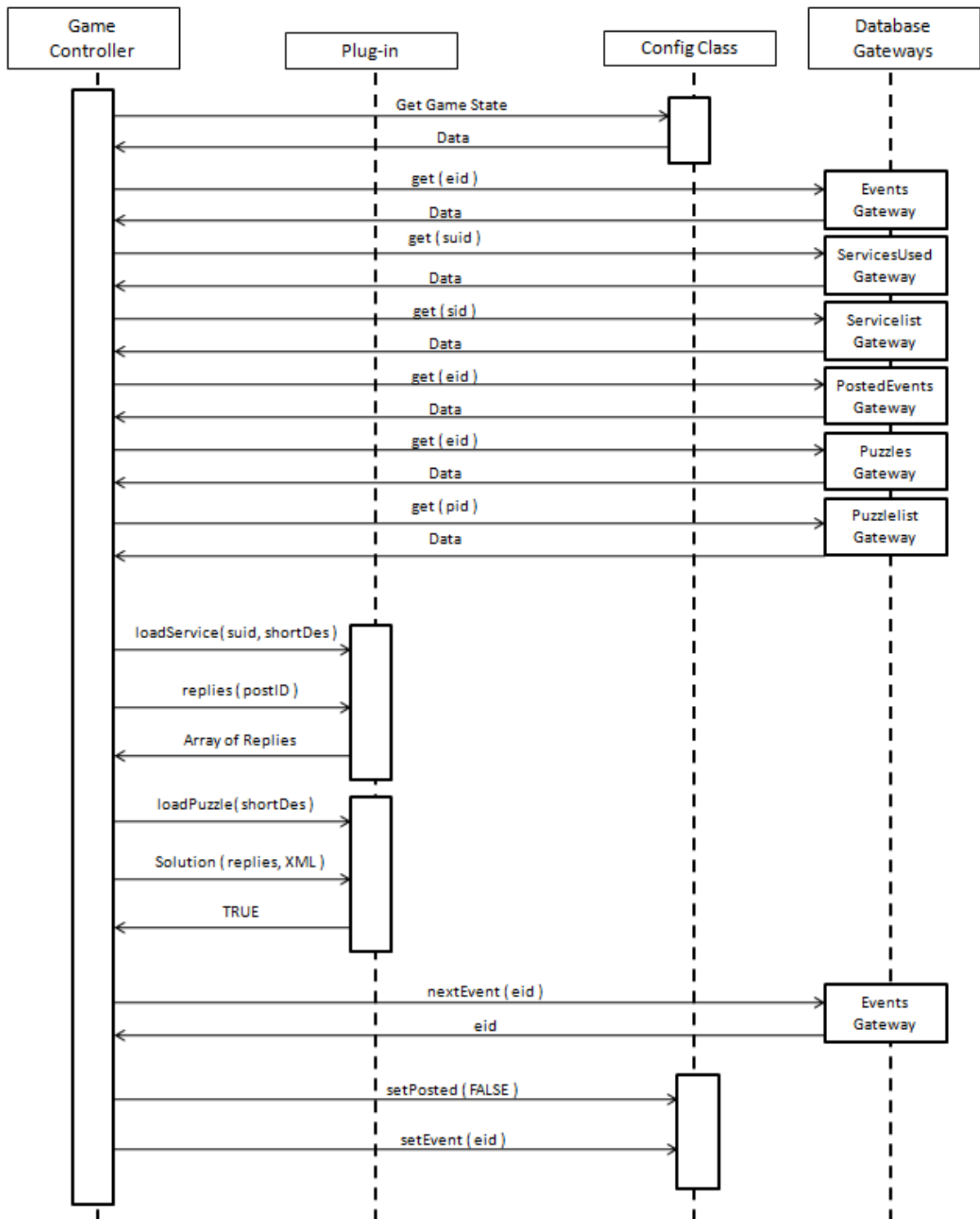


Figure 37: Checking for Puzzle Solutions Sequence Diagram

3.11: Character Interaction Handling

Due to time constraints, character interaction handling specified by Requirements 13 and 14 were not implemented.

3.12: Summary

This chapter has introduced the overall design approach used for this system.

Requirements 1 to 12 inclusive have been satisfied by the system. Due to time constraints, Requirements 13 and 14 have not been satisfied.

The PHP programming language was used for implementation, combined with XAMPP for web-server deployment. The Zend Framework provided a means of simplifying system development whilst implementing communication to many web services. The Smarty template engine was used to separate application and presentation logic.

Implementation was split into five phases according to the modified waterfall model.

A MYSQL database was used for data storage. All data communication commands are encapsulated within classes implementing the Table Data Gateway pattern. This provides a high-level interface to stored data.

A plug-in architecture was designed based upon the Plug-in and Separated Interface patterns. Plug-ins relate to three categories:

- Linking web service accounts to characters.
- Communicating with a web service.
- Solving puzzles.

Thus, the plug-in architecture is used by the various engine components in order to perform their responsibilities:

- The web service and puzzle plug-ins are used by the user interface to represent the game state to the user.
- The game controller uses web service plug-ins to publish plot segments, and obtain replies to posts.
- The game controller uses puzzle plug-ins to detect if a solution exists in a set of replies.

Three plug-ins were implemented to demonstrate this behaviour. The system provides support for Blogger, Emails and keyword scanning.

The game controller is periodically scheduled for operation on a web-server to monitor the game state, achieved using the `cron` command. Its responsibility is to post event content and check for solutions to puzzles.

Chapter 4: Testing

4.1: Introduction

This chapter discusses the two aspects of testing performed on the system and provides a detailed analysis of the results obtained from these tests.

4.2: Unit Testing

Unit testing was performed before any further user testing was carried out. This was in order to provide a degree of confidence that the system demonstrates a working implementation of the system requirements.

Such tests were automated using test harness classes, in which test cases are inputted into the system. The system response is logged such that the success rate can be monitored. Using test cases which contain both valid and invalid output ensures that data is being correctly stored or interpreted, and that validation checks are functional. Consequently, the system can be modified to account for any erroneous cases that are detected, to provide a more stable environment for user testing.

This testing was designed to prove that each of the implemented requirements was satisfied in practice. In order to accomplish this, several basic functions of the system must also be tested. The following aspects of the system were tested during the unit testing phase, and where necessary both valid and invalid input is supplied:

4.2.1 Database Communication

For each of the gateway classes, the following actions were performed:

- Update an existing entry.
- Select an existing entry.

Inserting and deleting data to each of the database tables is explicitly handled by later tests.

4.2.2 Configuration File

For the configuration file, the following actions were performed:

- Read every variable.
- Write to every variable.

4.2.3 Plug-ins

These tests relate to specific actions implemented by web service and puzzle type plug-ins, including:

- Instantiate plug-ins using all factories.
- Call every function implemented by plug-ins whilst no plug-in is loaded.
- For each web service plug-in, attempt to post event content. The database will be modified directly to contain invalid data, where necessary.
- For each web service plug-in, obtain a set of replies for an event.
- For each puzzle plug-in, check for a solution to a given set of replies.
- For each plug-in, validate any required data.

4.2.4 Game Management

These tests relate to the actions performed when a game has been started:

- Given a set of game states and any appropriate data in order to match such states, run the game controller script.

In addition, the following tests will be performed manually using the web forms provided by the system. These tests represent the passage of data through the system, building upon the automated testing:

4.2.5 Character Management

These tests relate to the system response to user data provided as a web form:

- Create and delete a character.
- Create and delete a faction.
- Add and remove a faction member.
- Link a web service.
- Delete a linked web service.

4.2.6 Storyline Management

These tests relate to the system response to user data provided as a web form:

- Create and delete an arc.
- Create and delete an event.

`Zend_Log` logs a user's progression and choices when interacting with the system. The resultant log file will be examined for specific details regarding a failed test case, and will be used as the framework for implementing a solution. Upon the failure of any test case, a repeat test will be initiated after a solution has been implemented. `Zend_Log` will also be used to store a second log file logging the output of the test cases.

4.2.7 Results

Appendix A contains the formatted output from unit testing. The results show that all the test cases outlined above, with the exception of one, all passed with regards to the invalid and valid data supplied – demonstrating that the system can accommodate from any errors arising from test cases above, and also provides evidence that the system correctly validates and handles input.

The test cases identified an issue relating to an incorrect interpretation of a string of keywords. Should a user use a quotation mark within a keyword, the system would disregard any of the string content following this quotation mark; triggering an exception. The scope of this error is potentially dangerous, as it identifies a vulnerability to SQL injection attacks. This was solved by correctly passing the keyword string to a safety function, which would add an escape character to the quotation mark. This test case passes a repeated test after implementing this fix.

4.3: User Testing

4.3.1 Test Description

User testing involves comparing the traditional method of creating an ARG with using the system to create an ARG, to establish if the benefits of using an engine are demonstrated in

practice. In order to compare these methods, data will be gathered regarding the performance of the user for each method. More specifically, the following aspects will be measured:

- How ‘restricted’ a user feels in terms of the availability of actions they are able to perform.
- The availability of help.
- The ability for the user to detect a mistake and take corrective action.
- The ability for the user to conceptualise their actions.
- The ease of performing the desired task.

Users were invited to create an ARG using the two methods. The ARG to be created was of the user’s own creation, with no restrictions upon the nature of their game to reduce the risk of introducing bias. Testing, therefore, took place in four phases. Users were briefed according to a pre-prepared script which introduced the concept of ARGs, and the nature of the test. This script is located in Appendix B. This ensured that bias was not introduced throughout the test by establishing a consistent starting point. The script gave no indications as how to perform either method, thus the testing of each method ended according to the user’s interpretation of their situation.

The second and third phases involved creating the ARG manually and using the system. The order of these phases is varied according to a “*Latin Square design*”⁽²²⁾ shown by Figure 38. This is to reduce the effects of the data being distorted due to effects such as varying levels of concentration.

Test Case	Test One	Test Two
One	By Hand	Using Engine
Two	Using Engine	By Hand
Three	By Hand	Using Engine
Four	Using Engine	By Hand

Figure 38: Latin Square Test Ordering

Testing took place on one workstation, which was set up with the engine pre-configured. The three plug-ins implemented during development were pre-installed in order to represent a typical ‘out-of-the-box’ setup. The author observed each user test and made notes regarding the user’s performance, and did not contribute during tests to ensure the resulting data was not distorted.

The system log file for each user was retained for analysis. This gives an insight into the user’s navigation through the system, identifying bottlenecks within the system design, and areas where users make errors.

4.3.2 Questionnaire Design

The fourth phase required users to complete a questionnaire. Questionnaires provide a means for mathematical analysis to be applied to identify correlations and associations within the data obtained. The questionnaire is structured into three sections:

4.3.2.1 About ARGs

This section contains two questions, to provide background information regarding the user's knowledge of playing and creating ARGs. This is used in order to contextualise the answers given to later questions. These questions are represented as a binary choice. A neutral option was not included as users were provided with a briefing in phase one of testing.

4.3.2.2 Creating ARGs

This section compares the two methods through five four-part questions. This section is loosely based on the sample performance questionnaire provided in Dix et.al (2004) ⁽¹⁶⁾. The two methods act as the two possible values of the independent variable, which is the method used to create an ARG. The dependent variable of the questionnaire is therefore the user's performance for each method, comprising of the five aspects highlighted above.

All five questions in this section adopt a common format, described in Figure 39. A statement (the 'question-statement') relating to an aspect of performance is made. For each method, the user is required to rate their level of agreement using a four-point Likert scale. A four-point scale is used such that an identifiable positive or negative answer is obtained, whilst providing "clarity in meaning" ⁽¹⁶⁾. By using two Likert scales, the statement can be worded such that no bias towards either method is introduced. Instead, the user is able to independently assess both techniques rather than being forced into making a contrasting comparison. In order to reduce data distortion due to ordering effects, the arrangement of the two methods is irregularly switched.

"Statement"

a. Method 1: Strongly Agree [] [] [] [] Strongly Disagree

b. Method 2: Strongly Agree [] [] [] [] Strongly Disagree

c. Which method do you prefer in this case?

Method 1 [] [] Method 2

d. Why have you made that choice?

Figure 39: Questionnaire Section 2 Template

Data obtained from Likert scales is of the ordinal type, representing quantitative, objective data. The user's response can be assigned a number, such that 1 = strongly disagree, 4 = strongly agree. These values can be ordered; however, as Likert scales do not represent a user's own interpretation of the difference between the levels of a scale, it is not possible to analyse such data using interval type techniques. ^{(23) (22)}

The third sub-question requests a binary comparison between the two methods in order to act as a summary for this question. This answer is contextualised using an open question, with the justification of the user's choice to sub-question three as a framework for their answer. Open questions relate to qualitative data which cannot be mathematically analysed, but instead provides subjective data to complement the quantitative data.

4.3.2.3 Overall comparison

The third section of the questionnaire requires a user to provide an overall summary of their preference as to which method they would use in future. Similarly to earlier comparative questions, a binary choice approach is taken. This provides quantitative data of the nominal type. Qualitative data is obtained using an open question, in which the user justifies their overall choice.

The full questionnaire is shown in Appendix C.

4.3.3 Results

Six users were invited to test the system, and were contacted via an advertisement placed on a social networking website. Four of these users were present for the actual testing, demonstrating a 66% turnout rate.

The following diagrams represent a graphical view of the questionnaire data. The raw data results are shown in Appendix D. Appendix E contains the log files for each test.

For results where a Likert scale was used, 1 represents ‘strongly disagree’ and 4 represents ‘strongly agree’.



