Adapting Feature-Driven Software Development Methodology to Design and Develop Educational Games in 3-D Virtual Worlds

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Sertac Ozercan

June 2010

This thesis titled

Adapting Feature-Driven Software Development Methodology to Design and Develop Educational Games in 3-D Virtual Worlds

by

SERTAC OZERCAN

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Chang Liu

Associate Professor of Electrical Engineering and Computer Science

---

Dennis Irwin

Dean, Russ College of Engineering and Technology

ABSTRACT

OZERCAN, SERTAC, M.S., June 2010, Computer Science

Adapting Feature-Driven Software Development Methodology to Design and Develop Educational Games in 3-D Virtual Worlds (104 pp.)

Director of Thesis: Chang Liu

Can educational games in 3-D virtual worlds be designed and developed using an effective software development methodology, which supports multi-disciplinary development with a small team size and frequent changes? Can this improve motivation for learning educational content and increase learning effectiveness? The research in this thesis discusses the effectiveness of feature-driven development in development of educational games in 3-D virtual worlds. The thesis describes two research projects that were developed using feature-driven development. First project is a financial literacy game, aimed at high school students to increase their financial literacy by using a 3-D virtual environment to simulate their personal finance. Second project, Second Life for Statistics Homework Sessions, is a learning aid for statistics homework sessions designed for university students to understand abstract statistical concepts. The projects were deployed to schools for testing and evaluation. The results show improvements in understanding of the domain concepts and indicate that these educational games can motivate students and contribute to learning.

Approved: _____

Chang Liu

Associate Professor of Electrical Engineering and Computer Science

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Chang Liu for founding the Virtual Immersive Technologies and Arts for Learning Laboratory and for his help and guidance. Thanks to Dr. Schwerha and Dr. Franklin for their support and guidance. Thanks to the members of the projects, particularly Jarrod Reuter, and Lev Neiman, for their great contributions.

I would also like to thank my family for always supporting and encouraging me.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ASD – Adaptive Software Development

DSDM – Dynamic Systems Development Method

FDD – Feature-Driven Development

HCI – Human-Computer Interaction

SDLC – Software Development Life Cycle

SL – Second Life

UML – Unified Modeling Language

XP – Extreme Programming

CHAPTER 1: INTRODUCTION

This chapter begins with an overview of the research work. It continues with description of software development, development practices and discusses the recent software engineering trends in terms of project successes. The chapter concludes with description and discussion of traditional plan-driven and agile software development methodologies and discusses their effectiveness in today's software development requirements.

Section 1: Thesis Overview

Research Objectives and Contributions

This thesis details a research project that uses feature-driven development, an agile software development methodology, in development of educational games in 3-D virtual worlds.

Agile software development is not a new topic in academic literature; however, developing educational games in 3-D virtual worlds using an agile methodology has not been examined thoroughly. Using an agile software development, educational games can be developed effectively, by addressing frequently changing, multi-disciplinary nature of the development environment by small teams. The educational games increase students' motivation for learning and improve learning effectiveness.

The researcher worked on the design, implementation and deployment of the Financial Education Game and Second Life for Statistics Homework Sessions. The experiment testing and results are shown in Chapter 5. The researcher worked on the development approach and analysis of using Feature-Driven Development software

development methodology in design and development of educational games in 3-D virtual worlds. Development approach is discussed in Chapter 4.

This thesis project incorporates research work that has been conducted with the Virtual Immersive Technologies and Arts for Learning (VITAL) laboratory at Ohio University. The following people contributed to the Financial Education Game; Jarrod Reuter, En Ye, Dr. Chang Liu, Dr. Teresa Franklin, Dr. Roger Shelor, Scott Moriarty, and Dr. Li Wei Peng. Jarrod Reuter and the researcher worked as the developer for the game. En Ye, and Dr. Chang Liu worked as the project managers. Scott Moriarty contributed as the project owner. Dr. Teresa Franklin and Dr. Roger Shelor contributed as the educational domain expert, and the financial domain expert, respectively. Dr. Li Wei Peng worked on the design of the questions for the schools, and on the statistical analysis of the data acquired. The following people contributed to the Second Life for Statistics Homework Sessions; Tripura Vadlamani, Lev Neiman, Dr. Diana Schwerha, and Dr. Chang Liu. The researcher and Lev Neiman contributed as the developers for the game. Tripura Vadlamani and Dr. Diana Schwerha worked on design of the questions and statistical analysis of the data.

Structure of this Thesis

This chapter presents background information on software development, and defines and analyzes traditional and agile software development methodologies. Serious game development process is discussed in Chapter 2. Chapter 3 introduces and discusses feature-driven development, an agile software development methodology. Chapter 4 introduces the case studies that the researcher designed and developed using the 3-D

virtual world Second Life, developed by Linden Lab. The education games that were developed are Financial Literacy Game, which teaches financial education to teens, and Second Life for Statistics Homework Sessions, which improves the traditional statistics homework sessions for university students. Chapter 4 explains the case studies in detail. The games were tested and evaluated with high school and university students. Chapter 5 discusses the experiment testing and results. Finally, Chapter 6 concludes the research work presented.

Goals of the Case Studies

The goals of the case studies are designing and developing educational games in 3-D virtual worlds effectively, by addressing frequently changing, multi-disciplinary nature of the development by small teams, and motivating students to learn educational content and improving the learning effectiveness by presenting the educational concepts in a fun and easy to understand way. The results yielded an increase in students' understanding of the educational topics and improvement in their motivation for learning educational content.

Section 2: Software Development

This subsection begins with description of software development and development practices. The subsection continues with discussion of the recent software development trends in terms of project successes.

Early Years of Software Development

Software has been around for more than 6 decades. In the early days, the development approach for software was not organized and was mostly referred to as

‘code and fix’ which was followed by a testing phases until it is ‘feature complete’ (Fowler, 2005). As software systems evolved and got increasingly more complicated, this approach could not support the software development. According to Russ and McGregor (2000), as systems become more complex, activities of a project require a formal structure. The idea of software development methodology came from this need for a structure.

Baskerville, Travis and Truex (1992) define methodology as “the set of known methods adapted for the development of computer-based information systems” (p. 241). Methodologies aim to provide efficient, reliable, repeatable, predictive and disciplined process to software development.

Different software development methodologies will be discussed in further detail in the next sections.

Recent Trends in Software Development

Charette (2005) provides common factors for software projects failures. Among the factors are many that are closely related to software engineering, such as “Sloppy development practices”, “Badly defined system requirements”, “Poor reporting of the project's status” and “Unrealistic or unarticulated project goals” (p. 43).

As mentioned, software engineering is a complex field, thus the overall success of software projects are somewhat low. According to Standish Group 1999 and 2004 reports, the failure rate in the software industry was 20 percent in 1999, and it improved to 16 percent in 2004, and successes increased to 34 percent in 2004 (Reel, 1999, Bain, 2008). However, the accuracy of these reports has been debated (Glass, 2006).

Section 3: Traditional Software Development Methodologies

This subsection describes the traditional software development methodologies. It opens with explaining typical stages of traditional software methodologies and concludes with critique of traditional methodologies.

What is Traditional Software Development?

As mentioned in the previous section, structure was needed to overcome the 'code and fix' approach used in the 1960s. In 1970, Winston W. Royce laid out the waterfall methodology; Although, Royce was actually suggesting that this approach to software development would not work efficiently for software development. However, this misinterpreted method became the standard for many organizations, including United States Department of Defense (DOD-STD-2167) (Coad, 1988).

Since the early years, many methodologies were proposed and these methodologies have received revisions and improvements over time. These mature process, plan and documentation-driven methodologies are referred to as the traditional methodologies. Waterfall methodology is one of the first, most famous and widely used among the traditional methodologies. In this thesis, waterfall methodology will be considered as the traditional software development approach.

Waterfall methodology is typically based on linear steps, and brought structure and formality to software development (Royce, 1970). There are typically five stages in traditional software development methodology. First stage is gathering of user requirements in which system and software requirements are determined. In second stage, the requirements are transformed into software design and documentation for the

software is established. After documentation has been completed, the development phase is started. The next phase is testing, where the software is tested and bugs and issues are fixed. Finally, the last stage is deployment and maintenance. Important thing to note is that all phases are linear, once you start a step there is no going back.

Waterfall methodology can be compared to construction project for a bridge. Building process of a bridge is repeatable; environment and conditions change but it is still a similar process. Bridges can be built linearly and designs rarely change once the requirements are determined.

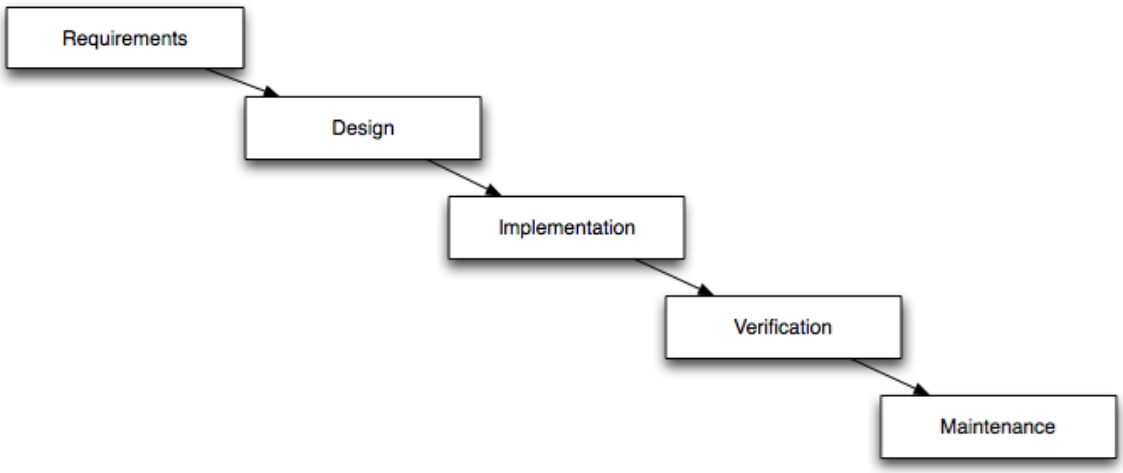

Figure 1. Waterfall Software Development Methodology.