Figure 40: Question 1 Bar Chart

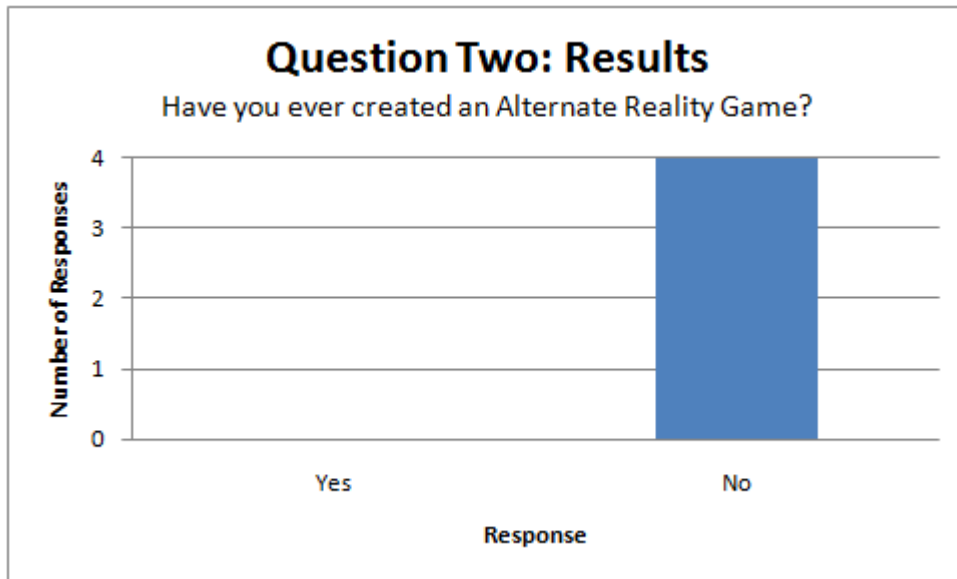


Figure 41: Question 2 Bar Chart

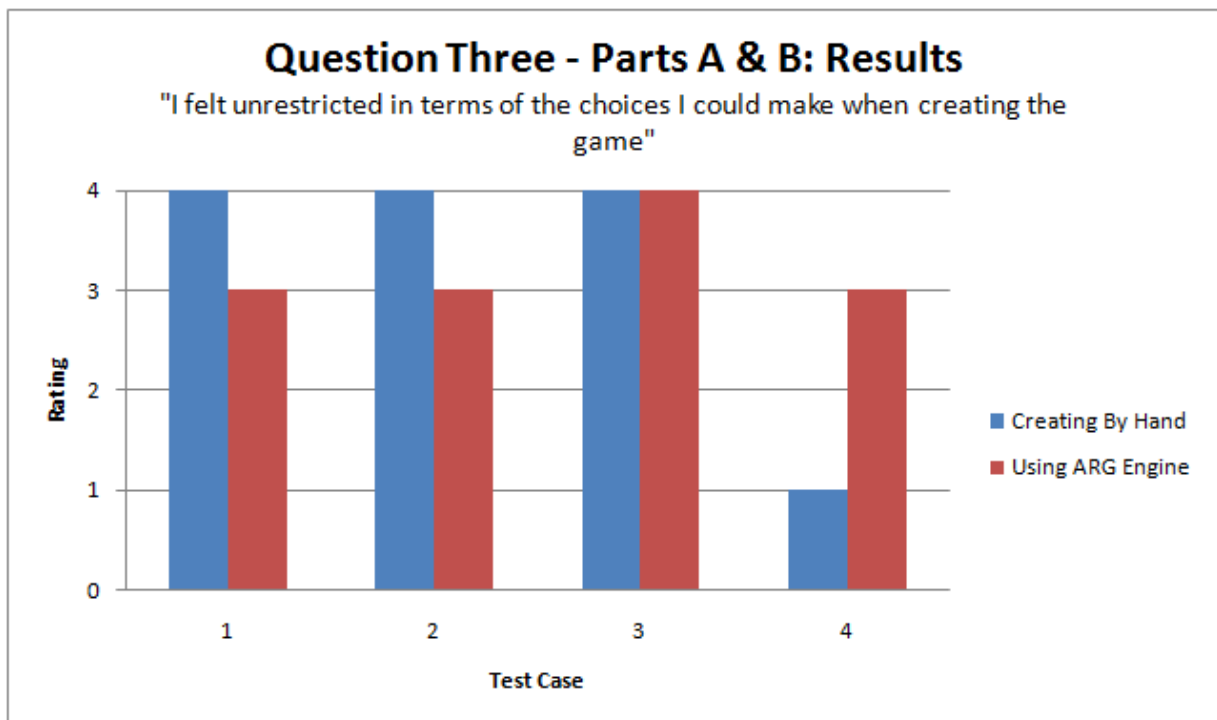


Figure 42: Question 3, Parts A&B Bar Chart

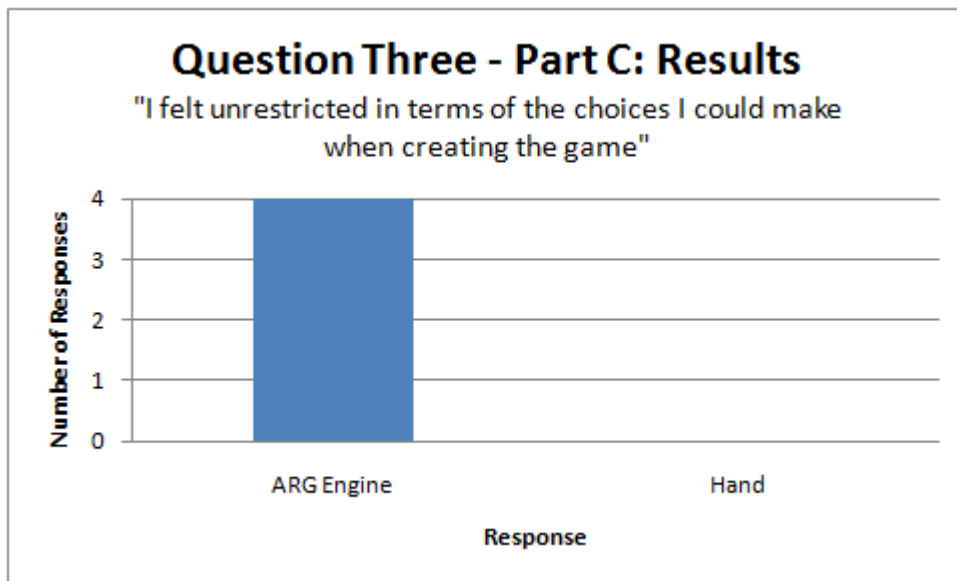


Figure 43: Question 3, Part C Bar Chart

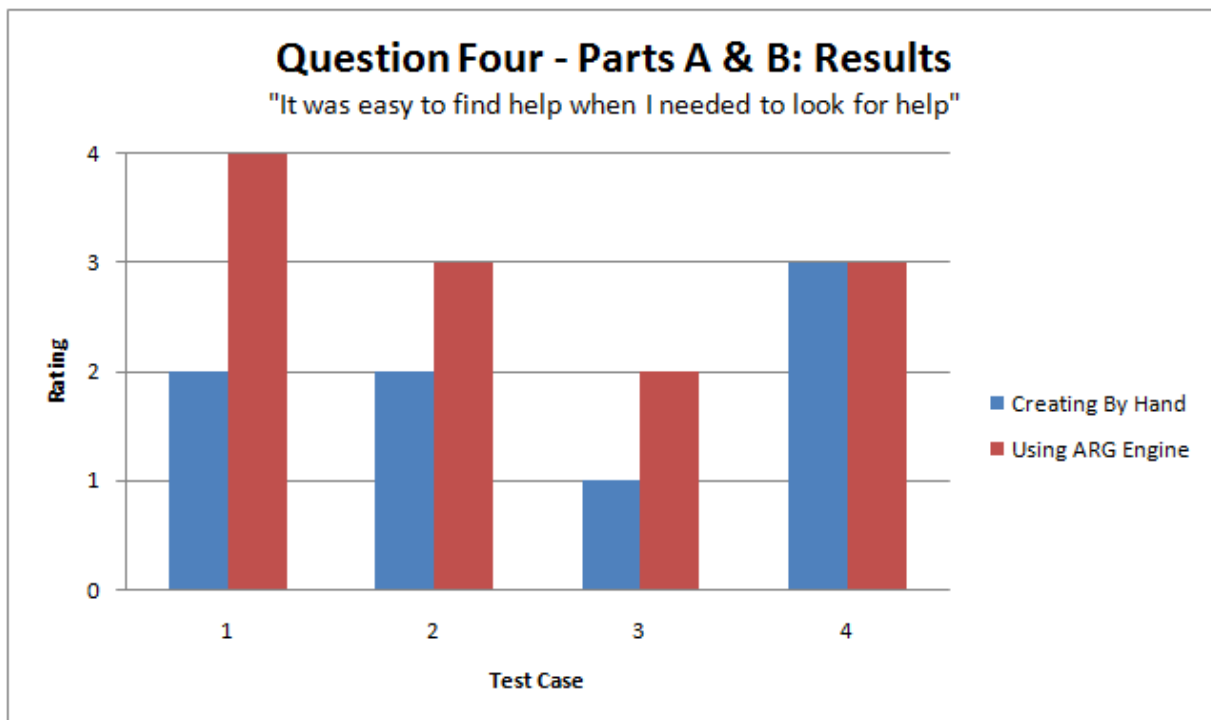


Figure 44: Question 4, Parts A&B Bar Chart

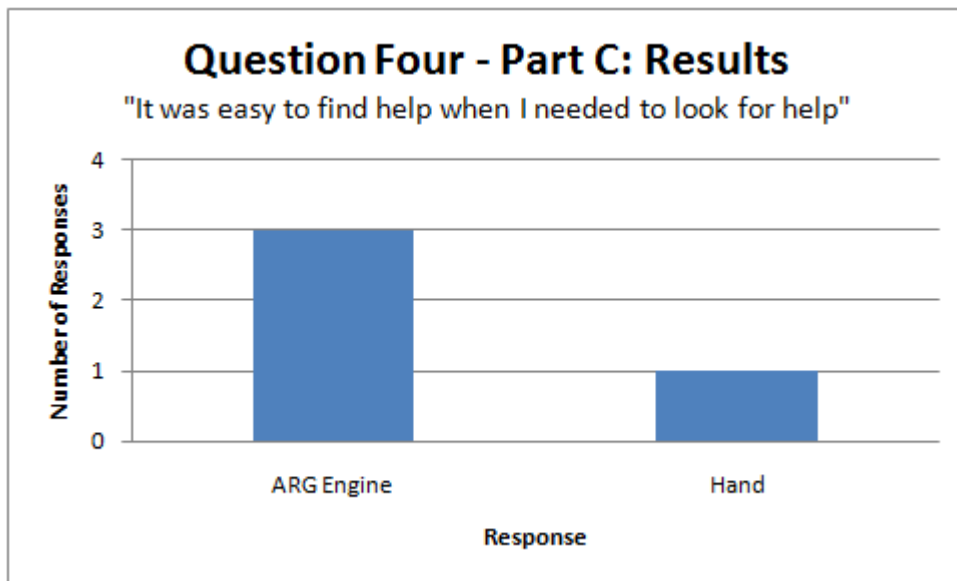


Figure 45: Question 4, Part C Bar Chart

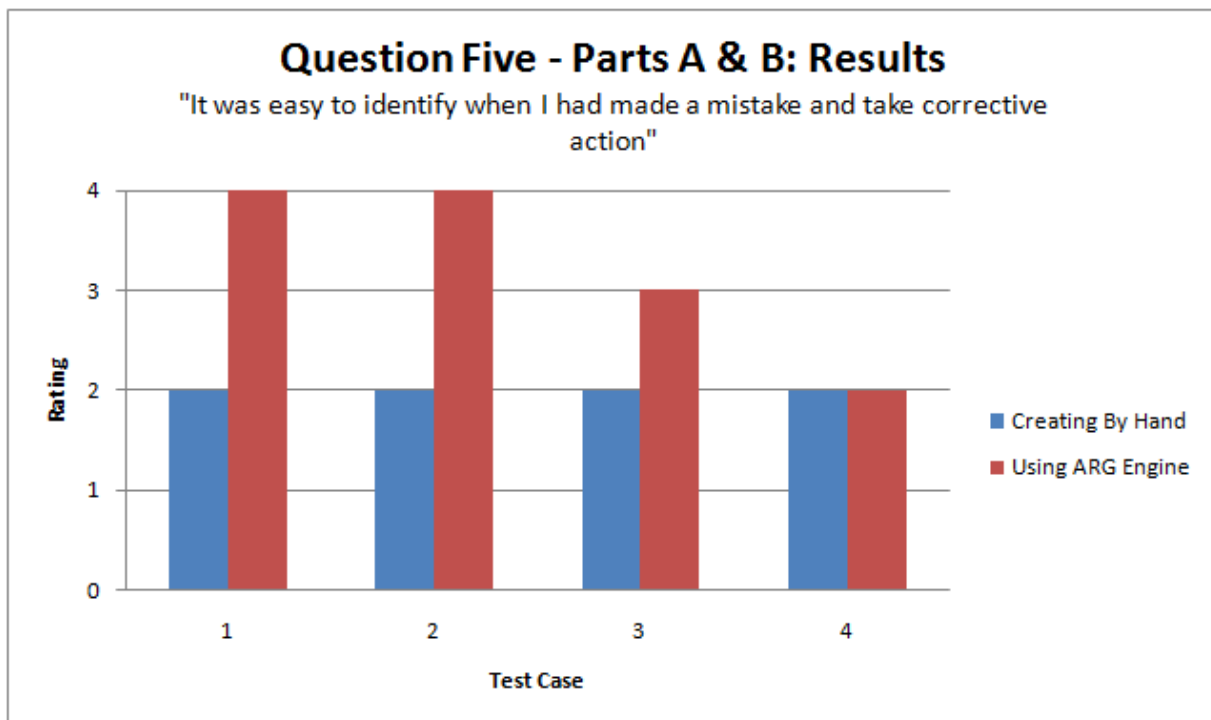


Figure 46: Question 5, Parts A&B Bar Chart

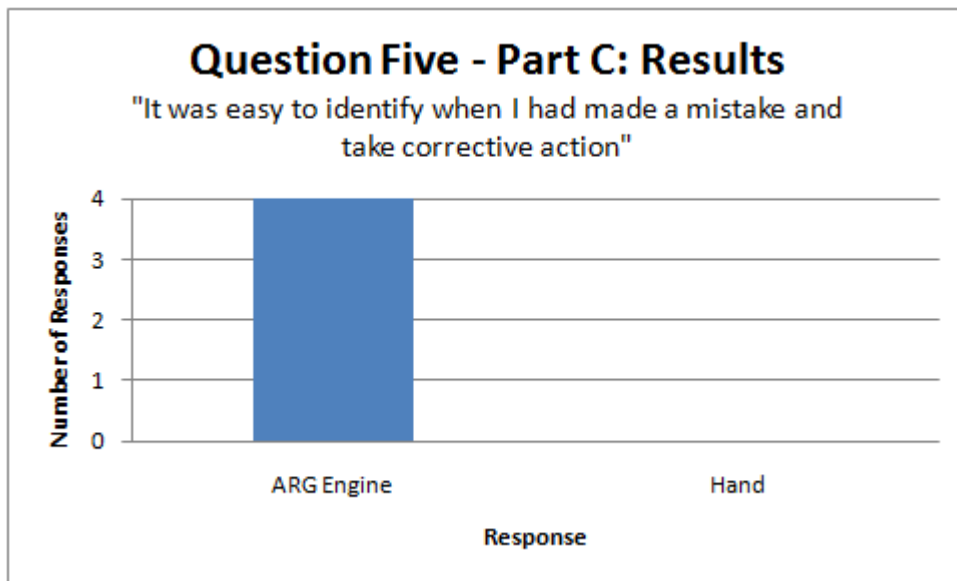


Figure 47: Question 5, Part C Bar Chart

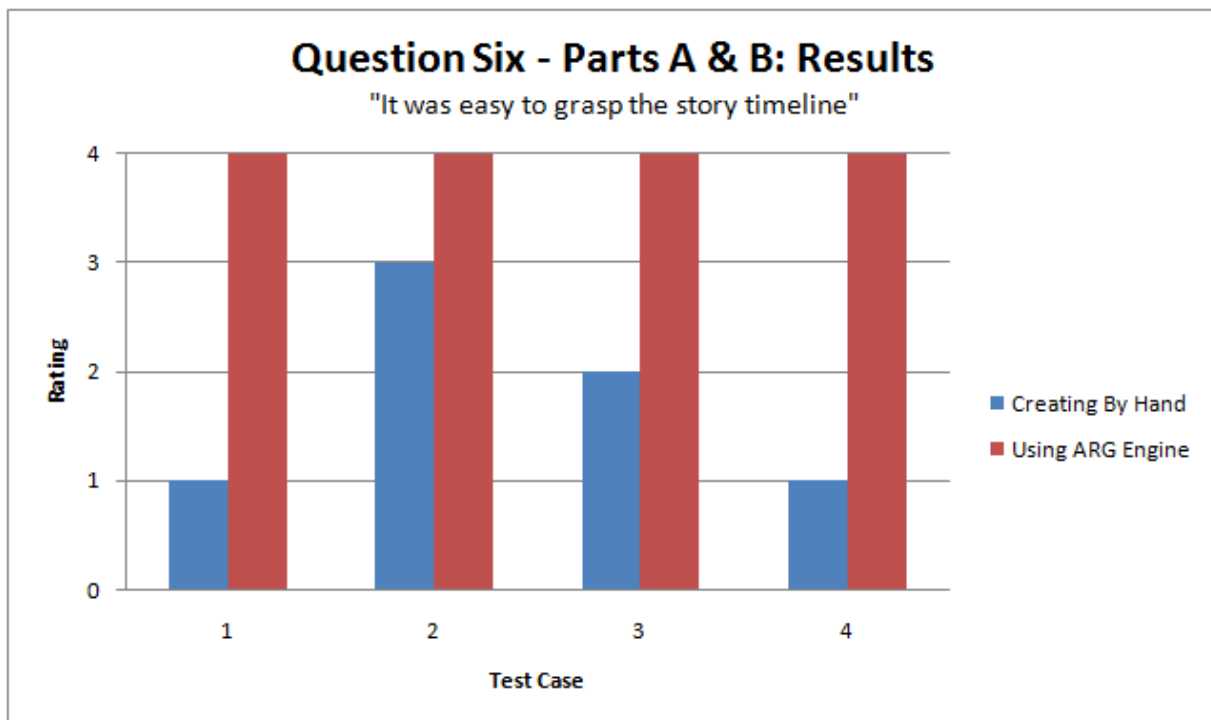


Figure 48: Question 6, Parts A&B Bar Chart

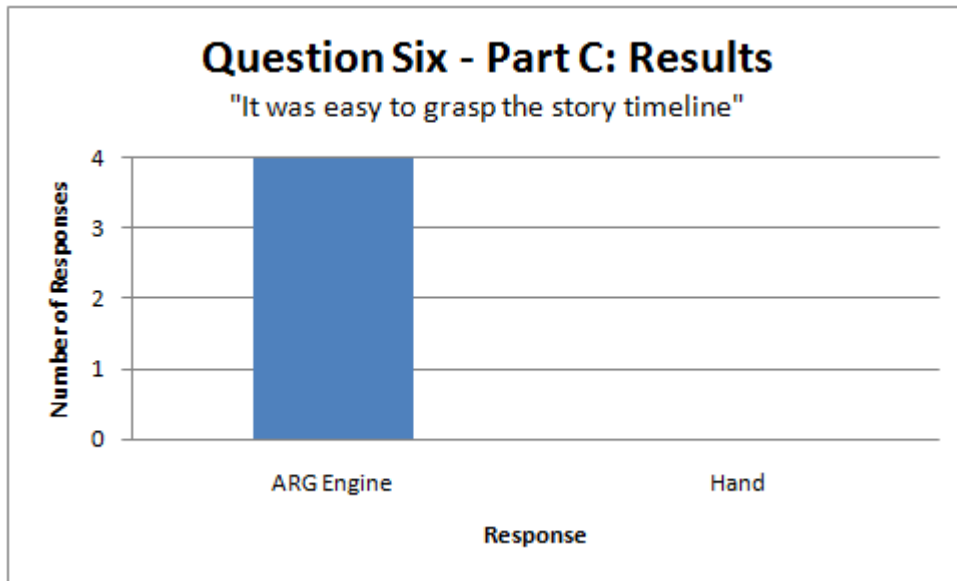


Figure 49 : Question 6, Part C Bar Chart

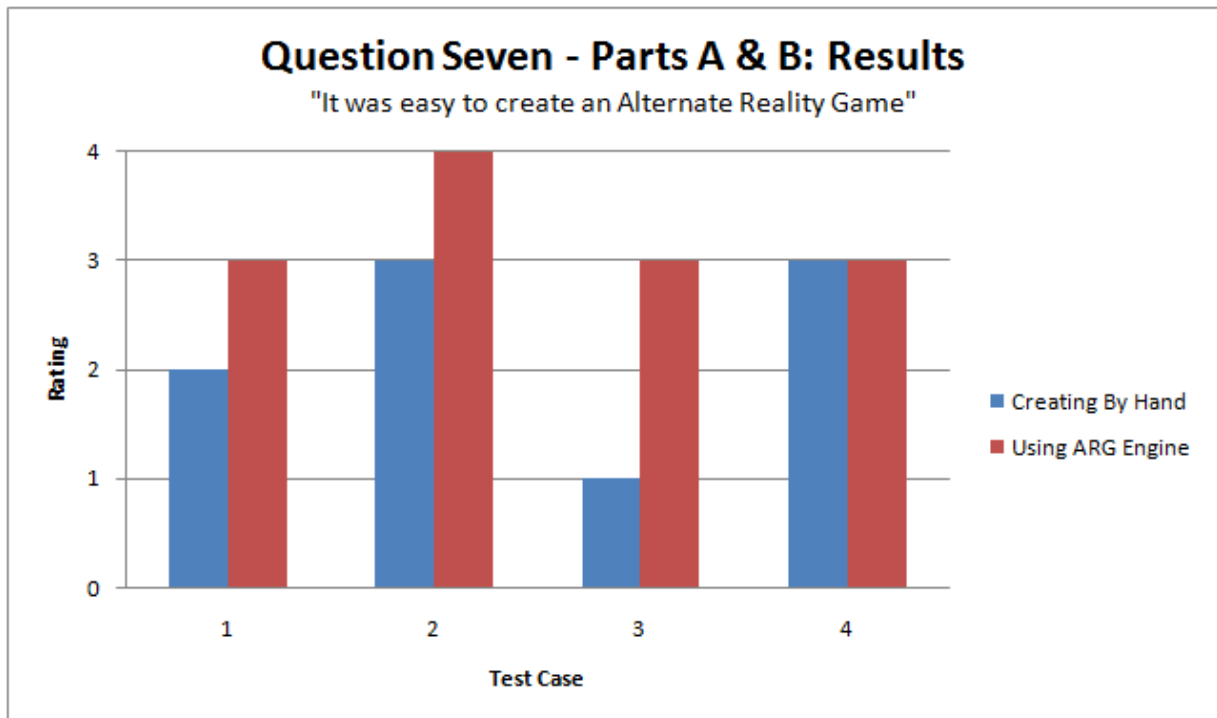


Figure 50 : Question 7, Parts A&B Bar Chart

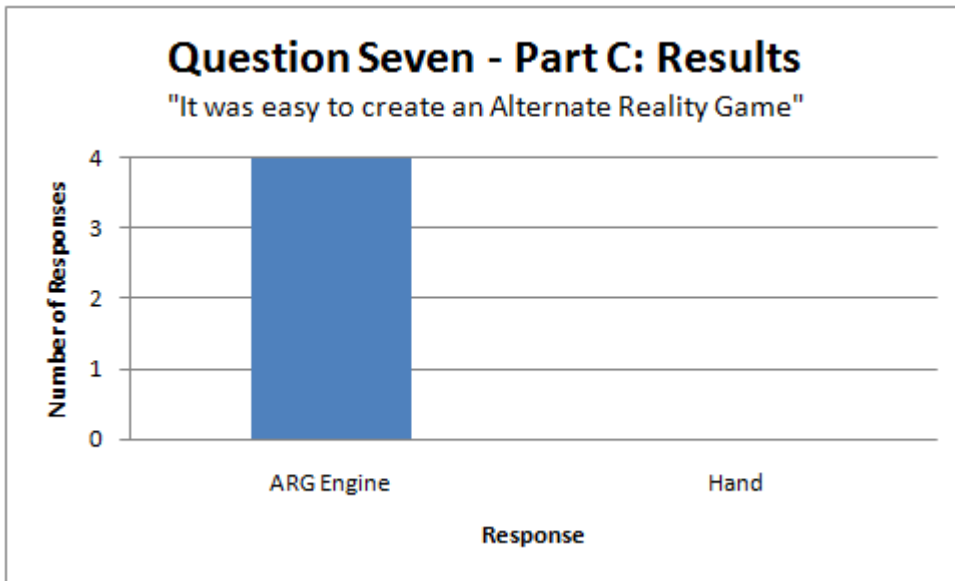


Figure 51: Question 7, Part C Bar Chart

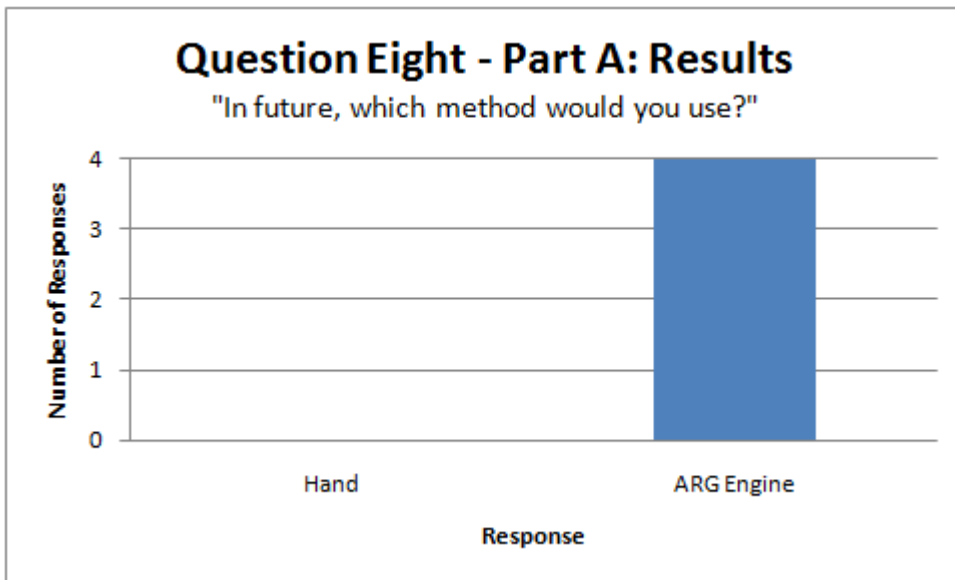


Figure 52: Question 8, Part A Bar Chart

4.4: Results Analysis

The results from questions one and two indicate that all users had no previous experience with playing or creating ARGs, as shown by Figure 40 and Figure 41 respectively. Thus, this contextualizes the remaining results as from a beginner's viewpoint.

A Likert scale represents ordinal data, which is best-suited to be evaluated using a non-parametric statistical test. As the structure of questions Three to Eight involves altering one independent variable with two samples, the Mann-Whitney U-Test is the most appropriate statistical analysis method⁽²³⁾. However, given the small sample sizes for this set of data, this test will not give an accurate measure, and therefore this test was not used in order to reduce the chance of false inferences being drawn⁽²²⁾.

Question three described how unrestricted a user felt when creating a game. Overall, Figure 42 demonstrates that users tended to agree more with manual creation compared to the system. Figure 53 shows that 75% of users ‘strongly agreed’ with the question-statement for manual creation, however this drops by 50% for the system; instead 75% of users ‘agreed’. In test cases one and two Twitter was used in order to post events. As this is not implemented in the system the users had to switch to a supported web service. Figure 42 confirms both users demonstrated this restriction in their answers by choosing ‘agree’ for the system. Despite this restriction, Figure 43 shows all users preferred to use the system in this sense. 75% of answers to Part D contained the word “*easier*”, whilst 25% contained the word “*quicker*”. The choice of words indicates that some restrictions in terms of grasping how to use each web service are alleviated by the system.