Critique of Traditional Software Development

Traditional software development methodologies became ineffective in today's rapidly changing software industry. There are a number of reasons for this ineffectiveness.

First of all, customers should provide the developers with a list of requirements in the first stage in a traditional methodology. However, it is common that customers do not have a complete understanding of their own requirements at the beginning of a project (Jeffries, Anderson & Hendrickson, 2000). This lack of understanding affects the software projects negatively, as software development depends on it when a project begins. When customers are unclear of the project requirements in the beginning, projects contain unused or rarely used features (Abrahamsson, 2005). According to Standish Group keynote speech in 2002, 64 percent of the features are never or rarely used (Hibbs, Jewett & Sullivan, 2003).

Furthermore, the documentation stage of traditional methodologies only provides 15 percent to 20 percent of shared understanding for the project (Highsmith, 2002). Shared understanding of a project is crucial to the development since it provides a common background of experience and knowledge (Tran & Biddle, 2008). Moreover, it increases productivity as team members do not have to explain basic details (Tran & Biddle, 2008).

The development documentation is hard and costly to change. The Economist Technology Quarterly of June 2003 reports that "A bug that costs $1 to fix on the programmer's desktop costs $100 to fix once it is incorporated into a complete program, and many thousands of dollars if it is identified only after the software has been deployed in the field. In some cases, the cost can be far higher: a bug in a piece of telecoms-routing equipment or an aircraft control system can cost millions to fix if the equipment is taken out of service" ("Building a better bug-trap," 2003, p. 18).

Traditional methodologies require separate processes for their structure. This process separation requires detailed communication between team members and leads to project members specializing in one part of the software development life cycle (SDLC) (Good, 2003). The need for specialization in a process or task can generate problems for the project, as team members become unavailable (i.e. leave the team, move to another department or company) (Good, 2003). In addition to this, developers may need to wait on other processes or departments to complete their task for implementation.

Outcomes for traditional software development are not very favorable. According to the CHAOS Report from Standish Group in 2001, only 28 percent of software projects that used the waterfall methodology succeeded, while 23 percent of the projects failed and 49 percent was challenged (Standish Group, 2004).

Section 4: Agile Software Development Methodologies

This subsection describes the agile software development methodologies. The subsection begins with describing what agile software development is and summarizes the most widely used agile methodologies. It continues with the agile manifesto and agile principles. The subsection continues by discussing why agile software development should be considered. Finally, it concludes with critique of agile software development.

What is Agile Software Development?

Merriam-Webster Online Dictionary defines agile as “marked by ready ability to move with quick easy grace, having a quick resourceful and adaptable character” (Merriam-Webster, 2009). The essence of agile software development methodology is similar in terms of frequent changes and adapting to those changes.

Agile software development approach offers a number of advantages over traditional development approaches, such as context-specific development, increased customer satisfaction, lower defect rates, faster development and responsiveness to rapidly changing requirements (Salo & Abrahamsson, 2007).

Most popular agile methods include Extreme Programming (XP), Feature Driven Development (FDD), Scrum, Adaptive Software Development (ASD), Crystal Methodologies, and Dynamic System Development Method (DSDM). Some of the most important methodologies are summarized below.

Extreme Programming (XP)

Extreme programming (XP) is an agile methodology that focuses on small development cycles, user stories, and pair programming. XP incorporates principles of communication, simplicity, feedback, and courage into its approach.

Beck and Andres (2004) outline the main practices of XP approach:

1. *Planning game*: Used to determine the scope of the project; it includes customer stories. Customer stories provide an efficient method for handling customer requirements. Customers use small index cards to write down features they want in the project and prioritize them accordingly.
2. *Small releases*: Produce smaller releases on short development cycles.
3. *Metaphor*: Provides shared understanding by using a metaphor to describe the system.
4. *Simple design*: Develop only required features. *You Aren't Going to Need It* (YAGNI) principle advocates removing unneeded elements.

5. *Testing*: Only develop features that can be automated for testing.
6. *Refactoring*: Restructuring of the system in order to simplify and remove duplicates.
7. *Pair programming*: Most controversial but helpful practice when used correctly, especially after considering the time consumed by developers when they are stuck working on an issue or just surfing the web or checking their email (McFarland, 2006). Two developers work together to develop code, while finding mistakes of each other and come up with solutions together (Constantine, 1995) to produce higher quality code introduced earlier in the cycle (Williams & Kessler, 2003).
8. *Collective ownership*: Any developer can change any code in the system.
9. *Continuous integration*: Releasing multiple builds a day. Continuous integration reduces conflicting changes (Beck & Andres, 2004) and shortens feedback cycle with rapid feedback and fixes.
10. *40-hour week*: Working maximum of 40 hours a week.
11. *On-site customer*: Include customer in the project team for questions and feedback.
12. *Coding standards*: Code accordingly to the previously determined rules and standards.

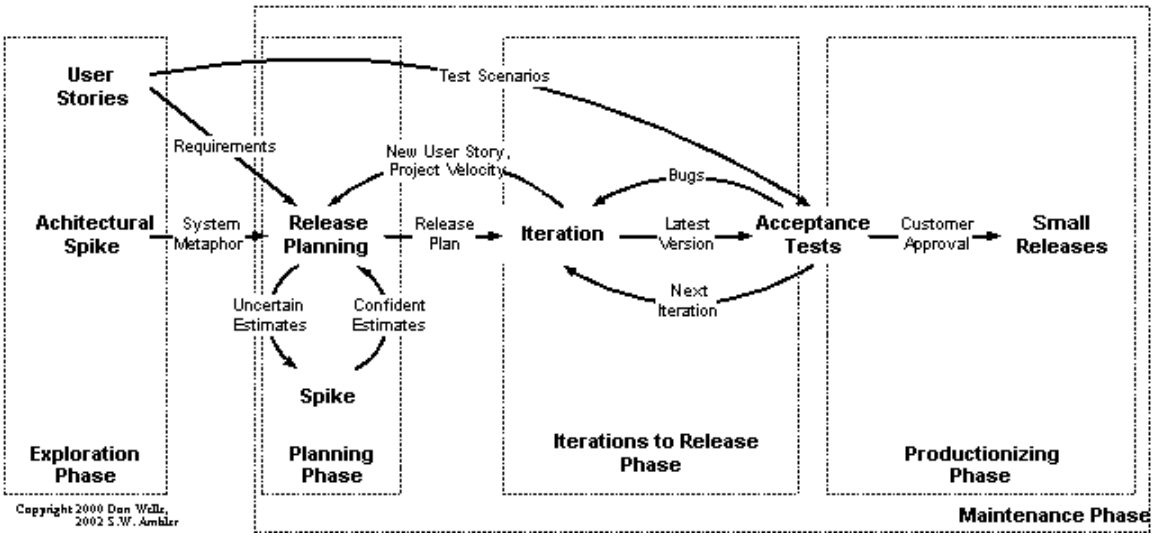

Figure 2. XP Process (used with permission from (Ambler, 2001))

Scrum

Introduced in 1996 by Ken Schwaber and Jeff Sutherland, Scrum focuses on the project management approach in agile software development. In Scrum, iterations are called sprints and each sprint is 30 days long. Sprints start with a meeting to determine the requirements in that sprint.

Schwaber and Beedle (2001) outline the main characteristics of Scrum:

1. *Product Backlog*: Produced in the initial sprint meeting. Contains requirements, tasks, defects and change requests.
2. *Sprint Backlog*: Established in the meeting by development team to determine the development details of the high-priority requirements.
3. *Sprint Review Meeting*: After the sprint has been finished, the project team inspects the project status.
4. *Daily Scrum*: Daily 15-minute meeting to answer the following three questions:

What have you done since last meeting?

What will you do between now and next meeting?

What obstacles stood in the way of doing work?