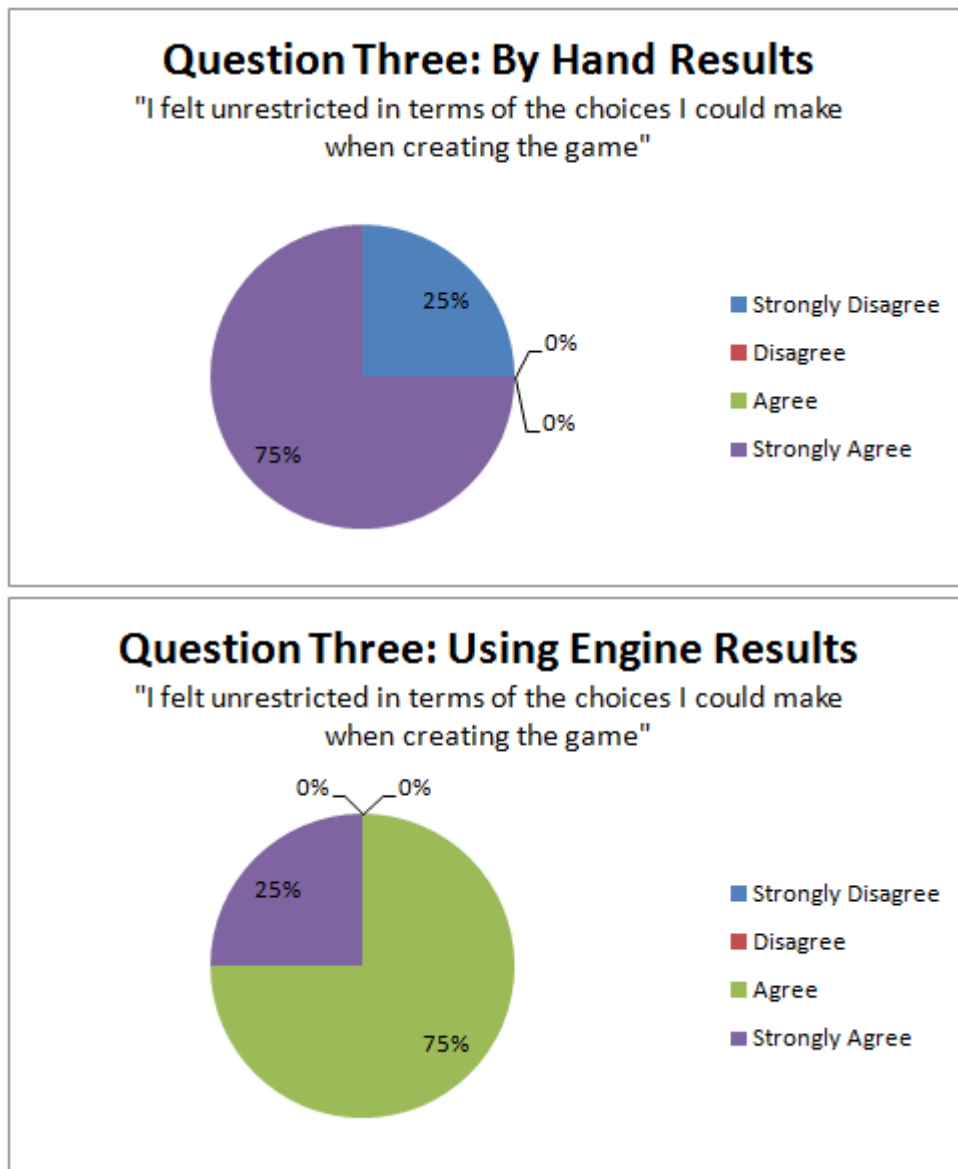


Figure 53: Question 3, Parts A&B Pie Charts

Question four relates to the availability of help. Figure 44 shows that users tended to agree with the question-statement when using the system and tended to disagree for manual creation.

Figure 54 demonstrates that 75% of responses were of a negative nature for manual creation, whereas 75% of responses were positive for the system. In both cases, 25% ‘strongly’ reacted. Part D identifies some reasoning. Test case one identified that when using the system, “*explanations were available on each page*”, whereas for manual creations test case two felt you are “*completely on your own*”. This is not strictly true; some help as to using a web service is available on the vendor’s website (as identified by test case four). However, overall, users perceived the availability of help to be lower for manual creation, and as such Figure 45 shows that 75% of users would prefer to use the system for this aspect.

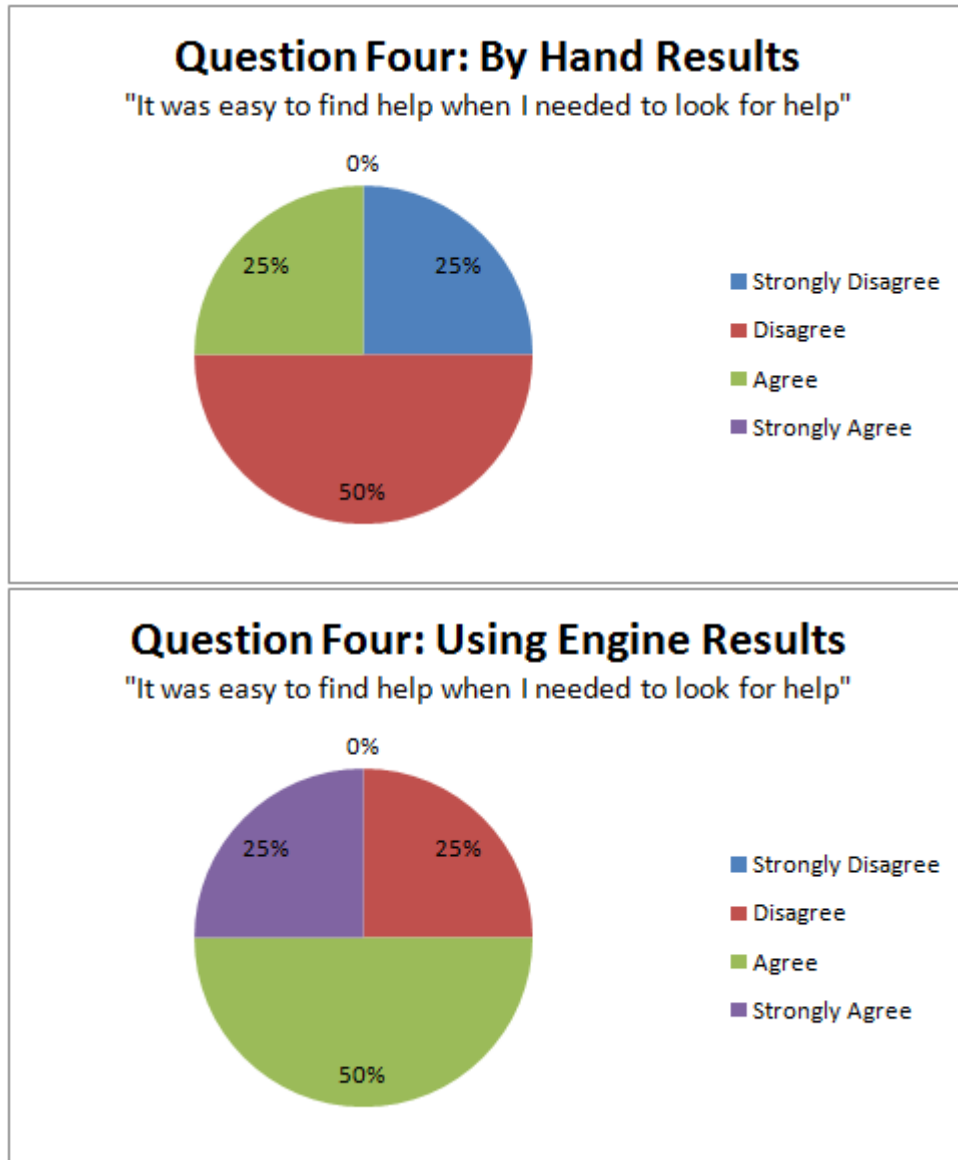


Figure 54: Question 4, Parts A&B Pie Charts

Question five relates to how easily a user can recognise a mistake, and take corrective action. Figure 46 shows a large level of agreement for the system, and a consistent level of disagreement for manual creation. 75% of users agreed with the question-statement, 50% ‘strongly’. Indeed, 100% of users disagreed with the question-statement for manual creation, as shown in Figure 55. Users tended to agree for the system due to the deployment of error

checking and data validity tests. Test case four identified that the system “*tells you what you have done*” due to automatic event tracking. Test case two stated that this allows users to “*easily see where you are going wrong*”. Test case four does not match the trend for the system, instead stating no difference between the ability to recognise mistakes for both methods. However, Figure 47 demonstrates that 100% of users preferred to use the system for this aspect, providing evidence that test case four may contain anomalous data.