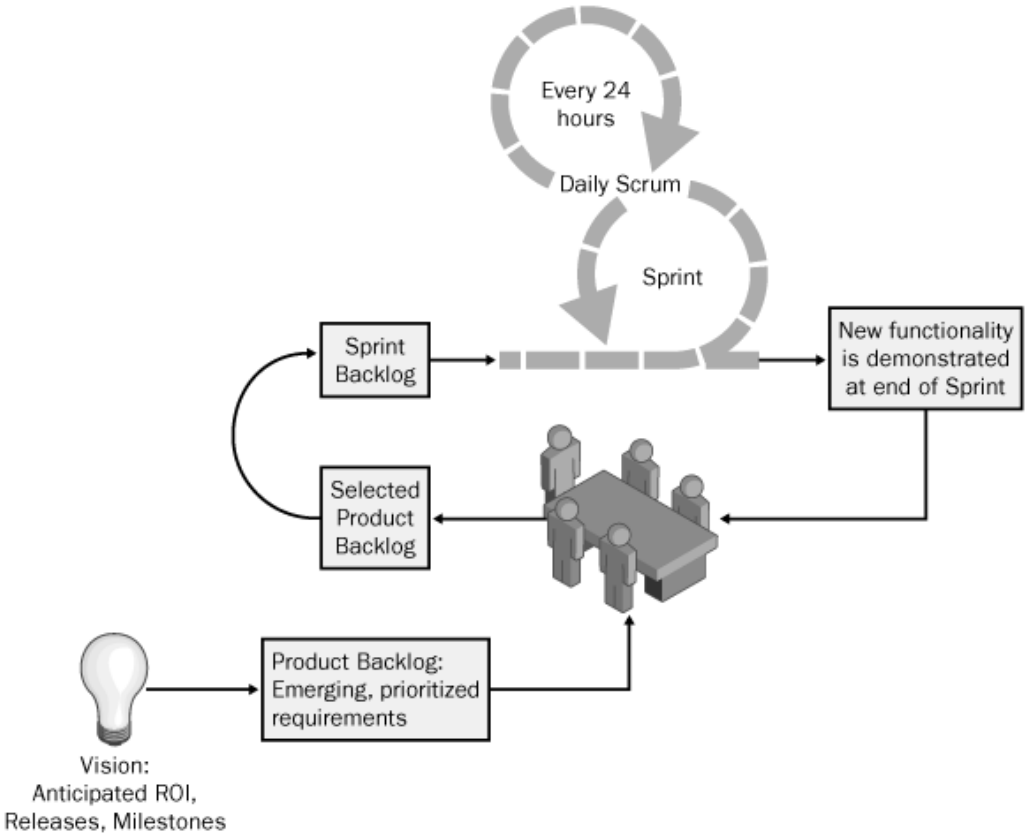

Figure 3. The Overall Scrum Process (used with permission from (Schwaber, 2004))

Adaptive Software Development (ASD)

Developed by Jim Highsmith, Adaptive Software Development (ASD) cannot really be called a software development methodology but a culture. ASD acknowledges uncertainty and unpredictability in software development and provides a framework to

deliver by making short releases, and learning from mistakes. ASD uses three non-linear and overlapping phases: *Speculate*, *Collaborate*, and *Learn* (Highsmith, 2002).

Feature Driven Development (FDD)

Created by Jeff De Luca in late 1990s, Feature-Driven Development (FDD) is an agile software development methodology, which focuses on customer requested features, grouping of these features into areas and iteratively designs and builds the features. FDD includes milestones for project status tracking and produces two-week regular builds to keep customers as up-to-date as possible.

Palmer and Felsing (2002) outline the main characteristics of FDD approach:

1. *Domain Object Modeling*: Creation of a visual model (class, UML) depicting the system. Model is updated in each iteration with new features.
2. *Developing by Feature*: Implement only the required features by the customer.
3. *Individual Class (Code) Ownership*: Each developer develops and maintains a class that is assigned to him or her.
4. *Feature Teams*: Working together to find a solution to a feature.
5. *Inspections*: Includes unit testing, and design and code inspections.
6. *Configuration Management*: Tracking changes.
7. *Regular Builds*: Demonstrate to the customer as up-to-date as possible.
8. *Visibility of progress and results.*

Feature-driven development will be explained in detail in Chapter 3.

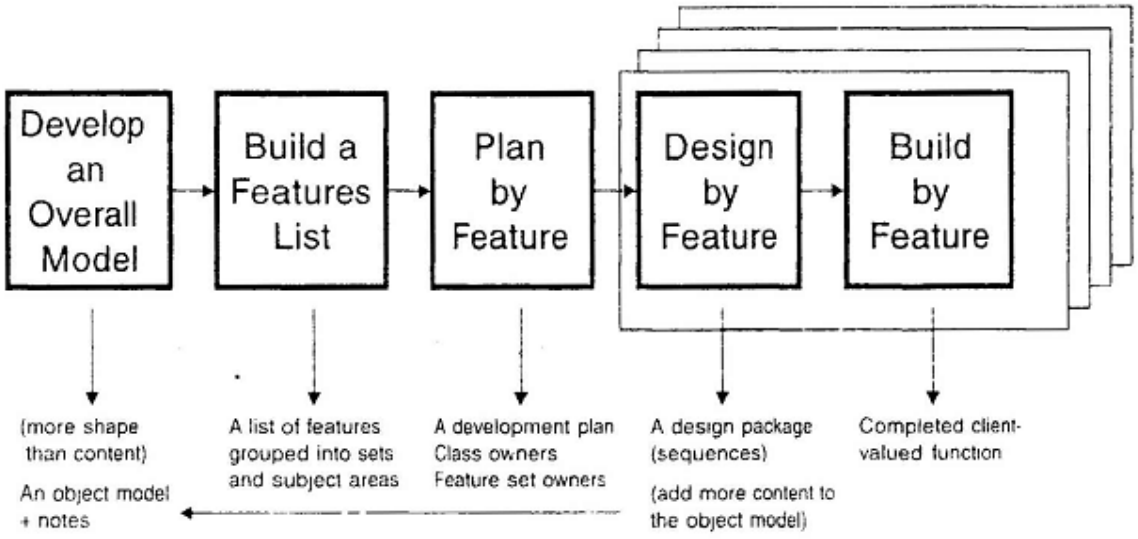

Figure 4. FDD's Development Process (used with permission from (Palmer & Felsing, 2002))

Agile Manifesto

Agile manifesto is a statement of principles that methodologists, who named themselves Agile Software Development Alliance, which was established in 2001. Highsmith and Fowler (2001) describe the manifesto containing the following values:

1. "*Individuals and interactions* over processes and tools" (p. 29-30)
2. "*Working software* over comprehensive documentation" (p. 29-30)
3. "*Customer collaboration* over contract negotiation" (p. 29-30)
4. "*Responding to change* over following a plan" (p. 29-30)

The manifesto states that agile development gives left side priority over right. These agile development values can be summarized as:

The first value states that it gives priority to communication and collaboration of competent people rather than industry processes and tools.

The second value stands for thoroughly tested, working and updated software is preferred over the abundant and possibly outdated and unnecessary documentation.

The third value states that active communication and participation of the customer is more important than completing requirements on a contract.

Finally, the fourth value stands for understanding that the change is inevitable and development should embrace change so it can adapt to rapidly changing industry requirements.

Why Use Agile Software Development?

Agile software development offers many advantages over traditional methodologies. First of all, customer satisfaction is the priority of agile software development. It provides satisfaction to the customer with rapid and continuous development (Highsmith & Fowler, 2001).

Agile software development minimizes the risks in any software project by increasing agility of development. It embraces change and adapts to rapidly changing requirements (Beck & Andres, 2004).

Testing is an integrated and continuous part of agile development process rather than a separate process (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). Functional testing methods like unit, component and integration testing provide less bug-prone software. Furthermore, documentation of user requirements before development is not going to be accurate, self-documenting code or reverse-engineering tools can be useful.

Agile software development is widely used in industry, by companies like Google and Microsoft (McFarland, 2006, West, 2009). Industry surveys for agile development

over the previously used methodologies show that agile development methodology provides 93 percent increased productivity, 88 percent increased quality for agile development (Shine Technologies, 2003). Furthermore, Reifer (2002) reports that average of 15 percent to 23 percent productivity improvement and 5 percent to 7 percent cost reduction based on industry benchmarks, 25 percent to 50 percent less development time compared to previous projects, and defect rate was on par with previous projects.

Critique of Agile Software Development

Agile software development has its shortcomings depending on the methodology used, developers, customers, organizational culture, and people. It will be analyzed based on these criteria.

Typically agile software development work best in smaller teams with creative and experienced members. It has been questioned that whether agile software development's success is based on competent individuals in the project team. Cockburn and Highsmith (2001) discusses that if individuals are good enough, they can use any process to succeed. However, when people collaborate they bring their skills to a higher level. Cockburn and Highsmith (2001) continue with "agile project teams focus on increasing both individual competencies and collaboration levels" (p. 132). Lindvall, Basili, Boehm, Costa, Dangle, Shull, et al. (2002) agrees with agile development increasing individual's skills and adds "It was estimated that 25%-33% of the project personnel must be 'competent and experienced', but the necessary percentage might even be as low as 10% if the teams practice pair programming due to the fact that they mentor each other." (p. 7).

Organizational culture is crucial while applying agile software development. Agile methods will not work if the organizational culture is not suitable. Nerur, Mahapatra and Mangalaraj (2005) identifies that agile software development requires favoring or shift to "leadership-and-collaboration" management style to achieve the flexibility and responsiveness needed to embrace agile development (p. 76). Nerur et al. (2005) continues by stating organizations that are more open to innovation will be able to transition to agile development easier than organizations with high bureaucracy and formalization.

Documentation of a project may contain critical information. Agile development tries to avoid documentation as it focuses on working software rather than outdated and unnecessary documentation. However, if project members leave the team for various reasons, it might be hard to get a new member up to speed without any documentation. On the other hand, as mentioned earlier, self-documenting code and reverse-engineering tools can help with this issue and with XP's pair programming approach the new team member can quickly learn the project.

Agile development values customer and satisfaction of the customer vital to success of the project. In some agile methodology, such as XP, there is an on-site customer present. However, if multiple customers have different viewpoints about an issue, it might cause a conflict in the project. With a traditional methodologies documentation phase, this might not be a big issue.

Section 5: Comparison of Agile and Traditional Software Development Methodologies

Agile practitioners believe that change is inevitable and necessary for innovation; while traditional methodology values minimizing change, emphasizing prediction and control (Vinekar, Slinkman & Nerur, 2008). Boehm (2002) identifies the areas the areas and conditions that whether agile and traditional methods can be advantageous. It is shown in Table 1.