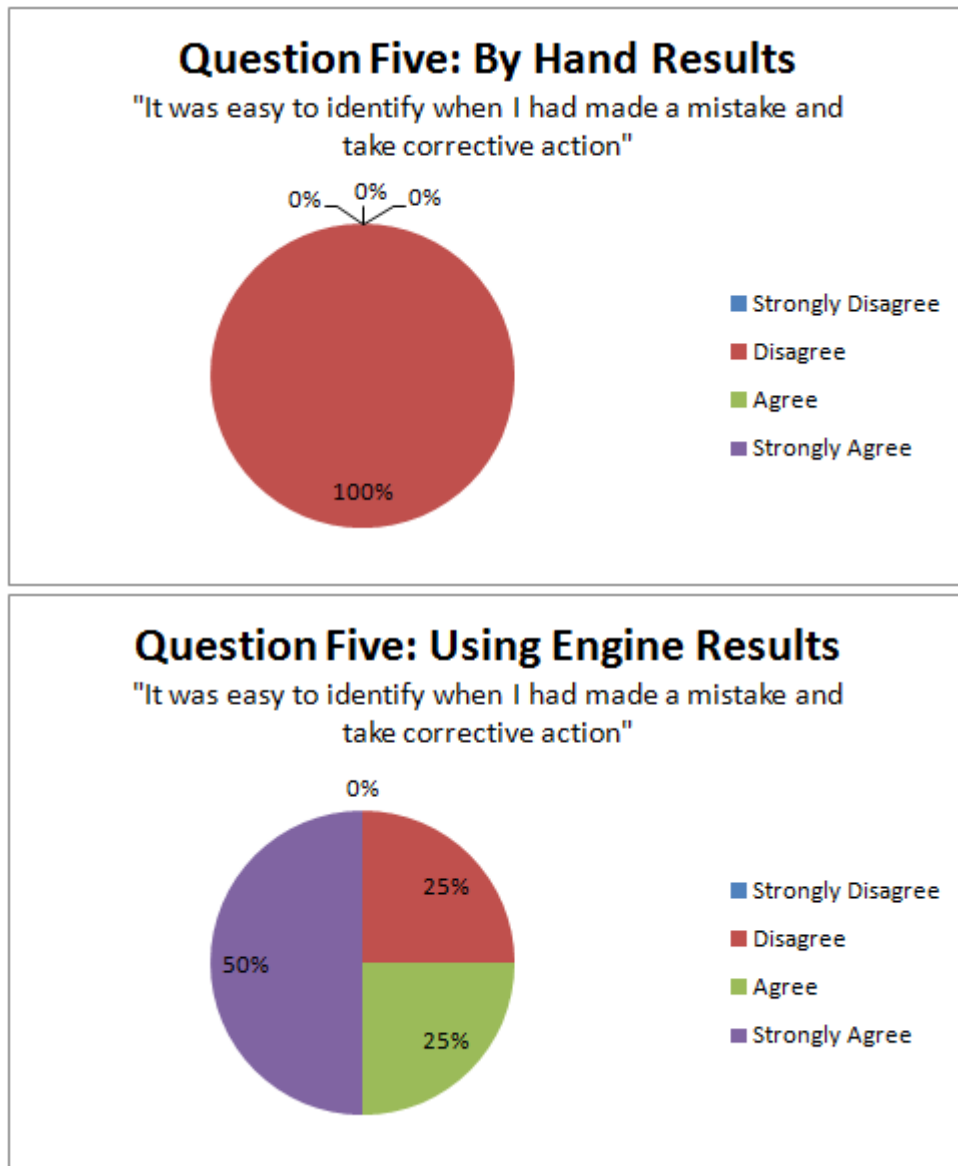


Figure 55: Question 5, Parts A&B Pie Charts

Question six tests how easily a user can perceive the game storyline. Figure 48 shows an overwhelming level of agreement for the system, and a trend of disagreement for manual creation. This is complemented by Figure 56, which shows that 100% of users ‘strongly agreed’ that it is easy to perceive the storyline, whereas only 25% of users agreed, albeit not ‘strongly’ that this is the case for manual creation. Figure 56 identifies that 75% of responses disagreed with the question statement, with 50% of responses ‘strongly’ disagreeing. Part D reveals that the flowchart representation presented by the system was, according to test case

three, “*simple and obvious*”, and according to test case four, “*easier to read*”. Only test cases one and four decided to map their storyline manually. Figure 48 shows that both these cases ‘strongly agreed’ when using the system, and ‘strongly disagreed’ for manual creation, adding validity to the author’s opinion for automatic flow chart generation. Due to this, Figure 49 shows that 100% of users preferred to use the system for this aspect.

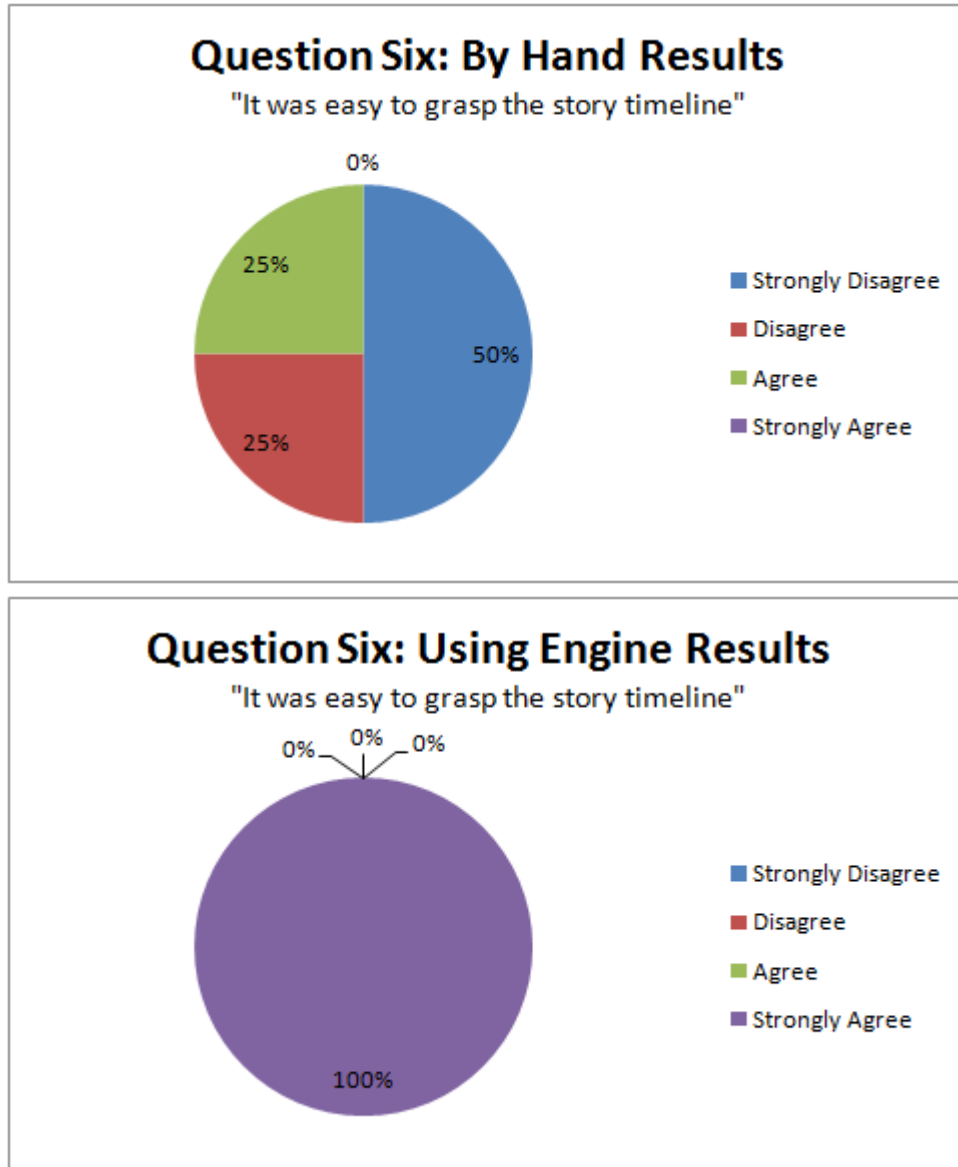


Figure 56: Question 6, Parts A&B Pie Charts

Question seven tests how easily a user can create a game using either method. Figure 50 shows an overall level agreement for using the system, compared to a mixed response for manual creation. Figure 57 shows that 100% of responses agreed that it is easy to create a game using the system; where 25% of these responses ‘strongly agreed’ with the question-statement. In direct comparison, there was a 50% reduction in the level of agreement for manual creation, which instead swung to a 50% level of disagreement. Whilst there is an even match of responses that agreed and disagreed for manual creation, 50% of the responses that disagreed (25% of overall responses) ‘strongly disagreed’. This indicates a stronger leaning towards

disagreement, identifiable in Figure 50. Evidence towards this is gathered from Part D, in which test case two stated that using the engine introduced “*speed and automation*”. This claim was reiterated by test cases three and four. As such, Figure 51 identifies that 100% of users preferred to use the system for this aspect.

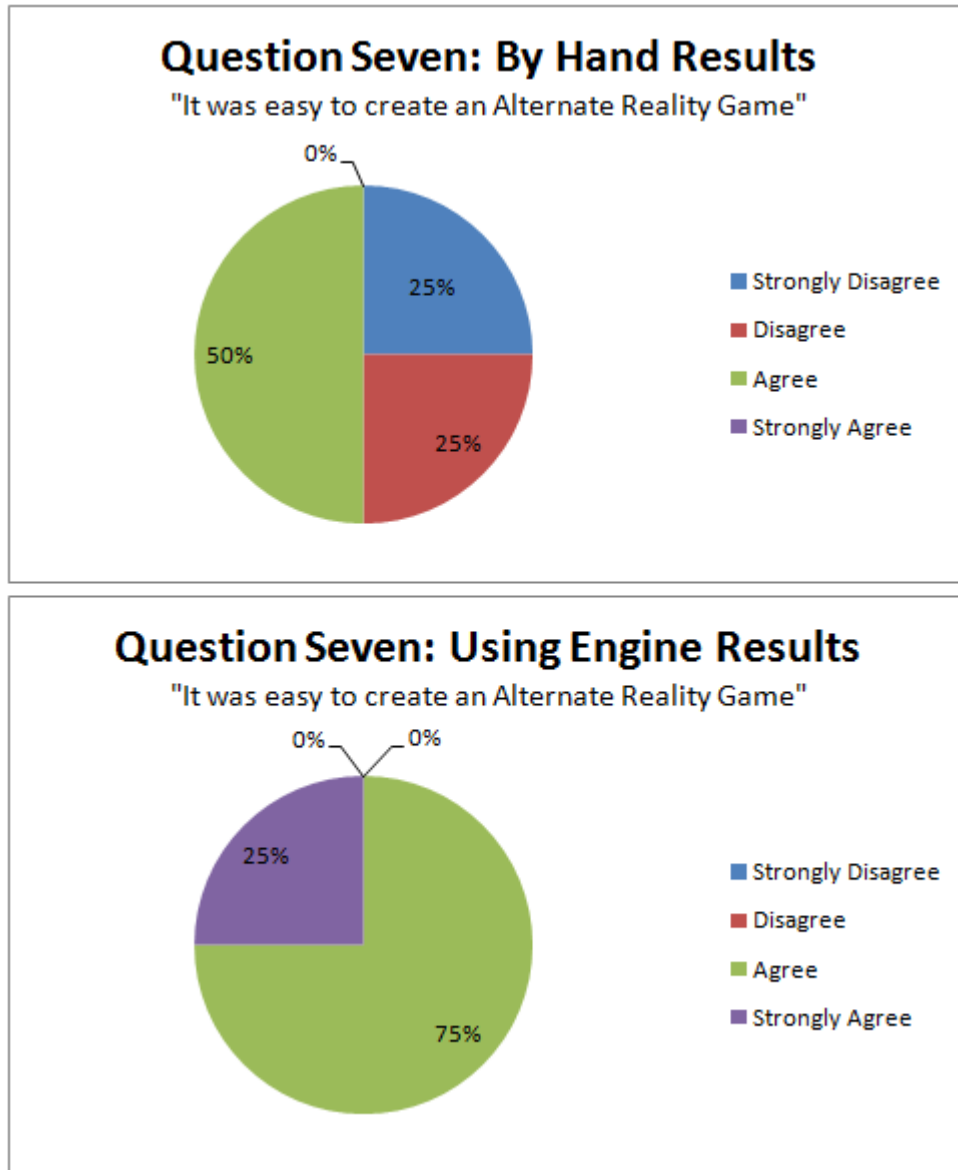


Figure 57: Question 7, Parts A&B Pie Charts

Question eight was a summary question, in which users would provide an overall, generalised opinion on the two methods. The response to Part A of this question is shown in Figure 52. This identifies that 100% of users would prefer to use the system in future as opposed to manually creating an ARG. Part B was an open question allowing users to justify their selection. Test case three identified their choice was made due to the system’s method of “*centralised management*”. This opinion was reinforced by similar opinions from the remaining test cases, and provides strong evidence that, from the perspective of a beginner, using an engine to act as a framework for developing presents an overall easier method of creating and managing an ARG.

The main concern identified from the log files of the users indicates confusion relating to the location of the link account functionality. Users became confused with ambiguity regarding the 'Plug-in' section heading. This section states all plug-ins the system can handle, rather than defining functionality to link an account with a character. All four test cases identified this problem, identifying a shortcoming of the original system design.

4.5: Chapter Summary

Two types of testing methods were performed to gather a complete representation of the system that was developed:

- Unit testing was performed via test harness classes - demonstrating that the requirements marked as satisfied in Chapter Three were satisfied in practice.
- User testing was carried out to provide an answer to the hypothesis outlined in Chapter One. Users were invited to create an ARG manually, and also using the system. A questionnaire was used for users to compare the two methods, so the data can be captured and analysed.

Six users were invited to conduct user testing. Four were present on the day of testing. Analysis of the questionnaire data from these users showed that:

- 100% of users preferred to use the system in comparison to manual creation overall. More specifically, 100% of users preferred to use the system for the following aspects of game creation:
 - Level of restrictiveness in available choices.
 - Identification and correction of mistakes.
 - Ease of understanding storyline.
 - Ease of game creation.
- 75% of users preferred to use the system in the system with regards to the availability of help.

The analysis of data was performed by directly comparing the results obtained from each user for each individual question. Typical analysis for such data is performed using the Mann-Whitney U-Test; however the level of user turnout is inappropriate in order to generate accurate and reliable results from such a test.

Chapter 5: Conclusion and Project Evaluation

5.1: Introduction

This chapter presents a conclusion to the hypothesis outlined in Chapter One; presented in context of the project aims and objectives. The conclusion will be justified based on the implementation of the system described in Chapter Three, and the subsequent analysis of testing data presented in Chapter Four. An evaluation on this project will be performed, discussing the overall approach taken to this project and the benefits and drawbacks of this approach. Recommendations relating to future work in this area are introduced and briefly discussed.

5.2: Conclusion

5.2.1 Satisfaction of the Aim and Objectives

The aim of the project was to develop an ARG engine in order to test the hypothesis. This was deconstructed in several objectives, which shall be returned to for discussion:

Objective One: Provide components within the system aimed at solving each of the main issues as highlighted in Chapter Two.

Due to time constraints, this objective was partially satisfied. The implemented system provided functionality for:

- Storyline management – Events are scheduled and represented as a series of flow charts. When in-game, the current storyline position is represented on these flow charts. All requirements relating to storyline management are satisfied. This constitutes as a complete solution for storyline management, which is confirmed as 100% of users in the user study agreed that it was easy to grasp the storyline using the system.
- Event management – The system monitors the game state automatically, such that it is able to post new events using an assigned web service, and detect solutions to their associated puzzles. All requirements relating to event management are satisfied, thus this constitutes as a complete solution for event management.
- Character management – The system stores the profiles of characters and permits the conceptual grouping of characters into factions. All requirements relating to character management, with the exception of character interaction handling requirements were satisfied. Thus, the system does not propose a complete solution to this issue.

Objective Two: Develop a plug-in architecture so that new puzzles and web services can be integrated into the engine.

The system uses a common plug-in system to handle web services and puzzles. Implementation of the Plug-in and Separated Interface patterns using Reflection confirms that the system provides a scalable plug-in system. Two web service plug-ins and one puzzle plug-in were developed successfully to demonstrate this in practise. Therefore, this objective was totally satisfied.

Objective Three: Evaluate the system on accessibility, game development efficiency and effort required by inviting users to test the application.

A user study was performed to compare the system with the manual method of ARG development. The analysis in Chapter Four demonstrated positive support for using the system. 100% of responses preferred to the system to manual creation.

However, the results obtained from the user study are not representative of the entire target audience of the system. All of the participants had no experience with creating ARGs, placing their answers, and consequently the overall testing results, firmly from the perspective of beginners. The small turnout of participants meant that using a statistical method for analysing the resultant data could not guarantee the accuracy of any trends it discovered.

Therefore, this objective was partially satisfied. A larger study sample, encompassing a larger demographic matching the target audience would provide more concrete results and satisfy this objective.

5.2.1.1 Aim

The system demonstrates an overall partial completeness of the specified objectives; therefore the aim is partially satisfied.

5.2.2 Overall Conclusion

The hypothesis of this project states that the ease of creating and managing an ARG will increase due to the development of a game engine. The system developed demonstrates that, from the perspective of beginner-level users, there is a significantly positive reaction to developing an ARG using an engine framework due to the following aspects:

- Users felt that the automation of aspects such as storyline and character management helped contextualize the overall narrative, as well as making the development of a game easier.
- Users stated that the system provides help specific to the creation of ARGs, which they identified as difficult to find when creating a game manually.
- The system allowed users to identify mistakes during development, and take the necessary action; thus reducing the risk of any errors affecting game play.

However, this conclusion could be further strengthened by conducting user studies for a larger demographic of the target audience.

5.3: Evaluation

5.3.1 Time Plan

Figure 58 demonstrates the time plan for this project, which has been retrospectively modified to model the actual development of the system. In general, the time plan was complied with in regards to system development. Slack time was used both during the research stage, and delays were present during development. The background research delay was caused due to the underestimation of the time taken to perform three case studies. The implementation overrun was due to the presence of a large amount of unanticipated errors experienced during the development of the plug-in system, and interacting with APIs. Additionally, the author experienced some troubles with initially using the Zend Framework.