Table 1: Home ground for agile and plan-driven methods (Boehm, 2002)

Home-ground area	**Agile methods**	**Plan-Driven methods**
Developers	Agile, knowledgeable, collocated, and collaborative	Plan-oriented; adequate skills; access to external knowledge
Customers	Dedicated, knowledgeable, collocated, collaborative, representative, and empowered	Access to knowledgeable, collaborative, representative, and empowered customers
Requirements	Largely emergent; rapid change	Knowable early; largely stable
Architecture	Designed for current requirements	Designed for current and foreseeable requirements
Refactoring	Inexpensive	Expensive
Size	Smaller teams and products	Larger teams and products
Primary Objective	Rapid value	High assurance

CHAPTER 2: SERIOUS GAME DEVELOPMENT

This chapter opens with a discussion of serious games for educational purposes in 3-D virtual worlds. It closes by discussing the game development process in software engineering.

Section 1: Educational Games

This section begins by describing serious educational games. It continues by discussing how educational games can benefit learning, and concludes with different approaches for the design of educational games.

Serious Games

Serious games are defined as games that prioritize education as primary purpose instead of entertainment purposes (Michael & Chen, 2006). While the ultimate goal of serious games is learning, having fun is highly encouraged. Over the years, education was improved by various tools, such as movies, books, and television; today, serious games are emerging to improve motivation and learning effectiveness as they become accepted and widely available (Michael & Chen, 2006). Tran points out that as serious games become more popular, there needs to have a better understanding other than the commercial entertainment market (Tran & Biddle, 2008).

Why Educational Games for Learning?

Dede, Clarke, Ketelhut, Nelson and Bowman (2005) argue that lectures in science classrooms are usually about abstract concepts and ideas and students are rarely given the chance to interact with the scientific experiments first hand and practice science in the classroom. Furthermore, as discussed by Long (2003), Deci, and Ryan (1985) claim that

learning process is more fun when the leaner has more autonomy, which in turn increases learner's motivation. These findings show that learners need to perform these interactive activities to increase their motivation and their learning effectiveness.

Motivation is vital for learning new things. Chan and Ahern (1999) claim that "when people are intrinsically motivated to learn, they not only learn more, they also have a more positive experience" (p. 152). Furthermore, Bizzocchi and Paras (2005) supports this and argues that well-designed games provide motivation that supports the learning process and learning outcome of well-designed games provides a motivating experience. Students are motivated for learning when they play games that support education and provide them with fun ways to learn educational content. The experience needs to be challenging for the learner to be effective. Chan and Ahern (1999) points out that facilitating motivation requires experience to be challenging or it will either be ignored by the learner or will irritate the learner.

Different Approaches to the Design of Educational Games

Moreno-Ger, Burgos, Martinez-Ortiz, Sierra and Fernandez-Manjon (2008) present different approaches on designing of educational games. First approach is multimedia material directly related to the educational content. Next approach is re-implementation of existing games based on the educational content. Last approach is the middle ground, where the game is specifically designed for the educational purpose to present the educational content and encouraging users to have fun, while increasing motivation and learning effectiveness (Moreno-Ger, et al., 2008).

The researcher chose the last option, where the game is designed and developed from the beginning with the educational content as the primary purpose, while making sure students are having fun in the game; increasing their motivation and supporting the learning process.

Section 2: Software Engineering in Game Development

This section begins by describing software development process in designing and development games. The section continues with discussion of traditional software development and game development. It concludes by discussing various development elements in game development.

Traditional Software Development versus Game Development

Game development is subset of software engineering that shares some of the problems related to abstract design. Petrillo, Pimenta, and Dietrich (2008) claim that games industry has all the problems found in the traditional game industry and solutions found in the traditional software development solve some problems in the game development. Game development includes additional problem unique to its field.

As discussed in Gibson and Sherba (2008), over the years game development became increasingly complex as the games evolved over time. With development and advances in various components, such as 3D graphics, animation, sound, artificial intelligence, physics, and communication and interaction between these components complexity of game development increased significantly.

Furthermore, as discussed in the previous section, the 'fun' element in games, especially educational games, proves to be a unique and significant part of game development.

Development of games should emphasize design with Human-Computer Interaction (HCI) principles in mind. An effectively designed interface increases the desire to interact and explore the system beyond one's needs (Mandel, 1997). These games are meant to be enjoyable; therefore an easy-to-use user interface with good usability will benefit the user experience greatly.

In addition to these, multi-disciplinary development teams are important in game development. Successful games require a team with various disciplines that specializes in their own fields. Multi-disciplinary development will be discussed further in the next section.

Nacke (2006) argues there is a difference between the game industry and traditional IT industries. Nacke (2006) claims that "flat hierarchy structure", "interdisciplinary teams" and "non-deferrable project schedule" are the most important features for the game industry, while "traditional hierarchy structure", "advanced training", and "stability" are the most important features in traditional industries (p. 24). Nacke (2006) explains that the flat hierarchy structure in the game industry is caused by the young age of the workers and good for innovation and provides flexibility to adapt to changes in requirements. He also explains that game developers are given non-deferrable and short time-frame for development because of marketing reasons.

Bethke (2003) identifies that electronic game industry is using ineffective software development methodologies, which are causing projects to be go over budget, over time and bug prone. Petrillo, et al. (2008) discusses the reasons for these failures and analyzes scope, schedule and technical problems as possible reasons. Among the problems identified by Petrillo, et al. (2008), most projects are cited as having problems with unreal requirements and feature creep; followed by, cutting features during the development process, design, schedule and technological problems respectively (Petrillo, 2008).

It can be identified that inefficient methodologies are causing game industry to have problems in software development. Agile software development can improve the development process in different ways.

First of all, game development requires a high level of involvement, since it requires multi-disciplinary teams and collaboration between multi-disciplinary team is essential. Frequent meetings and object diagrams ensure the shared understanding for all the members of the project team.

With the help of iterative development, game developers can ensure that game mechanics can be established in a timely manner. Iterative milestones can provide status of the projects. Agile software development facilitates adding more features in to the game easily with iterative development.

Multi-disciplinary development

Multi-disciplinary teams are important in today's game development industry. As Michael and Chen points out team of artists, modelers and musicians working on large

size game projects has grown up to 10 times the size of the programming team (Michael & Chen, 2006).

According to Tran and Biddle (2008), developing games tend to require experts with different specializations to be effective. Among the experts identified are computer scientists, domain experts, graphic artists, usability specialists, instructional designers, and project managers.

These different experts are needed for their own specialty in their fields. Computer scientists are needed for coding and scripting the game, domain experts are responsible for selection and usage of the content, graphic artists are required for developing artistic materials, such as designing of structures, textures, production of multimedia content, instructional designers to design the educational content for maximizing the learning effectiveness, usability specialists are needed to make sure the game is easy and comfortable to use for learners, and project managers to manage and organize the project.

CHAPTER 3: DEVELOPING GAMES WITH FEATURE-DRIVEN DEVELOPMENT

This chapter discusses how feature-driven software development, which is an agile software development methodology, provides an effective means for developing serious games for educational purposes with the aforementioned requirements, which leads to motivation in learning and improves learning effectiveness.

Section 1: Feature-Driven Development

What is Feature-Driven Development?

Feature-Driven Development (FDD) is an agile software development methodology that was created by Jeff DeLuca between 1997 and 1999. It is one of the most widely used agile methodologies in industry. It can be described as "right first time" approach to agile software development. FDD claims to be suitable for mission critical systems (Palmer & Felsing, 2002).

The essential element in FDD is the feature. Palmer and Felsing (2002) defines *feature* as:

"<action> <result> <object>" (p. 57)

in which object may refer to person, place or thing. For example, *calculate* the *interest* of *loaned money*.

Designing feature-driven requires simpler design which is to develop only the required and requested features. FDD focuses on continuous process, integration and small releases, which reduces conflicting changes (Beck & Andres, 2004) and shortens feedback cycle by providing rapid feedback and fixes.

Steps of Feature-Driven Development

FDD consists of two phases, containing five steps. Khramthchenko (2005) describes FDD stages as discovering the features to implement and implementing each feature. Before beginning the development, domain experts, chief programmers and chief architects needs to be selected.

Develop an Overall Model

Initial phase is the startup phase, which is performed once in a project. This phase begins by developing an overall model for the project, and it's known as shape modeling or domain object modeling. UML diagrams depicting what classes are in the domain and how they connect and interact with each other is prepared and the required methods and attributes are placed in the classes. Sequence diagrams are also prepared, if required. The models are crucial for having a shared understanding for the multi-disciplinary team. Visual models provide guidance to the developers and customers. The models are updated in each iteration with new features.

Khramthchenko (2005) identifies the three different roles and the modeling process in the domain object modeling phase. In this phase, domain experts identify user stories, developers to build class diagrams using color UML according to these user stories, and project managers to organize the project and identify team progress. It is important that all user stories are supported in the visual model that was created by the team (Khramthchenko, 2005). The model is updated with new methods, attributes and relations for every user story that extends the current model. Color UML associates color

codes with UML diagrams. It was proposed by Peter Coad, Eric Lefebvre, and Jeff De Luca (Coad, Luca & Lefebvre, 1999). Colors are associated with different archetypes to determine their type. Coad, Luca and Lefebvre (1999) defines archetypes as the following:

1. Pink (*Moment/Interval*): For business purposes, events that occur at a moment in time or in an interval of time.
2. Yellow (*Roles*): Different roles being played.
3. Blue (*Description*): Description that labels and classifies objects.
4. Green (*Party, Place, or Thing*): Tangible thing that is uniquely identifiable.