Alternate Reality Game Engine

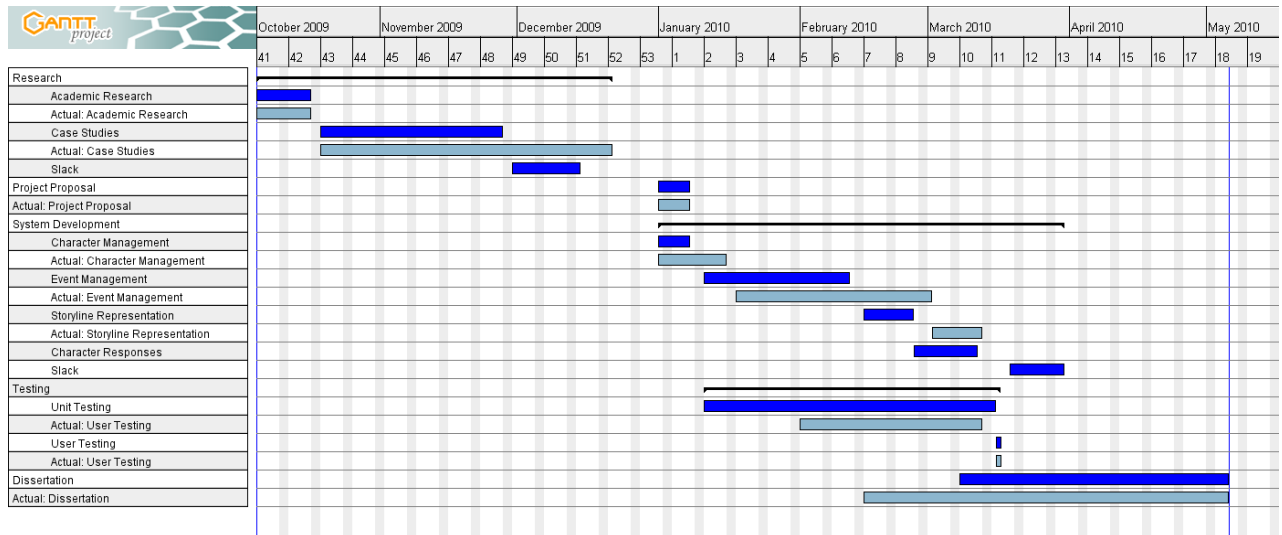


Figure 58: Project Timetable Retrospective Gantt Chart

The modified waterfall method worked well for the implementation of this project. The components of the engine directly related to specific phases, which can be both tested in isolation and when integrated. Allowing earlier phases to be returned to, in contrast to forcing perfect design up-front, ensured that the author felt less constrained and more relaxed during development.

5.3.2 What Has Been Learnt

The author has gained a significant understanding of the PHP programming language and the nature of its deployment on a web server. The Zend Framework has allowed the author to gain a deeper understanding regarding interacting with web services. The author has gained great interest in Alternate Reality Gaming due to the research for this project, and has enjoyed developing a system which could aid the ARG community to develop higher-quality games by removing some of the genre's limitations.

5.3.3 What Went Well

The author believes that the chosen implementation strategy was the best-suited method for developing the system. Using PHP and extensions such as Zend and Smarty provided a large library for use, and an effective method of organizing application and presentation logic. The design of the plug-in system worked well, complementing the implementation strategy by using the advantages of the Plug-in and Separated Interface patterns.

5.3.4 What Could Have Been Done Better

If this project were to be repeated, the author would perform the following actions differently:

- The system does not provide help with regards to designing plug-ins. This risks errors being introduced into plug-ins. The author would therefore develop a plug-in validation tool, a template plug-in and a documented guide to creating a plug-in.
- The in-game event management strictly enforces that only one web service account is scanned for solutions. This could be improved so that multiple linked accounts could be simultaneously scanned; making the system less constraining upon its users.

- The soundness of the conclusion is directly affected by the low turnout rate and the sample demographic for the user study. The context of the conclusion is purely from a beginner's perspective due to this. If the project were to be revisited, the author would conduct a user study more representative of the target audience; perhaps by inviting relevant companies to trial the system.

5.4: Further Work

The system provides an implementation of character management, storyline management and events management. However, due to the limited amount of time to complete the project, the requirements associated with handling character interactions were not implemented. The author proposes that future work in this area takes into consideration the ability to handle character interactions.

This could possibly be achieved using the plug-in architecture used for web services and puzzles. Various responder plug-ins would be associated different style of interaction. For example, a plug-in could be responsible for handling replies to emails using a keyword-detection auto-response mechanism, whereas another plug-in could provide a customized implementation of a real-time natural language processor such as ELIZA for use in instant messaging scenarios.

The research conducted in Chapter Two indicates that it is common practice for storylines in ARGs to remain linear in nature. The system is specifically designed for such events; however this may act as a restriction to more inventive or larger-scale storylines. Further work in the storyline management component could implement the ability to 'branch' a storyline into several threads. However, careful consideration must be placed upon the implications of this action. A suitable solution could be to restrict the number of possible branches, and forcing branches to converge.

5.5: Summary

The following objectives were fully met by the system:

- Develop a plug-in architecture so that new puzzles and web services can be integrated into the engine.

The following objectives were partially met by the system:

- Provide components within the system aimed at solving each of the main issues highlighted in Chapter Two. This was not met due to time constraints.
- Evaluate the system on accessibility, game development efficiency and effort required by inviting users to test the application. This was not met as the demographic of the participants of the user study cannot provide a conclusive answer to the hypothesis.

Thus, the conclusion of this project is that from the perspective of a beginner-level user, the system demonstrates positive support for the hypothesis.

The author has identified two main areas of future work relating to ARG engines that would strengthen the work of this project:

- Incorporating in-game character interaction handling into the engine.

Alternate Reality Game Engine

- Providing functionality for more complex storylines, in terms of parallel storyline branches.
- Perform a larger user-study, incorporating a larger demographic of users.

References

- [1] Salen, K; Zimmerman, E: ***Rules of Play: Game Design Fundamentals*** (Massachusetts: MIT Press, 2003)
- [2] Allen, J. P; Kim, J. Y; Lee, E: ‘**Alternate Reality Gaming**’, *Communications of the ACM*, Vol. 51, No. 2 (February 2008): 36 – 42
- [3] Szulborski, D: ***This Is Not A Game: A Guide to Alternate Reality Gaming*** (Exec Active Media Group, 2005)
- [4] Morrison, A; Turner, J: ‘**Suit Keen Renovator: Alternate Reality Design**’, *Proceedings of the Second Australasian Conference on Interactive Entertainment* (Sydney: ACM, 2005): 209-213
- [5] Montola, M: ‘**Exploring the Edge of the Magic Circle: Defining Pervasive Games**’, *Proceedings of the Digital Arts and Culture Conference* (University of Copenhagen, 2005)
- [6] Miligram, P; Takemura, H; Utsumi, A; Kishino, F: ‘**Augmented Reality: A class of displays of the reality-virtuality continuum**’, *Proceedings of the Telem manipulator and Telepresence Technologies Conference*, Vol 2351 (Kyoto: SPIE, 1994): 282-292
- [7] Dawson, J. D: ***Are Alternate Reality Games more Immersive than Traditional Board Games and Modern Computer Games?*** (Newcastle-Upon-Tyne: University of Newcastle, 2008)
- [8] Hon, A: ‘**How to Make an Alternate Reality Game Or Perplex City: A Look behind the Scenes**’, Youtube (<http://www.youtube.com/watch?v=LtnOUrV8Jb4>, 21 March 2010)
- [9] Mind Candy, ‘**Ascendancy Point Flow Chart**’, *Perplex City Stories* (<http://seasonone.perplexcitystories.com/story/apflowchart.jpg>, 16 April 2010)
- [10] Stewart, S: ‘**The A.I. Web Game**’, *Sean Stewart.org* (<http://www.seanstewart.org/interactive/aiintro/>, 21 March 2010)
- [11] Hi-Res Design, ‘**The Lost Experience**’, *Hi-Res Archives* (<http://archive.hi-res.net/thelostexperience/index.html>, 21 March 2010)
- [12] Epic Games, Inc: ‘**Unreal Technology Overview**’, *Unreal Technology* (<http://www.unrealtechnology.com/features.php?ref=technology-overview> 24 March 2010)
- [13] Whitton, N: ‘**Alternate Reality Games for Orientation, Socialisation and Induction (ARGOSI)**’ (Bristol: Joint Information Systems Committee, 2009)
- [14] Xenophile Media: ‘**Xenophile Media**’, Xenophile Media Technology (<http://www.xenophile.ca/html/technology.html>)
- [15] McGonigal, J: ‘**Immersive Aesthetics and Collective Play**’, *Proceedings of the Digital Arts and Culture Conference* (Melbourne: Royal Melbourne Institute of Technology, 2003)
- [16] Dix, A (ed): ***Human-Computer Interaction (3rd Edition)*** (Harlow: Pearson Education Ltd: 2004)

- [17] The PHP Group: '**PHP: General**', *PHP* (<http://php.net/manual/en/faq.general.php> 2 May 2010)
- [18] Evron, S: '**Zend Framework Components for Non-Framework Development**', *Zend Technologies* (<http://www.slideshare.net/shabar/zend-framework-components-for-nonframework-use> 02 May 2010)
- [19] New Digital Group, Inc: '**Smarty: Why Use It?**' *Smarty Template Engine* (<http://www.smarty.net/whyuse.php> 02 May 2010)
- [20] Siedler, K: '**XAMPP**', Apache Friends (<http://www.apachefriends.org/en/xampp.html> 05 May 2010)
- [21] Fowler, M: *Patterns of Enterprise Application Architecture* (Boston: Pearson Education, Inc, 2003)
- [22] Field, A; Hole, G: *How to Design and Report Experiments* (London: Sage Publications Ltd, 2003)
- [23] Cengage Learning; '**Review of Tests: Ordinal Data**', *Choosing the Correct Statistical Test*, Wadsworth Cengage Learning (http://www.wadsworth.com/psychology_d/templates/student_resources/workshops/stat_workshop/chose_stat/chose_stat_24.html 2005)

Appendix A: Unit Testing Results

Automated Testing

Arcs Table Data Gateway

There are 6 tests.

6 Tests passed:

- get() with valid arc id
- get() with invalid arc id
- getName() with valid arc id
- getName() with invalid arc id
- getAll()
- getAllNames()

0 Tests failed:

Characters Table Data Gateway

There are 6 tests.

6 Tests passed:

- get() with valid character id
- get() with invalid character id
- getName() with valid character id
- getName() with invalid character id
- getAll()
- getAllNames()

0 Tests failed:

Factions Table Data Gateway

There are 3 tests.

3 Tests passed:

- get() with valid faction id
- get() with invalid character id
- getAll()

0 Tests failed:

Factionmembers Table Data Gateway

There are 6 tests.

6 Tests passed:

- getMembers() with valid faction id
- getMembers() with invalid faction id

- getMembers() with no members
- getNonMembers() with valid faction id
- getNonMembers() with invalid faction id
- getNonMembers() with no non-members

0 Tests failed:

Puzzlelist Table Data Gateway

There are 6 tests.

6 Tests passed:

- get() with valid puzzle id
- get() with invalid puzzle id
- getName() with valid puzzle id
- getName() with invalid puzzle id
- getAll()
- getAllNames()

0 Tests failed:

ServiceList Table Data Gateway

There are 6 tests.

6 Tests passed:

- get() with valid service id
- get() with invalid service id
- getName() with valid service id
- getName() with invalid service id
- getAll()
- getAllNames()

0 Tests failed:

Users Table Data Gateway

There are 2 tests.

2 Tests passed:

- get() with valid user id
- get() with invalid user id

0 Tests failed:

Servicesused Table Data Gateway

There are 10 tests.

10 Tests passed:

- findByID() with valid service id
- findByID() with invalid serviceuse id
- findByUser() with valid user id
- findByUser() with invalid user id
- findByService() with valid service id
- findByService() with invalid service id
- findByCharacter() with valid character id
- findByCharacter() with invalid character id
- getName() with valid serviceuse id
- getName() with invalid serviceuse id

0 Tests failed:

Events Table Data Gateway

There are 7 tests.

7 Tests passed:

- get() with valid event id
- get() with invalid event id
- getSummary() with valid event id
- getSummary() with invalid event id
- getAll()
- findByArc() with valid arc id
- findByArc() with invalid arc id

0 Tests failed:

Puzzles Table Data Gateway

There are 5 tests.

5 Tests passed:

- get() with valid puzzleuse id
- get() with invalid puzzleuse id
- getAll()
- findByArc() with valid event id
- getByEvent() with invalid event id

0 Tests failed:

PostedEvents Table Data Gateway

There are 3 tests.

3 Tests passed:

- get() with valid event id
- get() with invalid event id
- getAll()
- Tests failed:

Configuration File

Proxy Settings

There are 5 tests.

5 Tests passed:

- enableProxy()
- disableProxy()
- setProxy() with valid data
- setProxy() with invalid data
- setProxy() with invalid port data

0 Tests failed:

bool(true) bool(false) int(1)

Game Settings

There are 9 tests.

9 Tests passed:

- startGame()
- endGame()
- setEvent() with valid data
- setEvent() with invalid data
- getEvent()
- setPosted(TRUE)
- setPosted(FALSE)
- setPosted(int)
- posted()

0 Tests failed:

Plug-ins

Service Reg Loader Plug-in Factory

There are 15 tests.

15 Tests passed:

- loadService() with valid data - blogger

- printForm() for blogger
- handleForm() for blogger.
- validateData() for blogger with valid data
- validateData() for blogger with invalid data
- loadService() with valid data - email
- printForm() for email
- handleForm() for email with valid data.
- handleForm() for email with invalid data.
- validateData() for email with valid data
- validateData() for email with invalid data
- loadService() with invalid data - twitter
- printForm() when no service has been loaded
- handleForm() when no service has been loaded
- validateData() when no service has been loaded

0 Tests failed:

Service Loader Plug-in Factory

There are 26 tests.

26 Tests passed:

- loadService() with valid data - blogger
- printForm() for blogger
- handleForm() for blogger with valid data
- handleForm() for blogger with invalid data
- validateData() for blogger with valid data
- validateData() for blogger with invalid data
- publish() for blogger with valid data
- publish() for blogger with invalid data
- replies() for blogger with valid data.
- replies() for blogger with invalid data

- loadService() with valid data - email
- printForm() for email
- handleForm() for email with valid data
- handleForm() for email with invalid data
- validateData() for email with valid data
- validateData() for email with invalid data
- publish() for email with valid data
- publish() for email with invalid data
- replies() for email with valid data.
- replies() for email with invalid data
- loadService() with invalid data - twitter
- printForm() when no service has been loaded
- handleForm() when no service has been loaded
- validateData() when no service has been loaded
- publish() when no service has been loaded
- replies() when no service has been loaded

0 Tests failed:

Puzzle Loader Plug-in Factory

There are 20 tests.

20 Tests passed:

- loadPuzzle() with valid data - keyword
- printForm() for keyword
- handleForm() for keyword with valid data.
- handleForm() for keyword with valid data - no keywords.
- handleForm() for keyword with valid data - invalid solution.
- parse() for keyword with valid XML.
- parse() for keyword with invalid XML.
- solution() with submission set to ALL, and data containing correct response.

- solution() with submission set to ALL, and data containing only one response.
- solution() with submission set to ALL, and data containing no responses.
- solution() with submission set to ONE, and data containing correct response.
- solution() with submission set to ONE, and data containing only one response.
- solution() with submission set to ONE, and data containing no responses.
- solution() with incorrect XML data.
- loadPuzzle() with invalid data - twitter
- printForm() when no puzzle has been loaded
- handleForm() when no puzzle has been loaded
- validateData() when no puzzle has been loaded
- solution() when no puzzle has been loaded
- parse() when no puzzle has been loaded

0 Tests failed:

Game Controller

There are 4 tests.

4 Tests passed:

- Post an event with valid content.
- Post an event with invalid content.
- Check for a solution with valid content.
- Check for a solution with invalid content.

0 Tests failed:

Manual Testing

Character Management

This was a **manual** test:

Test: Creating a character with valid data.

2010-05-05T21:03:51+01:00 Priority
Level - INFO

Displaying the create character form.

2010-05-05T21:04:09+01:00 Priority
Level - INFO

Received submit request to create a new character - valid.

Result: Test Passed

Test: Creating a character with invalid data.

2010-05-05T21:04:10+01:00 Priority
Level - INFO

Characters front page reached.

2010-05-05T21:04:11+01:00 Priority
Level - INFO

Displaying the create character form.

2010-05-05T21:04:17+01:00 Priority
Level - ERR

Received submit request to create a character - invalid.
Displaying the create character form.

2010-05-05T21:04:21+01:00 Priority
Level - ERR

Received submit request to create a character - invalid.
Displaying the create character form.

2010-05-05T21:04:26+01:00 Priority
Level - ERR

Received submit request to create a character - invalid.
Displaying the create character form.

Result: Test Passed

Test – Deleting a Character

2010-05-05T21:04:31+01:00 Priority
Level - INFO

Characters front page reached.

2010-05-05T21:04:32+01:00 Priority
Level - INFO

Received request to view a
character.

2010-05-05T21:04:34+01:00 Priority
Level - INFO

Received request to delete
character.

Result: Test Passed

Test: Create a faction with valid data,

2010-05-05T21:04:36+01:00 Priority
Level - INFO

Characters front page reached.

2010-05-05T21:04:38+01:00 Priority
Level - INFO

Reached the factions page..

2010-05-05T21:04:39+01:00 Priority
Level - INFO

Received request to display the
create faction form.

2010-05-05T21:04:43+01:00 Priority
Level - INFO

Received submit request to
create a faction - valid.

Result: Test Passed

**Test – Creating a faction with invalid
information**

2010-05-05T21:04:44+01:00 Priority
Level - INFO

Reached the factions page..

2010-05-05T21:04:45+01:00 Priority
Level - INFO

Received request to display the
create faction form.

2010-05-05T21:04:46+01:00 Priority
Level - ERR

Received submit request to
create a faction - invalid. Display
the create faction form.

Result: Test Passed

Test: Adding a character to a faction

2010-05-05T21:04:48+01:00 Priority
Level - INFO

Reached the factions page.

2010-05-05T21:27:08+01:00 Priority
Level - INFO

Received request to add a
character to faction.

Result: Test Passed

Test: Removing a character from a faction

2010-05-05T21:27:24+01:00 Priority
Level - INFO

Received request to delete a
character from faction.

Result: Test Passed

Test – Delete a Faction

2010-05-05T21:27:27+01:00 Priority
Level - INFO

Reached the factions page.

2010-05-05T21:27:28+01:00 Priority
Level - INFO

Received request to view a
faction.

2010-05-05T21:27:34+01:00 Priority
Level - INFO

Alternate Reality Game Engine

Received request to delete a faction.

Result: Test Passed

Test – Link a valid web service

2010-05-05T21:27:47+01:00 Priority Level - INFO

Characters front page reached.

2010-05-05T21:27:48+01:00 Priority Level - INFO

Received request to view a character.

2010-05-05T21:27:50+01:00 Priority Level - INFO

Received request to go to step 1 of regCharService.

2010-05-05T21:27:50+01:00 Priority Level - INFO

Outputting step one.

2010-05-05T21:27:53+01:00 Priority Level - INFO

Received request to go to step 2 of regCharService.

2010-05-05T21:27:53+01:00 Priority Level - INFO

Step one validation has passed.

2010-05-05T21:27:53+01:00 Priority Level - INFO

Outputting step two.

2010-05-05T21:27:53+01:00 Priority Level - INFO

No further registration needed for Blogger. Nothing to output.

2010-05-05T21:28:05+01:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-05-05T21:28:07+01:00 Priority Level - INFO

Validation with the blogger web service successful.

2010-05-05T21:28:07+01:00 Priority Level - INFO

Step two validation has passed.

2010-05-05T21:28:07+01:00 Priority Level - INFO

Outputting step three.

2010-05-05T21:28:07+01:00 Priority Level - INFO

Registering service - created user.

2010-05-05T21:28:07+01:00 Priority Level - INFO

Registering service - created service.

2010-05-05T21:28:07+01:00 Priority Level - INFO

No further registration needed for Blogger. SUCCESSFUL.

2010-05-05T21:28:07+01:00 Priority Level - INFO

Registering service - COMPLETED.

Result: Test Passed

Test – Link an invalid web service

2010-05-05T21:28:09+01:00 Priority Level - INFO

Received request to view a character.

2010-05-05T21:28:11+01:00 Priority Level - INFO

Received request to go to step 1 of regCharService.

2010-05-05T21:28:11+01:00 Priority Level - INFO

Outputting step one.

2010-05-05T21:28:12+01:00 Priority Level - INFO

Alternate Reality Game Engine

Received request to go to step 2 of regCharService.

2010-05-05T21:28:12+01:00 Priority Level - INFO

Step one validation has passed.

2010-05-05T21:28:12+01:00 Priority Level - INFO

Outputting step two.

2010-05-05T21:28:12+01:00 Priority Level - INFO

No further registration needed for Blogger. Nothing to output.

2010-05-05T21:28:19+01:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-05-05T21:28:19+01:00 Priority Level - ERR

username: is not valid, so step two validation has failed.

2010-05-05T21:28:19+01:00 Priority Level - INFO

Step one validation has passed.

2010-05-05T21:28:19+01:00 Priority Level - INFO

Outputting step two.

2010-05-05T21:28:19+01:00 Priority Level - INFO

No further registration needed for Blogger. Nothing to output.

2010-05-05T21:28:27+01:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-05-05T21:28:27+01:00 Priority Level - ERR

password is not valid, so step two validation has failed.

2010-05-05T21:28:27+01:00 Priority Level - INFO

Step one validation has passed.

2010-05-05T21:28:27+01:00 Priority Level - INFO

Outputting step two.

2010-05-05T21:28:27+01:00 Priority Level - INFO

No further registration needed for Blogger. Nothing to output.

2010-05-05T21:28:38+01:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-05-05T21:28:39+01:00 Priority Level - ERR

Authentication failed with blogger web service - zend_gdata_app_authexception

2010-05-05T21:28:39+01:00 Priority Level - INFO

Step one validation has passed.

2010-05-05T21:28:39+01:00 Priority Level - INFO

Outputting step two.

2010-05-05T21:28:39+01:00 Priority Level - INFO

No further registration needed for Blogger. Nothing to output.

2010-05-05T21:28:48+01:00 Priority Level - INFO

Characters front page reached.

2010-05-05T21:28:49+01:00 Priority Level - INFO

Received request to view a character.

2010-05-05T21:28:52+01:00 Priority Level - INFO

Received request to delete service entry.

Result: Test Passed

Overall Summary:

10 tests passed, 0 tests failed.

No retries needed.

Storyline Management

This was a **manual** test:

Test: Create an arc using valid data

2010-05-05T22:23:26+01:00 Priority
Level - INFO

Reached storyline page.

2010-05-05T22:23:28+01:00 Priority
Level - INFO

Received request to display the
create arc form.

2010-05-05T22:23:35+01:00 Priority
Level - INFO

Received submit request to
create a arc - valid.

Result: Test Passed

Test: Create an arc using invalid data

2010-05-05T22:24:29+01:00 Priority
Level - INFO

Reached storyline page.

2010-05-05T22:24:31+01:00 Priority
Level - INFO

Received request to display the
create arc form.

2010-05-05T22:24:34+01:00 Priority
Level - ERR

Received submit request to
create an arc - invalid. Displaying
the create arc form.

2010-05-05T22:24:40+01:00 Priority
Level - ERR

Received submit request to
create an arc - invalid. Displaying
the create arc form.

Result: Test Passed

Test: Create an event using valid data

2010-05-05T22:25:41+01:00 Priority
Level - INFO

Reached storyline page.

2010-05-05T22:25:42+01:00 Priority
Level - INFO

Received request to view an arc.

2010-05-05T22:25:43+01:00 Priority
Level - INFO

Received request to go to step 1
of regEvent.

2010-05-05T22:25:43+01:00 Priority
Level - INFO

Outputting step one.

2010-05-05T22:25:44+01:00 Priority
Level - INFO

Received request to go to step 2
of regEvent.

2010-05-05T22:25:44+01:00 Priority
Level - INFO

Step one validation passed.

2010-05-05T22:25:44+01:00 Priority
Level - INFO

Outputting step two.

2010-05-05T22:25:45+01:00 Priority
Level - INFO

Received request to go to step 3
of regEvent.

2010-05-05T22:25:45+01:00 Priority
Level - INFO

Step two validation has passed.

Alternate Reality Game Engine

2010-05-05T22:25:45+01:00 Priority
Level - INFO

Outputting step three.

2010-05-05T22:25:57+01:00 Priority
Level - INFO

Received request to go to step 4
of regEvent.

2010-05-05T22:25:57+01:00 Priority
Level - INFO

Validation for the Blogger event
successful.

2010-05-05T22:25:57+01:00 Priority
Level - ERR

No Keywords have been specified
by the user.

Result: Test Failed

Test: Create an event using invalid data.

2010-05-05T22:27:07+01:00 Priority
Level - INFO

Reached storyline page.

2010-05-05T22:27:08+01:00 Priority
Level - INFO

Received request to view an arc.

2010-05-05T22:27:09+01:00 Priority
Level - INFO

Received request to go to step 1
of regEvent.

2010-05-05T22:27:09+01:00 Priority
Level - INFO

Outputting step one.

2010-05-05T22:27:10+01:00 Priority
Level - INFO

Received request to go to step 2
of regEvent.

2010-05-05T22:27:10+01:00 Priority
Level - INFO

Step one validation passed.

2010-05-05T22:27:10+01:00 Priority
Level - INFO

Outputting step two.

2010-05-05T22:27:12+01:00 Priority
Level - INFO

Received request to go to step 3
of regEvent.

2010-05-05T22:27:12+01:00 Priority
Level - INFO

Step two validation has passed.

2010-05-05T22:27:12+01:00 Priority
Level - INFO

Outputting step three.

2010-05-05T22:27:14+01:00 Priority
Level - INFO

Received request to go to step 4
of regEvent.

2010-05-05T22:27:14+01:00 Priority
Level - ERR

No event summary has been
supplied, so step three validation
has failed.

2010-05-05T22:27:14+01:00 Priority
Level - INFO

Step two validation has passed.

2010-05-05T22:27:14+01:00 Priority
Level - INFO

Outputting step three.

2010-05-05T22:27:28+01:00 Priority
Level - INFO

Received request to go to step 4
of regEvent.

2010-05-05T22:27:28+01:00 Priority
Level - ERR

No Title has been specified by
the user.

Alternate Reality Game Engine

2010-05-05T22:27:28+01:00 Priority
Level - INFO

Step two validation has passed.

2010-05-05T22:27:28+01:00 Priority
Level - INFO

Outputting step three.

2010-05-05T22:27:41+01:00 Priority
Level - INFO

Received request to go to step 4
of regEvent.

2010-05-05T22:27:42+01:00 Priority
Level - INFO

Validation for the Blogger event
successful.

2010-05-05T22:27:42+01:00 Priority
Level - ERR

No Keywords have been specified
by the user.

2010-05-05T22:27:42+01:00 Priority
Level - INFO

Step two validation has passed.

2010-05-05T22:27:42+01:00 Priority
Level - INFO

Outputting step three.

Result: Test Passed

Test: Delete an event

2010-05-05T22:30:10+01:00 Priority
Level - INFO

Reached storyline page.

2010-05-05T22:30:22+01:00 Priority
Level - INFO

Received request to view an arc.

2010-05-05T22:30:24+01:00 Priority
Level - INFO

Reached the view event page.

2010-05-05T22:30:26+01:00 Priority
Level - INFO

Received request to delete
event.

Result: Test Passed

Test: Delete an arc

2010-05-05T22:30:29+01:00 Priority
Level - INFO

Reached storyline page.

2010-05-05T22:30:30+01:00 Priority
Level - INFO

Received request to view an arc.

2010-05-05T22:30:31+01:00 Priority
Level - INFO

Received request to delete arc.

Result: Test Passed

**Test: Create an event using valid data
[RETRY]**

2010-05-05T22:31:10+01:00 Priority
Level - INFO

Reached storyline page.

2010-05-05T22:31:33+01:00 Priority
Level - INFO

Received request to view an arc.

2010-05-05T22:31:59+01:00 Priority
Level - INFO

Received request to go to step 1
of regEvent.

2010-05-05T22:31:59+01:00 Priority
Level - INFO

Outputting step one.

2010-05-05T22:32:00+01:00 Priority
Level - INFO

Received request to go to step 2
of regEvent.

2010-05-05T22:32:00+01:00 Priority
Level - INFO

Alternate Reality Game Engine

Step one validation passed.

2010-05-05T22:32:00+01:00 Priority
Level - INFO

Outputting step two.

2010-05-05T22:32:01+01:00 Priority
Level - INFO

Received request to go to step 3
of regEvent.

2010-05-05T22:32:01+01:00 Priority
Level - INFO

Step two validation has passed.

2010-05-05T22:32:01+01:00 Priority
Level - INFO

Outputting step three.

2010-05-05T22:32:12+01:00 Priority
Level - INFO

Received request to go to step 4
of regEvent.

2010-05-05T22:32:12+01:00 Priority
Level - INFO

Validation for the Blogger event
successful.

2010-05-05T22:32:13+01:00 Priority
Level - INFO

Keyword Puzzle validation
successful.

2010-05-05T22:32:13+01:00 Priority
Level - INFO

Outputting step four.

2010-05-05T22:32:13+01:00 Priority
Level - INFO

Validation for the Blogger event
successful.

2010-05-05T22:32:13+01:00 Priority
Level - INFO

Registering event: Registered
event data.

2010-05-05T22:32:13+01:00 Priority
Level - INFO

Keyword Puzzle validation
successful.

2010-05-05T22:32:13+01:00 Priority
Level - INFO

Registering event: Registered
puzzle data.

2010-05-05T22:32:13+01:00 Priority
Level - INFO

Event successfully registered -
COMPLETE.

Result: Test Passed

Overall Summary:

5 tests passed, 1 test failed.

1 test passed on retry

Appendix B: User Testing Script

“This test requires you to create an Alternate Reality Game using two different methods. Alternate Reality Games are web-based games. They involve creating fictional characters, and assigning them web services. A story, of your own choosing, is split into distinct events. Events consist of a piece of storyline and a puzzle relating to that content. The events are posted using a web service account owned by a character. Players will reply to these publications with proposed solutions to the puzzles you set. When a solution is detected, the next event in the timeline is published.

This testing involves you creating an Alternate Reality Game manually, in which you will set up all appropriate web services and plot out the storyline. Also, you will create the same game using the Alternate Reality Game Engine. A short questionnaire will follow.”

Appendix C: Questionnaire

About Alternate Reality Gaming

- Have you ever played an Alternate Reality Game?

YES ☐ ☐ NO

- Have you ever created an Alternate Reality Game?

YES ☐ ☐ NO

Creating Alternate Reality Games

For each of the statements below, please indicate your level of agreement.

- “I felt unrestricted in terms of the choices I could make when creating the game (for example – choices regarding web services and puzzles).
- Creating by hand: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Using the ARG Engine: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Which method do you prefer in this case?
- CREATING BY HAND ☐ ☐ ARG ENGINE
- Why have you made that choice?
-
-
-
- It was easy to find help when I needed to look for help.
- Creating by hand: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Using the ARG Engine: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Which method do you prefer in this case?
- CREATING BY HAND ☐ ☐ ARG ENGINE
- Why have you made that choice?
-
-
-
- It was easy to identify when I had made a mistake and take corrective action.
- Using the ARG Engine: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Creating by hand: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Which method do you prefer in this case?
- CREATING BY HAND ☐ ☐ ARG ENGINE
- Why have you made that choice?
-
-
-
- It was easy to grasp the story timeline.
- Using the ARG Engine: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Creating by hand: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree

- Which method do you prefer in this case?
- CREATING BY HAND ☐ ☐ ARG ENGINE
- Why have you made that choice?
-
-
-
- It was easy to create an Alternate Reality Game.
- Creating by hand: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Using the ARG Engine: Strongly Agree ☐ ☐ ☐ ☐ Strongly Disagree
- Which method do you prefer in this case?
- CREATING BY HAND ☐ ☐ ARG ENGINE
- Why have you made that choice?
-
-
-

Comparison

- If you were to create an Alternate Reality Game in future, which method would you use?
- CREATING BY HAND ☐ ☐ ARG ENGINE
- Why have you made that choice?

--

Appendix D: Questionnaire Results

Question One

“Have you ever played an Alternate Reality Game?”

Test Case	Yes	No
1		1
2		1
3		1
4		1
Totals	0	4
Percentages	0%	100%

Question Two

“Have you ever created an Alternate Reality Game?”

Test Case	Yes	No
1		1
2		1
3		1
4		1
Totals	0	4
Percentages	0%	100%

Question Three

“I felt unrestricted in terms of the choices I could make when creating the game”

Part A – Creating by hand:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1				1
2				1
3				1
4		1		
Totals	1	0	0	3
Percentages	25%	0%	0%	75%

Part B – Using the ARG Engine:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1			1	
2			1	
3				1
4			1	
Totals	0	0	3	1
Percentages	0%	0%	75%	25%

Part C – Which method do you prefer in this case?:

Test Case	Creating By Hand	ARG Engine
1		1
2		1
3		1
4		1
Totals	0	4
Percentages	0%	100%

Part D – Why have you made that choice?

Test Case	Answer
1	“It was easier to schedule events and organise characters using the engine”
2	“ARG engine is quicker”
3	“Much easier to track events (Flowchart view)”
4	“It made everything easier by linking it together”

Question Four

“It was easy to find help when I needed to look for help”

Part A – Creating by hand:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1		1		
2		1		
3	1			
4			1	
Totals	1	2	1	0
Percentages	25%	50%	25%	0%

Part B – Using the ARG Engine:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1				1
2			1	
3		1		
4			1	
Totals	0	1	2	1
Percentages	0%	25%	50%	25%

Part C – Which method do you prefer in this case?:

Test Case	Creating By Hand	ARG Engine
1		1
2		1
3		1
4	1	
Totals	0	3
Percentages	25%	75%

Part D – Why have you made that choice?

Test Case	Answer
1	“Explanations were available on each page of the ARG engine”
2	“Completely on your own when done by hand. ARG engine had instructions”
3	“No help available for manual method”
4	“Could use each services help system”

Question Five

“It was easy to identify when I had made a mistake, and take corrective action”

Part A – Using the ARG Engine:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1				1
2				1
3			1	
4		1		
Totals	0	1	1	2
Percentages	0%	25%	25%	50%

Part B – Creating by Hand:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1		1		
2		1		
3		1		
4		1		
Totals	0	0	0	0
Percentages	0%	100%	0%	0%

Part C – Which method do you prefer in this case?:

Test Case	Creating By Hand	ARG Engine
1		1
2		1
3		1
4		1
Totals	0	4
Percentages	0%	100%

Part D – Why have you made that choice?

Test Case	Answer
1	“Error messages were displayed if I entered incorrect information”
2	“Can easily see where you’re going wrong with ARG engine. By hand you’d have to manually keep track”
3	“ARG engine allows user to build entire game before firing of events”
4	“Tells you about what you have done”

Question Six

“It was easy to grasp the story timeline”

Part A – Using the ARG Engine:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1				1
2				1
3				1
4				1
Totals	0	0	0	4
Percentages	0%	0%	0%	100%

Part B – Creating by Hand:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1	1			
2				1
3		1		
4	1			
Totals	2	1	1	0
Percentages	50%	25%	25%	0%

Part C – Which method do you prefer in this case?:

Test Case	Creating By Hand	ARG Engine
1		1
2		1
3		1
4		1
Totals	0	4
Percentages	0%	100%

Part D – Why have you made that choice?