Colored UML enables building UML models faster, and promotes shared understanding for domain experts and developers. In FDD, it is encouraged for modeling to be done in teams consisting of developers and domain experts.

Build a Features List

Startup phase continues with building a detailed, prioritized features list. Features list team is formed to build the features list. Team members need to have an understanding of the basic scope of the project to develop the feature list. Features should be small and should usually take about two weeks to complete. If a feature takes more than two weeks to complete, it should be decomposed into smaller features. Short iterations provide and demonstrate working up-to-date builds to the customer. In order to group similar activities, the features are divided into feature sets and subject areas.

It is recommended to have a formal documentation available in FDD. Internal websites, such as wikis, is good place to store information about the project. Project

diagrams, list of team members, feature list, work assigned to developers, reports are examples of documentation that can be stored. These documentation become especially important in large scale projects that has long development times since it is useful to see past decisions and reasons for them. It is also important for the customer to have access to the latest requirements.

Plan by Feature

After building the feature list, *Plan by Feature* phase begins. In this phase, the order of features to be implemented is determined. The order is based on various factors such as complexity of the feature, work load of developers and dependencies between features. Planning team assigns due dates for the activities based on dependencies, work load, complexity of the development.

Furthermore, the team assigns classes for the developers. The developer is responsible for the classes that are assigned to him or her. Palmer and Felsing (2002) identify advantages and disadvantages in individual class ownership. According to Palmer and Felsing (2002), an individual is assigned to maintain and develop classes which give the individual responsibility. The class will be faster to maintain and develop when the individual is highly familiar. With many developers working on the project, individual class ownership provides easier to find an expert on a particular class when a developer needs assistance for a dependency or similar work. Finally, it gives the individual to take pride in his or her work (Palmer & Felsing, 2002).

On the other hand, individual class ownership has disadvantages. If developers are busy, code dependencies may slow down. If class owner leaves the project, it takes

time to get another developer up to speed. Collective code ownership solves the code dependency waiting issue but introduces non-ownership or ownership by few dominant developers inside the development team.

Design by Feature

After startup phase, development continues with the construction phase. Construction phase is iterative for feature sets, activities will be performed for each feature. First activity is *Design by Feature Set*, where features that are going to be implemented in approximately two weeks are selected by the chief programmer and sequence diagrams for the features is designed. Object model is updated with the new and updated class details. For each feature, these make up a design package. Design inspection is performed to review the work.

Build by Feature

Design by Feature is followed by *Build by Feature* process. In *Build by Feature* process, features selected by the chief programmer and passed successfully in the design inspection are implemented. Code is developed by the class owners. After the code is complete, unit test and code inspection is performed to make sure all the requirements are met. If it passes the inspections successfully, it will be promoted to a build.

Milestones

Definition of milestones is important in software projects, as they establish an explicit and unambiguous definition of what needs to be done (De Luca, 2003). De Luca (2003) identifies that usual developer responses to whether a feature is completed are ambiguous, there established milestones are required for a sharp definition of tasks.

FDD uses milestones to track project development progress. Each feature consists of six milestones and progress is completed sequentially. Percentages are added to indicate completeness and to provide more accurate monitoring. De Luca (2003) outlines the activities in each feature as the following:

1. *Domain Walkthrough* – 1%
2. *Design* – 40%
3. *Design Inspection* – 3%
4. *Code* – 45%
5. *Code Inspection* – 10%
6. *Promote To Build* – 1%

Domain Walkthrough, *Design* and *Design Inspection* are *Design by Feature* activities, while *Code*, *Code Inspection* and *Promote to Build* are *Build by Feature* activities.

Section 2: FDD for Serious Educational Game Development

FDD provides benefits for designing and developing serious educational games. FDD brings common agile improvements to software development, such as iterative development, faster development times, integrated testing, and rapid response to changing requirements.

FDD provides shared understanding for the multi-disciplinary team with easy to understand models. Colored UML models ensure that user stories identified by the domain experts are supported and represented accurately.

It can be scaled to apply from small to large-sized teams. FDD provides flexibility and easy adaption to work with teams of different sizes, and experience levels.

Feature list is essential for understanding the scope of the project. This understanding helps determining the development sequence, based on code dependencies, complexity of features, balancing load for developers, and customer expectations. Building a feature list, with each feature taking less than two weeks keep the team very agile and small iteration cycles makes sure that customers have an up-to-date build.

In addition to these, milestones provide systematic monitoring, accurate and easy to understand project status. As identified earlier, without sharp definition of milestones, the requirements for finishing a feature may be ambiguous. Using clear definitions and explaining these to the developers help the management of software projects more accurate.

CHAPTER 4: 3-D VIRTUAL WORLD AND CASE STUDIES

This chapter discusses the 3-D virtual world used in the research projects. First part introduces the Second Life virtual world and its architecture. Second part describes the educational games that the researcher designed and developed in Second Life virtual world using feature-driven development.

Section 1: Second Life

What is Second Life?

Second Life (SL) is one of the most popular online 3-D virtual worlds (http://www.secondlife.com). A virtual world is a computer simulated environment where players can interact with the environment and with other players through the use of their representations, which are known as avatars, in the virtual world. SL allows its residents to create their own worlds and inhabit it.

SL is massive multiplayer virtual world which allows a great number of players to be online at the same time. It has some of the characteristics of a game; however, it does not include traditional aspects, such as scores and levels, unless it is created by its residents ("What is Second Life?," n.d.). Players can find many activities in SL, such as games, arts, education, and science ("Destination Guide," n.d.).

Furthermore, SL allows players to socialize with other players. Players can use variety of methods to communicate, such as instant message (IM), group chat, public chat, and voice chat. Instant message allows players to send each other private messages. Group chat allows players who are in the same group to communicate with each other. Public chat provides a way for all nearby players to communicate with each other.

Finally, SL supports voice chat, where players can talk to each other through Voice over IP (VoIP). SL also allows various animations and gestures for players to express themselves.

In addition to these, SL includes 3-D modeling tools and scripting language called Linden Scripting Language (LSL). 3-D modeling tools allow players to create and construct objects, which they can share or sell to other players. LSL and SL architecture will be discussed further in the chapter. SL offers streaming audio and video capabilities. SL is free to use, there is no cost associated to create a SL account.

Second Life provides a powerful experience where players can express themselves in a 3-D online immersive virtual environment. Second Life was chosen as the development environment for the educational projects because of its advantages discussed.

Second Life Architecture

Linden Scripting Language (LSL) is a state-event driven, high-level language ("LSL Portal," n.d.). An LSL scripts contains at least the default state, and it may contain variables, function definitions, and events in additional states. Currently, LSL provides over 310 built-in functions, and players can define their own functions as long as its name does not conflict with built-in reserved words. LSL scripts are attached to objects, also known as prims, to function and multiple scripts can be attached to a single object, which can be combined to create more behaviors. SL provides its own editor for editing and compiling, but text from external editors can be used via copy and paste. To view and edit a script, one need to have proper permissions for the script itself, otherwise, even if

they have permissions to edit the object that the script is attached to, they may not be able to view or edit.

Scripts are compiled into executable bytecode, similar to Java. They are interpreted and stored by Linden Lab virtual machines, known as simulators. Each LSL script receives a time slice of total time allocated to scripts by the simulator. Scripts are also assigned its own limited 16K memory space, containing script's bytecode, heap and stack ("LSL Script Memory," n.d.).

Objects in SL communicate via chat channels commonly used by players to chat. In inter-object communication, one object transmits, while others listen. Objects can use any channel number integers between -2147483648 to 2147483647. Channel 0 is used for local chat for player characters, therefore objects using channel 0 can say and hear things from player characters.

LSL has severe limitations such as not supporting object-oriented programming, dynamic memory allocation, and memory constraints. Until recently, Second Life only supported memory constraint of 16K for scripts, limiting the potential size and growth for the scripts. To overcome this issue, we decided to use linked message to communicate between scripts with each script having only some of the variables. Scripts listen for linked messages from other scripts in a comma separated form, which breaks the messages into function call and data, which is to be used in the function.

Recently, Second Life added the Mono scripting engine to LSL, increasing stack size to 64K and adding support for dynamic memory allocation ("LSL Script Memory," n.d.). However, although Mono scripting engine supports other languages, Second Life

did not add support for them. Having the 64K stack size allowed more space for the scripts and dynamic memory allocation solved our memory issues.

Section 2: Financial Literacy Game

What is Financial Literacy Game?

Financial Literacy Game is a 3-D online multi-player educational game that teenagers can play in the teen grid of Second Life virtual world. The goal of the game is to teach financial literacy to teenagers. The game enables them to make financial decisions and manage their finance in a virtual world, where they can safely experiment and see their results of their decisions. It is important to promote financial literacy to teenagers since they have little experience with financial decisions at their age. It is crucial not to overwhelm or confuse them in the process. The game enables the players to experience a virtual life time where they can make life decisions and see the results of their decisions. The game takes place after graduating from high school, players can choose to either attend college or immediately start work. Players have the opportunity to buy or rent various cars and houses with different price ranges. In addition to these, players can also save for retirement and invest in 401(k). Financial Literacy Game keeps track of financial status of players and provides them with feedback.

Financial Literacy Game aims to teach financial literacy to teenagers by simulating a virtual lifetime in a fast-paced virtual world. By simulating in a much faster pace, players can go through virtual lifetimes, experiment with life decisions and see consequences of their decisions and actions in a safe virtual world. After experiencing

various bad and good financial decisions in the virtual world, players can learn from their experiences and mistakes and avoid decisions that affect them negatively in real life.

As mentioned before, players start the game after graduating from high school, where they have starting money but no car or house. Starting money was set as $10,000 in the first development phase of the game, but was determined as $2,000 in the second development phase. Amount of starting money forces the players to be careful about their financial decisions to avoid going bankrupt early in the game. After choosing whether to go to college or immediately start working, they need to purchase a car, then a house. There are various options available to players when purchasing a car or house, depending on their credit score and financial status. At any time, players can choose purchase a new car and buy or rent a new house. The game also supports multiplayer features to keeps players engaged, enabling players to purchase a car or house from another player.

Figure 5. Players can choose to finance or pay cash for their cars in the car dealership.

Depending on whether a player chooses to go to college or start working, there are multiple life paths available. These life paths have unique salaries, promotions, and obstacles. Each path contains quests that support interactivity and helps to expand knowledge. In addition, players will encounter random life events that simulate real life events.

Financial Literacy Game offers different life paths for players. At the start of the game, players must choose to either go to college or start working. If players choose to go to college, they will be asked to choose from different career options, such as architect, video game designer, lawyer, teacher, doctor, nurse, and accountant. Each of the career choices includes various quests that keep the players engaged, while learning about financial choices. For example, in order to graduate from college, an architect needs to complete course assignments. For this, he needs to find a drafting table he can work on and no other player is currently using, find architecture books to gain knowledge, and finally, complete the drawing. The player may not succeed at the first try and might need extra help to complete the assignment.

If players choose to go to work instead of college, they are presented with different list of career choices. Among the choices are restaurant server, salesperson, construction worker, police officer. Players will still be required to do quests for their chosen career. For example, construction worker is required to complete a building. To accomplish this, he needs to purchase construction tools from the mall. There are different levels of quality in the construction tools, and player must choose tools that he can afford, and finally complete the building.

The game includes various events that occur unexpectedly. The type of random event depends on the current age and career of the player. For example, player in the college might experience that his research is going too well, but he needs to travel abroad which costs him money or player working as a construction worker may need to join a union but pay dues. Unexpected events also include car repairs and marriages. Depending on the type of car player chose to buy, it might need more repairs then a quality buy expensive car.

Figure 6. Construction area for the construction worker career path.

If the financial health of player is too low, they have to file bankruptcy. Filing bankruptcy causes players to lose their possessions and ends the game.

Figure 7. Players can choose to rent or buy a house in the housing development area.

Players needs to have HUD (Head-Up Display) equipped to play the game. HUD can be acquired by clicking on the sign at the starting position of the game.

HUD provides controls and important information for the players.

1. Cash: Account balance.
2. Credit Score: Similar to real life score. Based on debt, payment punctuality, recent activity, credit history and type of credit.
3. Age: Current age of the player. Ranges from 18 to 65. After 65, player retires and game finishes.
4. Speed: Speed of the game, from 1x to 6x. Default is 3x.

If pressed on the ‘More Info’ button, it will display further information.

1. Date: Current date (month and week) for the game.

2. Mortgage/Rent: Player's monthly rent or mortgage payments.
3. Car Payment: Player's monthly car payment.
4. 401K Balance: Balance of player's 401K.
5. Paycheck: Player's 2-week paycheck.
6. Home Equity: The Equity accumulated while player own a house. This money can be applied as a down payment for the next house or added to your cash balance if player chooses to rent later.
7. Student Loan Payment: Monthly payments with interest after graduating from college.

Figure 8. Player in the starting hub with his HUD equipped.

Objectives button provides the player with a window that shows his or her previously completed tasks and current task.

Players can choose to stop the game and log out of Second Life at any time. The game saves the current progress of the player and resumes when they log back in at a later time.

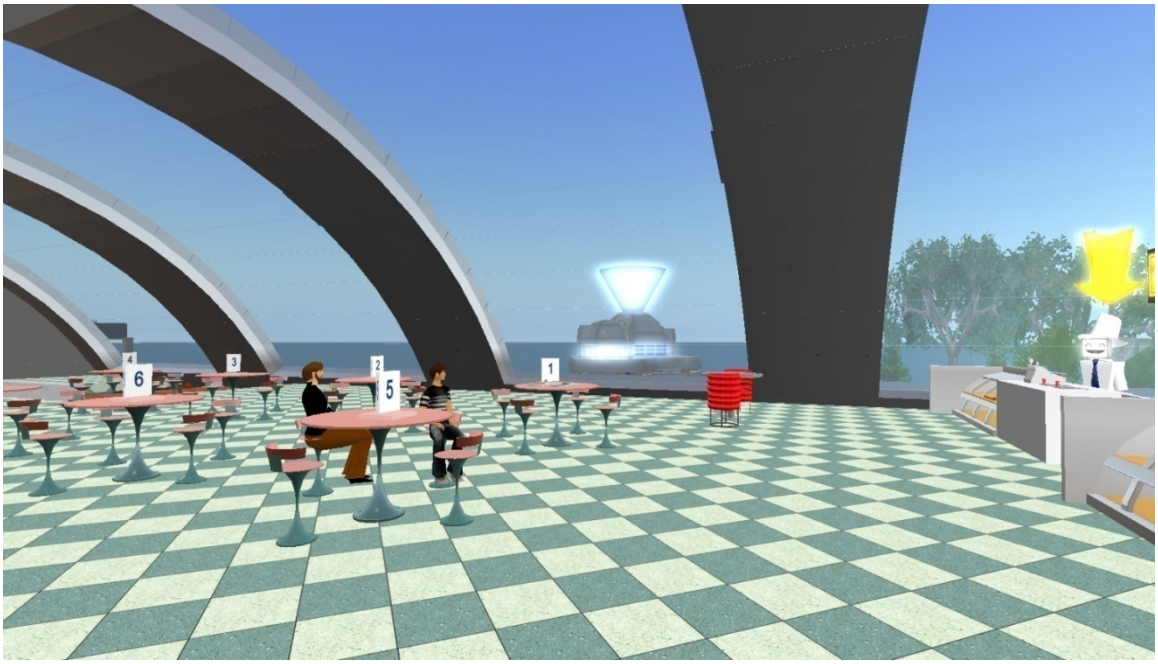

Figure 9. Quest for the restaurant server career path.

The game was deployed to high schools and university for testing and evaluation. Experiment testing and results for the Financial Literacy Game will be discussed in Chapter 5.

Development Approach

Financial Literacy Game was developed using feature-driven agile software development practices. Many of features of FDD were adapted to design and develop the Financial Literacy Game.

Development of the game was divided into two phases. First phase focused on laying out the initial framework of the game. It consisted of financing a car, buying or renting a house, going to college, and investing in retirement. In the second phase, the focus was refining the phase one features, while improving the game with interactive quests to engage players. Phase two features consisted of house and car quests, which requires players to learn about financial choices and answers related questions. Completing these quests offers various bonuses in the game. Furthermore, it includes college and work paths with unique and individual quests, salaries, random events and promotions, and a new dialog system.

In order to start the development process, multi-disciplinary team, consisting of financial and educational domain experts, instructional designer, art designer, programmers and project managers were selected.

The multi-disciplinary development team began the development with a kick off meeting to establish an overall model for the game. Domain experts provided user stories, which provided the developers to build a model that outlines the game. This model provides a visual guide to the development process and shows the overall project. It is important that the model to be understood by the multi-disciplinary team to provide shared understanding for the project. Use cases, functional and non-functional specifications and requirements from the customer are essential when building the overall model. The model was updated in each iteration with new features.