Test Case	Answer
1	“The ARG engine clearly showed the storyline timeline on screen”
2	“Flowcharts in ARG = good”
3	“Flowchart view simple and obvious”
4	“Easier to read diagrams showing path of game”

Question Seven

“It was easy to create an Alternate Reality Game”

Part A – Creating by hand:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1		1		
2				1
3	1			
4			1	
Totals	1	1	2	0
Percentages	25%	25%	50%	0%

Part B – Creating by Hand:

Test Case	Strongly Disagree	Disagree	Agree	Strongly Agree
1			1	
2				1
3			1	
4			1	
Totals	0	0	3	1
Percentages	0%	0%	75%	25%

Part C – Which method do you prefer in this case?:

Test Case	Creating By Hand	ARG Engine
1		1
2		1
3		1
4		1
Totals	0	4
Percentages	0%	100%

Part D – Why have you made that choice?

Test Case	Answer
1	“It was easier to schedule and organise events”
2	“Speed and automation”
3	“Managing events across multiple web services difficult. Need to be present and manually filter responses. ARG Engine automates all this”
4	“Most work is done for you”

Question Eight

“If you were to create an Alternate Reality Game in future, what method would you choose?”

Part A – Select a system:

Test Case	Creating By Hand	ARG Engine
1		1
2		1
3		1
4		1
Totals	0	4
Percentages	0%	100%

Part B – Why have you made that choice?

Test Case	Answer
1	"I prefer the ARG Engines method of creating and organising events"
2	"Takes significantly less effort, no need to manually track the story yourself"
3	"ARG Engine allows centralised management of all web services and puzzles. Simple event flow chart to manage events. Automatic response checking"
4	"All services are controlled by system"

Appendix E: User Testing Logs

Test Cases

Test Case One

2010-03-16T09:58:18+00:00 Priority Level - INFO

Index Page reached.

2010-03-16T10:01:03+00:00 Priority Level - INFO

Reached game page.

2010-03-16T10:01:08+00:00 Priority Level - INFO

Index Page reached.

2010-03-16T10:01:26+00:00 Priority Level - INFO

Index Page reached.

2010-03-16T10:02:22+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T10:02:22+00:00 Priority Level - ERR

No characters exist.

2010-03-16T10:29:20+00:00 Priority Level - INFO

Index Page reached.

2010-03-16T10:29:31+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T10:29:31+00:00 Priority Level - ERR

No characters exist.

2010-03-16T10:29:32+00:00 Priority Level - INFO

Displaying the create character form.

2010-03-16T10:31:07+00:00 Priority Level - INFO

Received submit request to create a new character - valid.

2010-03-16T10:31:09+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T10:31:11+00:00 Priority Level - INFO

Displaying the create character form.

2010-03-16T10:31:32+00:00 Priority Level - INFO

Received submit request to create a new character - valid.

2010-03-16T10:31:34+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T10:31:47+00:00 Priority Level - INFO

Received request to view a character.

2010-03-16T10:31:51+00:00 Priority Level - INFO

Received request to go to step 1 of regCharService.

2010-03-16T10:31:51+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T10:31:55+00:00 Priority Level - INFO

Received request to go to step 2 of regCharService.

2010-03-16T10:31:55+00:00 Priority Level - INFO

Step one validation has passed.

2010-03-16T10:31:55+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T10:31:55+00:00 Priority Level - INFO

Returning the address of the template file - emailReg.tpl.

2010-03-16T10:32:03+00:00 Priority Level - INFO

Received request to view a character.

2010-03-16T10:32:05+00:00 Priority Level - INFO

Received request to go to step 1 of regCharService.

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T10:32:05+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T10:32:08+00:00 Priority Level - INFO

Received request to go to step 2 of regCharService.

2010-03-16T10:32:08+00:00 Priority Level - INFO

Step one validation has passed.

2010-03-16T10:32:08+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T10:32:08+00:00 Priority Level - INFO

Returning the address of the template file - emailReg.tpl.

2010-03-16T10:32:40+00:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-03-16T10:32:40+00:00 Priority Level - INFO

Validation with the email web service successful.

2010-03-16T10:32:40+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T10:32:40+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T10:32:40+00:00 Priority Level - INFO

Registering service - created user.

2010-03-16T10:32:40+00:00 Priority Level - INFO

Registering service - created service.

2010-03-16T10:32:40+00:00 Priority Level - INFO

Successfully registered email details with the database.

2010-03-16T10:32:40+00:00 Priority Level - INFO

Registering service - COMPLETED.

2010-03-16T10:32:44+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T10:32:45+00:00 Priority Level - INFO

Received request to view a character.

2010-03-16T10:32:49+00:00 Priority Level - INFO

Received request to go to step 1 of regCharService.

2010-03-16T10:32:49+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T10:32:51+00:00 Priority Level - INFO

Received request to go to step 2 of regCharService.

2010-03-16T10:32:51+00:00 Priority Level - INFO

Step one validation has passed.

2010-03-16T10:32:51+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T10:32:51+00:00 Priority Level - INFO

No further registration needed for Blogger. Nothing to output.

2010-03-16T10:35:30+00:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-03-16T10:35:31+00:00 Priority Level - INFO

Validation with the blogger web service successful.

2010-03-16T10:35:31+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T10:35:31+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T10:35:31+00:00 Priority Level - INFO

Registering service - created user.

2010-03-16T10:35:31+00:00 Priority Level - INFO

Registering service - created service.

2010-03-16T10:35:31+00:00 Priority Level - INFO

Developing a Game Engine for the Creation and Management of Alternate Reality Games

No further registration needed for Blogger. SUCCESSFUL.

2010-03-16T10:35:31+00:00 Priority Level - INFO

Registering service - COMPLETED.

2010-03-16T10:35:35+00:00 Priority Level - INFO

Reached services page.

2010-03-16T10:35:40+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T10:35:40+00:00 Priority Level - INFO

No arcs exist in database.

2010-03-16T10:35:44+00:00 Priority Level - INFO

Received request to display the create arc form.

2010-03-16T10:35:59+00:00 Priority Level - INFO

Received submit request to create a arc - valid.

2010-03-16T10:36:02+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T10:36:03+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T10:36:07+00:00 Priority Level - INFO

Received request to go to step 1 of regEvent.

2010-03-16T10:36:07+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T10:36:11+00:00 Priority Level - INFO

Received request to go to step 2 of regEvent.

2010-03-16T10:36:11+00:00 Priority Level - INFO

Step one validation passed.

2010-03-16T10:36:11+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T10:36:15+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T10:36:15+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T10:36:15+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T10:38:20+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T10:38:21+00:00 Priority Level - INFO

Validation for the Email event successful.

2010-03-16T10:38:21+00:00 Priority Level - INFO

Outputting step four.

2010-03-16T10:38:21+00:00 Priority Level - INFO

Registering event: Registered event data.

2010-03-16T10:38:21+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T10:38:21+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T10:38:29+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T10:38:33+00:00 Priority Level - INFO

Received request to go to step 1 of regEvent.

2010-03-16T10:38:33+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T10:38:38+00:00 Priority Level - INFO

Received request to go to step 2 of regEvent.

2010-03-16T10:38:38+00:00 Priority Level - INFO

Step one validation passed.

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T10:38:38+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T10:38:41+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T10:38:41+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T10:38:41+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T10:40:15+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T10:40:15+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T10:40:15+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T10:41:16+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T10:41:17+00:00 Priority Level - ERR

No BlogID has been specified by the user.

2010-03-16T10:41:17+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T10:41:17+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T10:42:48+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T10:42:49+00:00 Priority Level - ERR

No BlogID has been specified by the user.

2010-03-16T10:42:49+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T10:42:49+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T10:43:16+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T10:43:17+00:00 Priority Level - ERR

No BlogID has been specified by the user.

2010-03-16T10:43:17+00:00 Priority Level - INFO

Outputting step four.

2010-03-16T10:43:17+00:00 Priority Level - INFO

Registering event: Registered event data.

2010-03-16T10:43:17+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T10:43:17+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T10:43:21+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T10:43:27+00:00 Priority Level - INFO

Received request to go to step 1 of regEvent.

2010-03-16T10:43:27+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T10:43:31+00:00 Priority Level - INFO

Received request to go to step 2 of regEvent.

2010-03-16T10:43:31+00:00 Priority Level - INFO

Step one validation passed.

2010-03-16T10:43:31+00:00 Priority Level - INFO

Outputting step two.

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T10:43:33+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T10:43:33+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T10:43:33+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T10:44:10+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T10:44:10+00:00 Priority Level - INFO

Validation for the Email event successful.

2010-03-16T10:44:10+00:00 Priority Level - INFO

Outputting step four.

2010-03-16T10:44:10+00:00 Priority Level - INFO

Registering event: Registered event data.

2010-03-16T10:44:10+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T10:44:10+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T10:44:13+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T10:44:17+00:00 Priority Level - INFO

Reached game page.

2010-03-16T10:44:19+00:00 Priority Level - INFO

Reached game page.

2010-03-16T10:44:19+00:00 Priority Level - INFO

--GAME STARTED Pressed by user.--

2010-03-16T10:44:39+00:00 Priority Level - INFO

Index Page reached.

Test Case Two

2010-03-16T10:58:44+00:00 Priority Level - INFO

Index Page reached.

2010-03-16T11:27:16+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T11:27:16+00:00 Priority Level - ERR

No characters exist.

2010-03-16T11:27:18+00:00 Priority Level - INFO

Displaying the create character form.

2010-03-16T11:27:28+00:00 Priority Level - ERR

Received submit request to create a character - invalid. Displaying the create character form.

2010-03-16T11:27:38+00:00 Priority Level - INFO

Received submit request to create a new character - valid.

2010-03-16T11:27:41+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T11:27:44+00:00 Priority Level - INFO

Reached the factions page..

2010-03-16T11:27:44+00:00 Priority Level - INFO

No factions exist in the database.

2010-03-16T11:27:49+00:00 Priority Level - INFO

Received request to display the create faction form.

2010-03-16T11:27:59+00:00 Priority Level - INFO

Received submit request to create a faction - valid.

2010-03-16T11:28:01+00:00 Priority Level - INFO

Reached the factions page..

2010-03-16T11:28:04+00:00 Priority Level - INFO

Developing a Game Engine for the Creation and Management of Alternate Reality Games

Received request to view a faction.	2010-03-16T11:29:12+00:00 Priority Level - INFO
2010-03-16T11:28:06+00:00 Priority Level - INFO	Outputting step two.
Received request to add a character to faction.	2010-03-16T11:29:12+00:00 Priority Level - INFO
2010-03-16T11:28:09+00:00 Priority Level - INFO	No further registration needed for Blogger. Nothing to output.
Reached services page.	2010-03-16T11:29:42+00:00 Priority Level - INFO
2010-03-16T11:28:28+00:00 Priority Level - INFO	Received request to go to step 3 of regCharService.
Characters front page reached.	2010-03-16T11:29:43+00:00 Priority Level - ERR
2010-03-16T11:28:30+00:00 Priority Level - INFO	Authentication failed with blogger web service - zend_gdata_app_authexception
Received request to view a character.	2010-03-16T11:29:43+00:00 Priority Level - INFO
2010-03-16T11:28:33+00:00 Priority Level - INFO	Step one validation has passed.
Received request to go to step 1 of regCharService.	2010-03-16T11:29:43+00:00 Priority Level - INFO
2010-03-16T11:28:33+00:00 Priority Level - INFO	Outputting step two.
Outputting step one.	2010-03-16T11:29:43+00:00 Priority Level - INFO
2010-03-16T11:28:37+00:00 Priority Level - INFO	No further registration needed for Blogger. Nothing to output.
Received request to go to step 2 of regCharService.	2010-03-16T11:30:44+00:00 Priority Level - INFO
2010-03-16T11:28:37+00:00 Priority Level - INFO	Received request to go to step 3 of regCharService.
Step one validation has passed.	2010-03-16T11:30:48+00:00 Priority Level - ERR
2010-03-16T11:28:37+00:00 Priority Level - INFO	Authentication failed with blogger web service - zend_gdata_app_authexception
Outputting step two.	2010-03-16T11:30:48+00:00 Priority Level - INFO
2010-03-16T11:28:37+00:00 Priority Level - INFO	Step one validation has passed.
No further registration needed for Blogger. Nothing to output.	2010-03-16T11:30:48+00:00 Priority Level - INFO
2010-03-16T11:29:11+00:00 Priority Level - INFO	Outputting step two.
Received request to go to step 3 of regCharService.	2010-03-16T11:30:48+00:00 Priority Level - INFO
2010-03-16T11:29:12+00:00 Priority Level - ERR	No further registration needed for Blogger. Nothing to output.
Authentication failed with blogger web service - zend_gdata_app_authexception	2010-03-16T11:31:17+00:00 Priority Level - INFO
2010-03-16T11:29:12+00:00 Priority Level - INFO	Received request to go to step 3 of regCharService.
Step one validation has passed.	

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T11:31:18+00:00 Priority Level - INFO

Validation with the blogger web service successful.

2010-03-16T11:31:18+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T11:31:18+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T11:31:18+00:00 Priority Level - INFO

Registering service - created user.

2010-03-16T11:31:18+00:00 Priority Level - INFO

Registering service - created service.

2010-03-16T11:31:18+00:00 Priority Level - INFO

No further registration needed for Blogger. SUCCESSFUL.

2010-03-16T11:31:18+00:00 Priority Level - INFO

Registering service - COMPLETED.

2010-03-16T11:31:26+00:00 Priority Level - INFO

Received request to view a character.

2010-03-16T11:31:28+00:00 Priority Level - INFO

Received request to go to step 1 of regCharService.

2010-03-16T11:31:28+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T11:31:30+00:00 Priority Level - INFO

Received request to go to step 2 of regCharService.

2010-03-16T11:31:30+00:00 Priority Level - INFO

Step one validation has passed.

2010-03-16T11:31:30+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T11:31:30+00:00 Priority Level - INFO

Returning the address of the template file - emailReg.tpl.

2010-03-16T11:32:20+00:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-03-16T11:32:20+00:00 Priority Level - INFO

Validation with the email web service successful.

2010-03-16T11:32:20+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T11:32:20+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T11:32:20+00:00 Priority Level - INFO

Registering service - created user.

2010-03-16T11:32:20+00:00 Priority Level - INFO

Registering service - created service.

2010-03-16T11:32:21+00:00 Priority Level - INFO

Successfully registered email details with the database.

2010-03-16T11:32:21+00:00 Priority Level - INFO

Registering service - COMPLETED.

2010-03-16T11:32:26+00:00 Priority Level - INFO

Reached services page.

2010-03-16T11:32:29+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T11:32:29+00:00 Priority Level - INFO

No arcs exist in database.

2010-03-16T11:32:39+00:00 Priority Level - INFO

Received request to display the create arc form.

2010-03-16T11:32:54+00:00 Priority Level - INFO

Received submit request to create a arc - valid.

2010-03-16T11:32:56+00:00 Priority Level - INFO

Reached storyline page.

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T11:33:02+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T11:33:05+00:00 Priority Level - INFO

Received request to go to step 1 of regEvent.

2010-03-16T11:33:05+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T11:33:10+00:00 Priority Level - INFO

Received request to go to step 2 of regEvent.

2010-03-16T11:33:10+00:00 Priority Level - INFO

Step one validation passed.

2010-03-16T11:33:10+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T11:33:14+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T11:33:14+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T11:33:14+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T11:34:22+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T11:34:22+00:00 Priority Level - INFO

Validation for the Email event successful.

2010-03-16T11:34:22+00:00 Priority Level - INFO

Outputting step four.

2010-03-16T11:34:22+00:00 Priority Level - INFO

Registering event: Registered event data.

2010-03-16T11:34:22+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T11:34:22+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T11:34:29+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T11:34:35+00:00 Priority Level - INFO

Received request to go to step 1 of regEvent.

2010-03-16T11:34:35+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T11:34:44+00:00 Priority Level - INFO

Received request to go to step 2 of regEvent.

2010-03-16T11:34:44+00:00 Priority Level - INFO

Step one validation passed.

2010-03-16T11:34:44+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T11:34:49+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T11:34:49+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T11:34:49+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T11:36:42+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T11:36:43+00:00 Priority Level - INFO

Validation for the Blogger event successful.

2010-03-16T11:36:43+00:00 Priority Level - INFO

Outputting step four.

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T11:36:44+00:00 Priority Level - INFO

Registering event: Registered event data.

2010-03-16T11:36:44+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T11:36:44+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T11:36:49+00:00 Priority Level - INFO

Reached the factions page..

2010-03-16T11:36:52+00:00 Priority Level - INFO

Reached services page.

2010-03-16T11:36:53+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T11:36:56+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T11:37:01+00:00 Priority Level - INFO

Received request to go to step 1 of regEvent.

2010-03-16T11:37:01+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T11:37:04+00:00 Priority Level - INFO

Received request to go to step 2 of regEvent.

2010-03-16T11:37:04+00:00 Priority Level - INFO

Step one validation passed.

2010-03-16T11:37:04+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T11:37:16+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T11:37:16+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T11:37:16+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T11:38:24+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T11:38:24+00:00 Priority Level - INFO

Validation for the Email event successful.

2010-03-16T11:38:24+00:00 Priority Level - INFO

Outputting step four.

2010-03-16T11:38:24+00:00 Priority Level - INFO

Registering event: Registered event data.

2010-03-16T11:38:24+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T11:38:24+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T11:38:30+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T11:38:36+00:00 Priority Level - INFO

Reached game page.

2010-03-16T11:38:40+00:00 Priority Level - INFO

Reached game page.