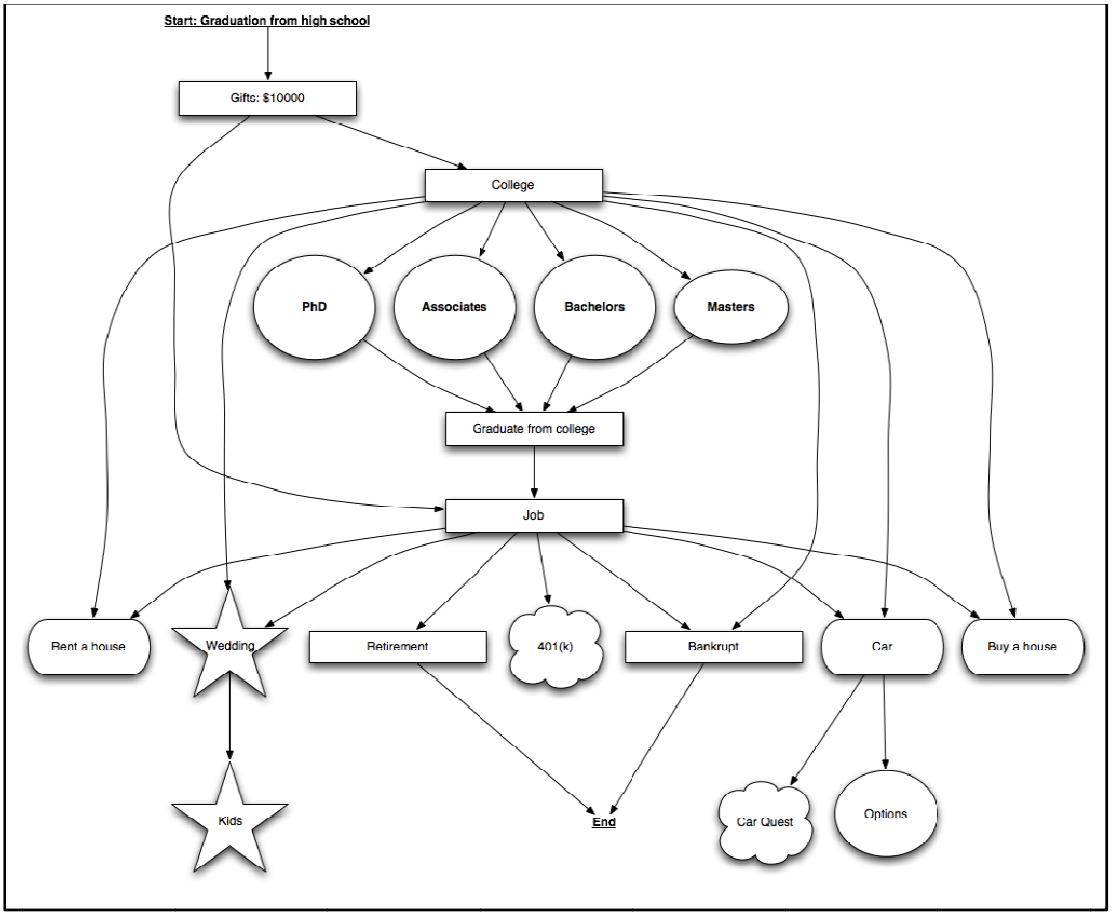

Figure 10. Model for the phase 1 of the Financial Literacy Game (Designed by the researcher, Jarrod Reuter, and Scott Moriarty).

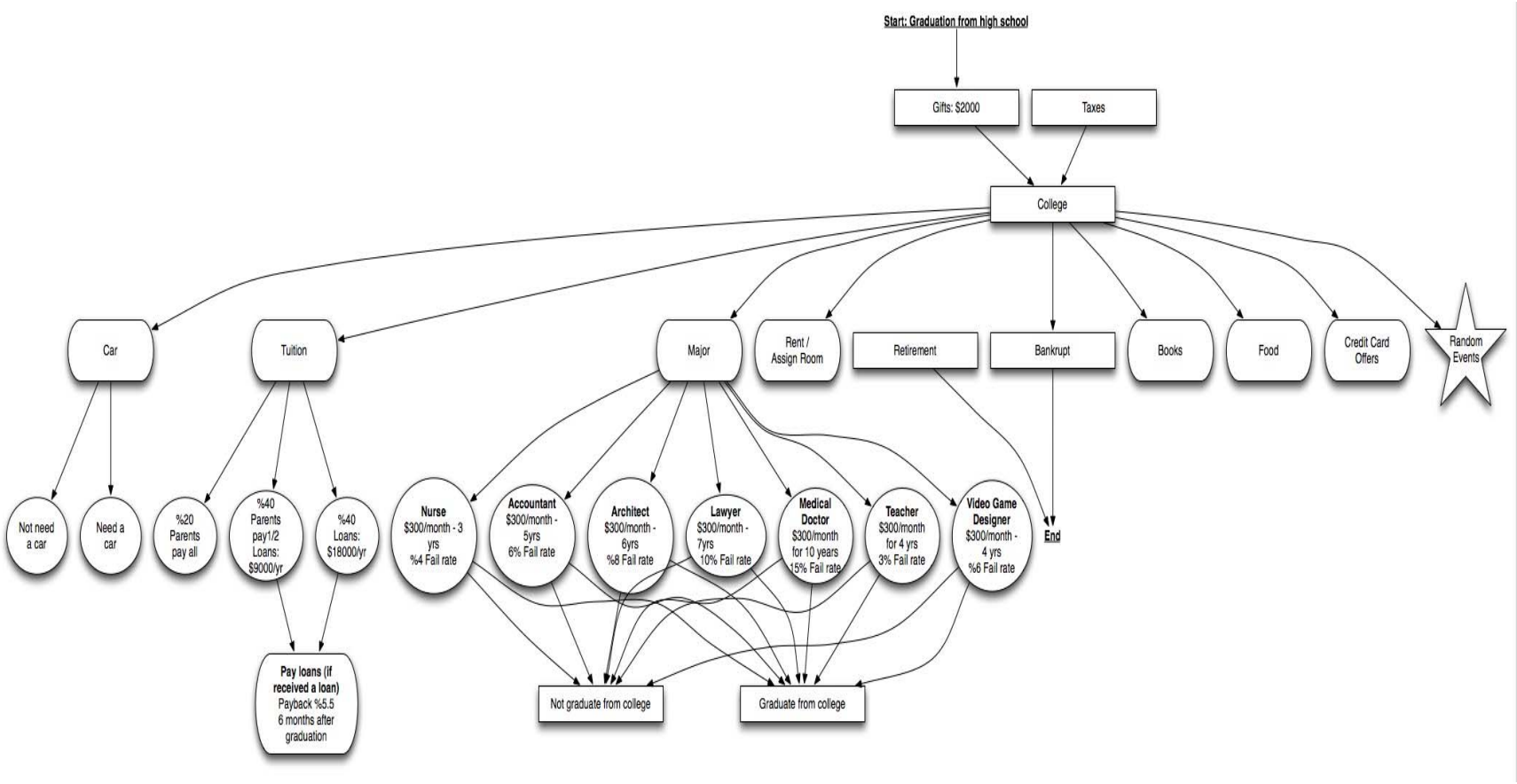

Figure 11. Model for the college life path for phase 2 of the Financial Literacy Game (Designed by the author, Jarrod Reuter, and Scott Moriarty).

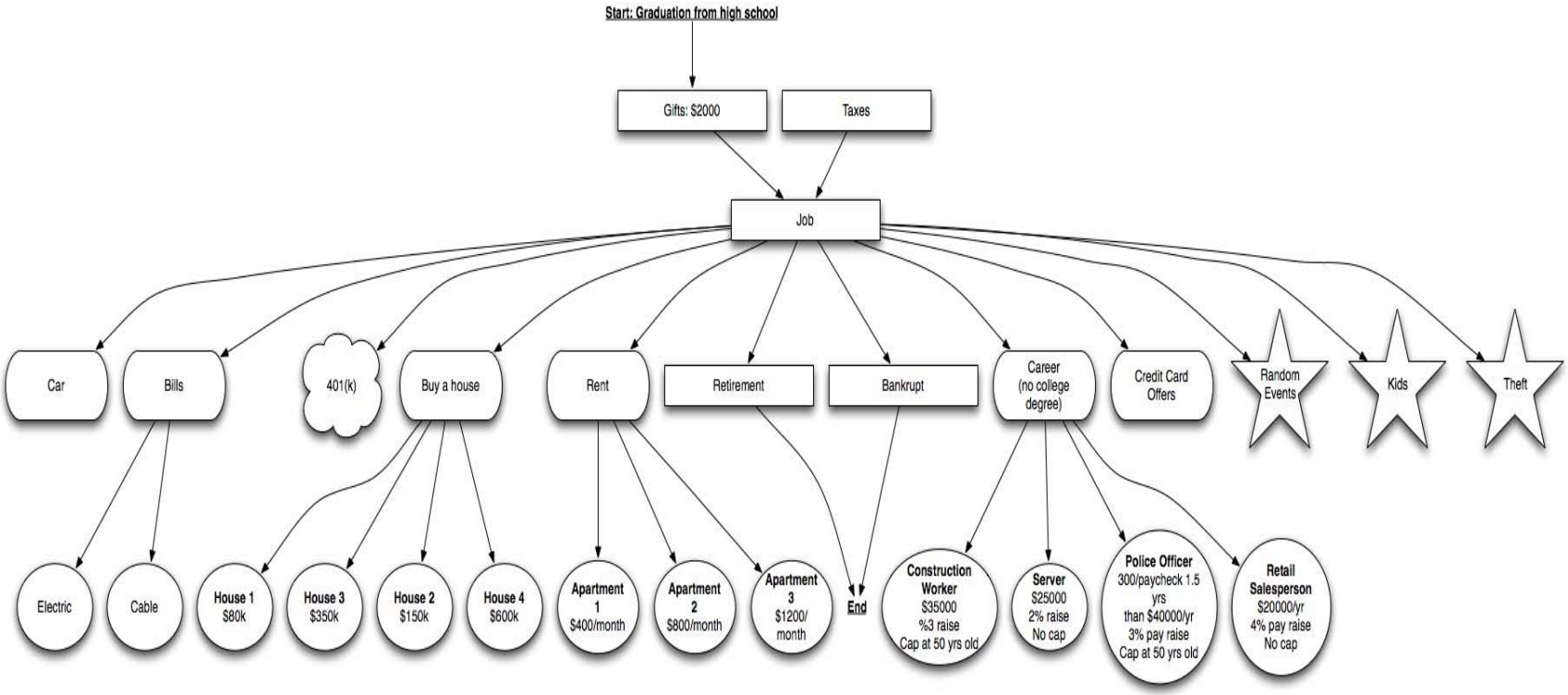

Figure 12. Model for the work life path for phase 2 of the Financial Literacy Game (Designed by the author, Jarrod Reuter, and Scott Moriarty).

Next process is building a detailed features list. Specifications and requirements are provided by the customer in the previous phase as user stories and object model is built with the requirements. It is vital for the multi-disciplinary team to have a shared understanding of the basic scope of the project to develop the feature list. From the object model and customer requirements, features were identified. Each feature is small and should not take more than two weeks to complete. If they are bigger, it is decomposed to smaller features. Most of the important game metrics, such as starting money, and loan rates, were discussed and defined in this stage. This ensures that the game can be built based on these values. We used an internal wiki to document the project documentation containing the object models, features list, list of team members, and assigned work for the developers, customer feedback and game instructions.