2010-03-16T11:38:40+00:00 Priority Level - INFO

--GAME STARTED Pressed by user.--

Test Case Three

2010-03-16T11:41:47+00:00 Priority Level - INFO

Index Page reached.

2010-03-16T12:08:53+00:00 Priority Level - INFO

Developing a Game Engine for the Creation and Management of Alternate Reality Games

Characters front page reached.	2010-03-16T12:09:48+00:00 Priority Level - INFO
2010-03-16T12:08:53+00:00 Priority Level - ERR	Outputting step one.
No characters exist.	2010-03-16T12:10:03+00:00 Priority Level - INFO
2010-03-16T12:08:55+00:00 Priority Level - INFO	Received request to view a character.
Displaying the create character form.	2010-03-16T12:10:05+00:00 Priority Level - INFO
2010-03-16T12:09:16+00:00 Priority Level - INFO	Received request to go to step 1 of regCharService.
Received submit request to create a new character - valid.	2010-03-16T12:10:05+00:00 Priority Level - INFO
2010-03-16T12:09:18+00:00 Priority Level - INFO	Outputting step one.
Characters front page reached.	2010-03-16T12:10:08+00:00 Priority Level - INFO
2010-03-16T12:09:22+00:00 Priority Level - INFO	Received request to go to step 2 of regCharService.
Reached the factions page..	2010-03-16T12:10:08+00:00 Priority Level - INFO
2010-03-16T12:09:22+00:00 Priority Level - INFO	Step one validation has passed.
No factions exist in the database.	2010-03-16T12:10:08+00:00 Priority Level - INFO
2010-03-16T12:09:28+00:00 Priority Level - INFO	Outputting step two.
Reached services page.	2010-03-16T12:10:08+00:00 Priority Level - INFO
2010-03-16T12:09:36+00:00 Priority Level - INFO	Returning the address of the template file - emailReg.tpl.
Characters front page reached.	2010-03-16T12:10:47+00:00 Priority Level - INFO
2010-03-16T12:09:38+00:00 Priority Level - INFO	Received request to go to step 3 of regCharService.
Reached the factions page..	2010-03-16T12:10:47+00:00 Priority Level - INFO
2010-03-16T12:09:38+00:00 Priority Level - INFO	Validation with the email web service successful.
No factions exist in the database.	2010-03-16T12:10:47+00:00 Priority Level - INFO
2010-03-16T12:09:39+00:00 Priority Level - INFO	Step two validation has passed.
Reached services page.	2010-03-16T12:10:47+00:00 Priority Level - INFO
2010-03-16T12:09:44+00:00 Priority Level - INFO	Outputting step three.
Characters front page reached.	2010-03-16T12:10:47+00:00 Priority Level - INFO
2010-03-16T12:09:46+00:00 Priority Level - INFO	Registering service - created user.
Received request to view a character.	2010-03-16T12:10:48+00:00 Priority Level - INFO
2010-03-16T12:09:48+00:00 Priority Level - INFO	Registering service - created service.
Received request to go to step 1 of regCharService.	

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T12:10:48+00:00 Priority Level - INFO

Successfully registered email details with the database.

2010-03-16T12:10:48+00:00 Priority Level - INFO

Registering service - COMPLETED.

2010-03-16T12:11:01+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T12:11:03+00:00 Priority Level - INFO

Received request to view a character.

2010-03-16T12:11:04+00:00 Priority Level - INFO

Received request to go to step 1 of regCharService.

2010-03-16T12:11:04+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T12:11:12+00:00 Priority Level - INFO

Reached services page.

2010-03-16T12:11:14+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T12:11:14+00:00 Priority Level - INFO

No arcs exist in database.

2010-03-16T12:11:16+00:00 Priority Level - INFO

Received request to display the create arc form.

2010-03-16T12:11:30+00:00 Priority Level - INFO

Received submit request to create a arc - valid.

2010-03-16T12:11:33+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T12:11:37+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T12:11:42+00:00 Priority Level - INFO

Received request to go to step 1 of regEvent.

2010-03-16T12:11:42+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T12:11:44+00:00 Priority Level - INFO

Received request to go to step 2 of regEvent.

2010-03-16T12:11:44+00:00 Priority Level - INFO

Step one validation passed.

2010-03-16T12:11:44+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T12:11:50+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T12:11:50+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T12:11:50+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T12:12:54+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T12:12:54+00:00 Priority Level - INFO

Validation for the Email event successful.

2010-03-16T12:12:54+00:00 Priority Level - INFO

Outputting step four.

2010-03-16T12:12:54+00:00 Priority Level - INFO

Registering event: Registered event data.

2010-03-16T12:12:54+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T12:12:54+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T12:13:00+00:00 Priority Level - INFO

Developing a Game Engine for the Creation and Management of Alternate Reality Games

Received request to view an arc.	2010-03-16T12:14:23+00:00 Priority Level - INFO
2010-03-16T12:13:04+00:00 Priority Level - INFO	Event successfully registered - COMPLETE.
Received request to go to step 1 of regEvent.	2010-03-16T12:14:26+00:00 Priority Level - INFO
2010-03-16T12:13:04+00:00 Priority Level - INFO	Received request to view an arc.
Outputting step one.	2010-03-16T12:14:28+00:00 Priority Level - INFO
2010-03-16T12:13:07+00:00 Priority Level - INFO	Received request to go to step 1 of regEvent.
Received request to go to step 2 of regEvent.	2010-03-16T12:14:28+00:00 Priority Level - INFO
2010-03-16T12:13:07+00:00 Priority Level - INFO	Outputting step one.
Step one validation passed.	2010-03-16T12:14:46+00:00 Priority Level - INFO
2010-03-16T12:13:07+00:00 Priority Level - INFO	Received request to go to step 2 of regEvent.
Outputting step two.	2010-03-16T12:14:46+00:00 Priority Level - INFO
2010-03-16T12:13:09+00:00 Priority Level - INFO	Step one validation passed.
Received request to go to step 3 of regEvent.	2010-03-16T12:14:46+00:00 Priority Level - INFO
2010-03-16T12:13:09+00:00 Priority Level - INFO	Outputting step two.
Step two validation has passed.	2010-03-16T12:14:48+00:00 Priority Level - INFO
2010-03-16T12:13:09+00:00 Priority Level - INFO	Received request to go to step 3 of regEvent.
Outputting step three.	2010-03-16T12:14:48+00:00 Priority Level - INFO
2010-03-16T12:14:23+00:00 Priority Level - INFO	Step two validation has passed.
Received request to go to step 4 of regEvent.	2010-03-16T12:14:48+00:00 Priority Level - INFO
2010-03-16T12:14:23+00:00 Priority Level - INFO	Outputting step three.
Validation for the Email event successful.	2010-03-16T12:15:26+00:00 Priority Level - INFO
2010-03-16T12:14:23+00:00 Priority Level - INFO	Received request to go to step 4 of regEvent.
Outputting step four.	2010-03-16T12:15:26+00:00 Priority Level - INFO
2010-03-16T12:14:23+00:00 Priority Level - INFO	Validation for the Email event successful.
Registering event: Registered event data.	2010-03-16T12:15:26+00:00 Priority Level - INFO
2010-03-16T12:14:23+00:00 Priority Level - INFO	Outputting step four.
Registering event: Registered puzzle data.	2010-03-16T12:15:26+00:00 Priority Level - INFO

Developing a Game Engine for the Creation and Management of Alternate Reality Games

Registering event: Registered event data.
2010-03-16T12:15:26+00:00 Priority Level - INFO
Registering event: Registered puzzle data.
2010-03-16T12:15:26+00:00 Priority Level - INFO
Event successfully registered - COMPLETE.
2010-03-16T12:15:29+00:00 Priority Level - INFO
Received request to view an arc.
2010-03-16T12:15:48+00:00 Priority Level - INFO
Reached game page.
2010-03-16T12:15:52+00:00 Priority Level - INFO
Reached game page.
2010-03-16T12:15:52+00:00 Priority Level - INFO
--GAME STARTED Pressed by user.--
2010-03-16T12:17:44+00:00 Priority Level - INFO
Index Page reached.

Test Case Four

2010-03-16T12:52:28+00:00 Priority Level - INFO
Index Page reached.
2010-03-16T12:52:30+00:00 Priority Level - INFO
Characters front page reached.
2010-03-16T12:52:30+00:00 Priority Level - ERR
No characters exist.
2010-03-16T12:52:32+00:00 Priority Level - INFO
Displaying the create character form.
2010-03-16T12:52:41+00:00 Priority Level - INFO
Received submit request to create a new character - valid.
2010-03-16T12:52:44+00:00 Priority Level - INFO

Characters front page reached.
2010-03-16T12:52:45+00:00 Priority Level - INFO
Displaying the create character form.
2010-03-16T12:53:01+00:00 Priority Level - INFO
Received submit request to create a new character - valid.
2010-03-16T12:53:04+00:00 Priority Level - INFO
Characters front page reached.
2010-03-16T12:53:11+00:00 Priority Level - INFO
Received request to view a character.
2010-03-16T12:53:15+00:00 Priority Level - INFO
Received request to go to step 1 of regCharService.
2010-03-16T12:53:15+00:00 Priority Level - INFO
Outputting step one.
2010-03-16T12:53:19+00:00 Priority Level - INFO
Received request to go to step 2 of regCharService.
2010-03-16T12:53:19+00:00 Priority Level - INFO
Step one validation has passed.
2010-03-16T12:53:19+00:00 Priority Level - INFO
Outputting step two.
2010-03-16T12:53:19+00:00 Priority Level - INFO
Returning the address of the template file - emailReg.tpl.
2010-03-16T12:53:50+00:00 Priority Level - INFO
Received request to go to step 3 of regCharService.
2010-03-16T12:53:50+00:00 Priority Level - ERR
Authentication with email service failed as the port is invalid.
2010-03-16T12:53:50+00:00 Priority Level - INFO
Step one validation has passed.

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T12:53:50+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T12:53:50+00:00 Priority Level - INFO

Returning the address of the template file - emailReg.tpl.

2010-03-16T12:54:04+00:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-03-16T12:54:04+00:00 Priority Level - INFO

Validation with the email web service successful.

2010-03-16T12:54:04+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T12:54:04+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T12:54:04+00:00 Priority Level - INFO

Registering service - created user.

2010-03-16T12:54:04+00:00 Priority Level - INFO

Registering service - created service.

2010-03-16T12:54:04+00:00 Priority Level - INFO

Successfully registered email details with the database.

2010-03-16T12:54:04+00:00 Priority Level - INFO

Registering service - COMPLETED.

2010-03-16T12:54:09+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T12:54:10+00:00 Priority Level - INFO

Received request to view a character.

2010-03-16T12:54:12+00:00 Priority Level - INFO

Received request to go to step 1 of regCharService.

2010-03-16T12:54:12+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T12:54:15+00:00 Priority Level - INFO

Received request to go to step 2 of regCharService.

2010-03-16T12:54:15+00:00 Priority Level - INFO

Step one validation has passed.

2010-03-16T12:54:15+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T12:54:15+00:00 Priority Level - INFO

No further registration needed for Blogger. Nothing to output.

2010-03-16T12:54:39+00:00 Priority Level - INFO

Received request to go to step 3 of regCharService.

2010-03-16T12:54:39+00:00 Priority Level - INFO

Validation with the blogger web service successful.

2010-03-16T12:54:39+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T12:54:39+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T12:54:39+00:00 Priority Level - INFO

Registering service - created user.

2010-03-16T12:54:39+00:00 Priority Level - INFO

Registering service - created service.

2010-03-16T12:54:39+00:00 Priority Level - INFO

No further registration needed for Blogger. SUCCESSFUL.

2010-03-16T12:54:39+00:00 Priority Level - INFO

Registering service - COMPLETED.

2010-03-16T12:54:46+00:00 Priority Level - INFO

Characters front page reached.

2010-03-16T12:54:51+00:00 Priority Level - INFO

Reached storyline page.

Developing a Game Engine for the Creation and Management of Alternate Reality Games

2010-03-16T12:54:51+00:00 Priority Level - INFO

No arcs exist in database.

2010-03-16T12:54:52+00:00 Priority Level - INFO

Reached services page.

2010-03-16T12:54:55+00:00 Priority Level - INFO

Reached the factions page..

2010-03-16T12:54:55+00:00 Priority Level - INFO

No factions exist in the database.

2010-03-16T12:54:57+00:00 Priority Level - INFO

Index Page reached.

2010-03-16T12:55:12+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T12:55:12+00:00 Priority Level - INFO

No arcs exist in database.

2010-03-16T12:55:13+00:00 Priority Level - INFO

Received request to display the create arc form.

2010-03-16T12:55:26+00:00 Priority Level - INFO

Received submit request to create a arc - valid.

2010-03-16T12:55:31+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T12:55:33+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T12:55:34+00:00 Priority Level - INFO

Received request to go to step 1 of regEvent.

2010-03-16T12:55:34+00:00 Priority Level - INFO

Outputting step one.

2010-03-16T12:55:39+00:00 Priority Level - INFO

Received request to go to step 2 of regEvent.

2010-03-16T12:55:39+00:00 Priority Level - INFO

Step one validation passed.

2010-03-16T12:55:39+00:00 Priority Level - INFO

Outputting step two.

2010-03-16T12:55:42+00:00 Priority Level - INFO

Received request to go to step 3 of regEvent.

2010-03-16T12:55:42+00:00 Priority Level - INFO

Step two validation has passed.

2010-03-16T12:55:42+00:00 Priority Level - INFO

Outputting step three.

2010-03-16T12:56:35+00:00 Priority Level - INFO

Received request to go to step 4 of regEvent.

2010-03-16T12:56:35+00:00 Priority Level - INFO

Validation for the Email event successful.

2010-03-16T12:56:35+00:00 Priority Level - INFO

Outputting step four.

2010-03-16T12:56:35+00:00 Priority Level - INFO

Registering event: Registered event data.

2010-03-16T12:56:35+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T12:56:35+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T12:56:43+00:00 Priority Level - INFO

Index Page reached.

2010-03-16T12:56:47+00:00 Priority Level - INFO

Reached storyline page.

2010-03-16T12:56:49+00:00 Priority Level - INFO

Developing a Game Engine for the Creation and Management of Alternate Reality Games

Received request to view an arc.	2010-03-16T12:57:36+00:00 Priority Level - INFO
2010-03-16T12:56:52+00:00 Priority Level - INFO	Event successfully registered - COMPLETE.
Received request to go to step 1 of regEvent.	2010-03-16T12:57:44+00:00 Priority Level - INFO
2010-03-16T12:56:52+00:00 Priority Level - INFO	Received request to view an arc.
Outputting step one.	2010-03-16T12:58:10+00:00 Priority Level - INFO
2010-03-16T12:56:56+00:00 Priority Level - INFO	Received request to go to step 1 of regEvent.
Received request to go to step 2 of regEvent.	2010-03-16T12:58:10+00:00 Priority Level - INFO
2010-03-16T12:56:56+00:00 Priority Level - INFO	Outputting step one.
Step one validation passed.	2010-03-16T12:58:12+00:00 Priority Level - INFO
2010-03-16T12:56:56+00:00 Priority Level - INFO	Received request to go to step 2 of regEvent.
Outputting step two.	2010-03-16T12:58:12+00:00 Priority Level - INFO
2010-03-16T12:56:58+00:00 Priority Level - INFO	Step one validation passed.
Received request to go to step 3 of regEvent.	2010-03-16T12:58:12+00:00 Priority Level - INFO
2010-03-16T12:56:58+00:00 Priority Level - INFO	Outputting step two.
Step two validation has passed.	2010-03-16T12:58:14+00:00 Priority Level - INFO
2010-03-16T12:56:58+00:00 Priority Level - INFO	Received request to go to step 3 of regEvent.
Outputting step three.	2010-03-16T12:58:14+00:00 Priority Level - INFO
2010-03-16T12:57:32+00:00 Priority Level - INFO	Step two validation has passed.
Received request to go to step 4 of regEvent.	2010-03-16T12:58:14+00:00 Priority Level - INFO
2010-03-16T12:57:32+00:00 Priority Level - INFO	Outputting step three.
Validation for the Blogger event successful.	2010-03-16T12:58:44+00:00 Priority Level - INFO
2010-03-16T12:57:32+00:00 Priority Level - INFO	Received request to go to step 4 of regEvent.
Outputting step four.	2010-03-16T12:58:44+00:00 Priority Level - INFO
2010-03-16T12:57:36+00:00 Priority Level - INFO	Validation for the Email event successful.
Registering event: Registered event data.	2010-03-16T12:58:44+00:00 Priority Level - INFO
2010-03-16T12:57:36+00:00 Priority Level - INFO	Outputting step four.
Registering event: Registered puzzle data.	2010-03-16T12:58:44+00:00 Priority Level - INFO

Developing a Game Engine for the Creation and Management of Alternate Reality Games

Registering event: Registered event data.

2010-03-16T12:58:44+00:00 Priority Level - INFO

Registering event: Registered puzzle data.

2010-03-16T12:58:44+00:00 Priority Level - INFO

Event successfully registered - COMPLETE.

2010-03-16T12:58:48+00:00 Priority Level - INFO

Received request to view an arc.

2010-03-16T12:58:55+00:00 Priority Level - INFO

Reached game page.

2010-03-16T12:58:58+00:00 Priority Level - INFO

Reached game page.

2010-03-16T12:58:58+00:00 Priority Level - INFO

--GAME STARTED Pressed by user.--

2010-03-16T12:59:35+00:00 Priority Level - INFO

Index Page reached.