Typical FDD structure is used to identify features, which is "<action> <result> <object>" (Palmer & Felsing, 2002, p. 57).. For example, buying a car or house from another is identified as *Buy* a *car from another player* and *Buy* a *house from another player*. These similar features are grouped into *feature sets* and are in structure "<action>-ing <business deliverable> <by|of|to|from|for|between> a(n) <object>" (Anderson, 2004, p. 155).. The example features are grouped into *Creating multi-player interactions between players* feature set. Finally, features and feature lists are organized into subject areas, which is typically in the form of "<object> management" (Anderson, 2004, p. 156). The example features and feature list will be represented by the *Player* management sub-system, which makes it easier to identify player-related components.

After the feature list is completed, the development team proceeded to prioritize and order the features to be implemented. The order is based on various factors such as complexity of the feature, work load of developers and dependencies between features. Another important factor to consider was the classroom use and how students can benefit from a feature, which gave features involving vital financial decisions priority. The development team assigned features for each developer and tentative due dates for each feature. Each developer was assigned specific parts of the project that they were familiar with. This ensured high familiarity and expertise for developers and especially helped when there were dependencies among codes written by developers.

Next phase is the construction phase. In the first part of this phase, project managers assigned features that are to be developed in an iteration. The features were refined and the overall model was updated. In the second part, developers wrote the code for their assigned features. Testing was performed after finishing a feature. If the features performed correctly, it was promoted to a build and deployed to the game.

Section 3: Second Life for Statistics Homework Sessions

It is usually expensive and time consuming to have supplementary in class exercises that are designed to help students understand abstract concepts by using concrete examples. Schwerha, Liu, Ozercan, Vadlamani and Neiman (2008) states that teachers have control over the in-class learning experience while the work outside of course is not regulated. Feedback is important in the learning process and insufficient feedback can lead to frustration, inefficient learning and incorrect knowledge (Schwerha, et al., 2008). By using Second Life as an alternative to traditional in-class exercises,

associated costs were minimized, learning time was maximized, and students were motivated to learn abstract statistical concepts and provide immediate feedback to students.

The study was applied in a large *Midwestern university* Industrial and Systems Engineering (ISE) course teaching engineering probability and statistics. The topics for the class include hypothesis testing, analysis of variance, statistical power, confidence intervals, types of error, and simple and multiple linear regression.

Students were required to evaluate usability of cell phones by making statistical inferences. The topics covered were single sample hypothesis testing and types of statistical error.

From the class of 36 students, 10 students volunteered for the Second Life exercise. Students were divided into teams of three and teams were designated colors (green, red, and blue). They were given instructions for the game and definitions of statistical terms used in the game. The exercise consisted of two parts.

In the first part, students worked in groups. Students were given virtual cell phones in their team area and data sets associated with finding an address or dialing a number. Students began the game by clicking on the start button, and were asked questions. They were instructed to discuss with their team members and answer the questions. To ensure participation of all team members, each correct answer gave them one point and each incorrect answer them deducted one point from their group score.

Figure 13. Data set for the one of the four virtual phones.

Figure 14. Students performing the team exercise.

In the later part, students were required to answer individually in modified Groupthink exercise, which was originally developed in Massachusetts Institute of Technology (MIT). Students were asked to gather at the tables for their team color. They were asked questions and had to answer them individually. Each correct answer gave them five points, and if all the team members got the question correctly answered, they were given an additional point. At the end of the exercise, their total team score was calculated.

Figure 15. Students engaged in the modified Groupthink exercise.

Scoring for the exercise favored group participation and learning. Performance of students performing the traditional homework assignments versus Second Life exercise were compared, and will be discussed in Chapter 5.

CHAPTER 5: EXPERIMENT TESTING AND RESULTS

This chapter discusses the deployment of the educational games to high school and university students. The chapter begins by discussing the details of the deployment, and continues with presenting the results and discussing the feedback.

Section 1: Financial Literacy Game

To evaluate the learning effectiveness and usability of the Financial Literacy Game, the game was deployed in *School One* and *School Two* high school class sessions and in a *Midwestern university* student seminar.

Financial Literacy Testing

In order to evaluate the learning effectiveness in financial literacy, the game was deployed in *School One* and *School Two* high school class sessions.

Procedure

Each student participated by playing the Financial Literacy Game. Second Life was already pre-installed on the computers in schools' computer lab. Appropriately configured notebook computers were provided, if needed. All the computers were capable of running the game with decent frame rates and sufficient network performance, therefore performance was not an issue. Students were provided with the credentials to log in to the virtual world.

Students started the game by equipping the heads-up display (HUD) required to play the game. Like mentioned before, HUD contains the required controls to play the game.

Before beginning the game, students can optionally participate in a tutorial to familiarize themselves with Second Life and the game locations.

At the beginning of the game, students are required to decide whether to attend college or start working. Depending on their choice, they need to decide on their careers.

First activity they need to do is to buy a car. To buy a car, they visit the car dealership. In order to guide them purchasing a car, there is an educational video that takes them through a car purchase process in real life. The video shows a typical real life situation they may have to experience in real life, and lets them know of important information in this financial decision. In the game, players have to decide on the payment choice when purchasing a car; they can pay cash, choose monthly payments, or do the car quest, which gives them discounts. In car quest, students are given a lesson on car purchase process and they are asked questions about it. Each correct answer gives them discounts for the in-game cars. Questions for the car quest can be found in the Appendix A.

After purchasing a car, they need to buy or rent a house. To buy or rent a house, they visit the housing development office. They can watch a video to guide them on how to buy a house in real life. This video also shows a typical real life situation, they may have to experience. In the game, students face another financial decision to whether to buy or rent a house with different price points, or they can do housing quest to get them discounts. Similar to car quest, housing quest provides them with a lesson on how to purchase a house and asks the students questions about the lesson. Each correct answer

gives them discount for the in-game houses. Questions for the housing quest can be found in the Appendix A.

After these required activities, students can live their lives as the career they have chosen and perform the quests associated with their career choice. They can also save for retirement by investing in 401(k).

The development team keeps the high scores of the players in the game, and players can compare their in-game scores. After completing the game session, participants were given post-test.

Results

Students were given pre-test questions before playing the Financial Literacy Game and post-test after playing the game. Total number of participants was 35 for *School One* and 9 for *School Two*.

First financial literacy testing was done in *School One* in May 20, 2009, among 35 high school students, ranging in age from 15 to 18. Appendix A shows the pretest and posttest that were given to the participants after playing the game. The classroom was for computer programming and the students had previous experience programming for Second Life. Results were not significantly different between the scores. The players were focused on the game programming aspect rather than financial education. For the post-test, two students were absent in the classroom, therefore they were removed from the results. Figure 19 shows the mean value of number of questions that were correctly answered by all students that took the test.

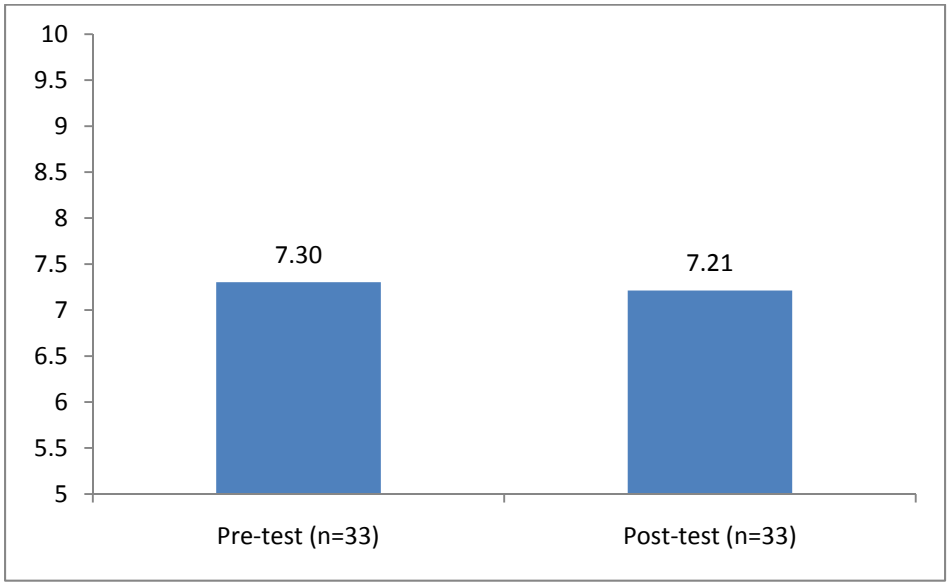

Figure 16. Number of questions students correctly answered (*School One, Data provided by Dr. Teresa Franklin and Dr. LiWei Peng, 2009)*

Table 2: Statistics of students' performance (*School One*, *Data provided by Dr. Teresa Franklin and Dr. LiWei Peng, 2009)*

	N	Mean	Std Dev	Std Error Mean
Pre-Test	33	2.65	0.156	0.027
Post-Test	33	2.61	0.123	0.021

The second financial literacy testing was conducted in *School Two* in May 27 2009, among 9 high school students, ranging in age from 15 to 18. Appendix A shows the pretest and posttest that were given to the participants after playing the game. The classroom used for testing was a finance class and students had some prior knowledge of the financial content that is covered in the game. Students were focused on learning finance. It can be seen that student's scores improved after playing the game. Figure 20 shows the mean value of number of questions that were correctly answered by all students that took the test.

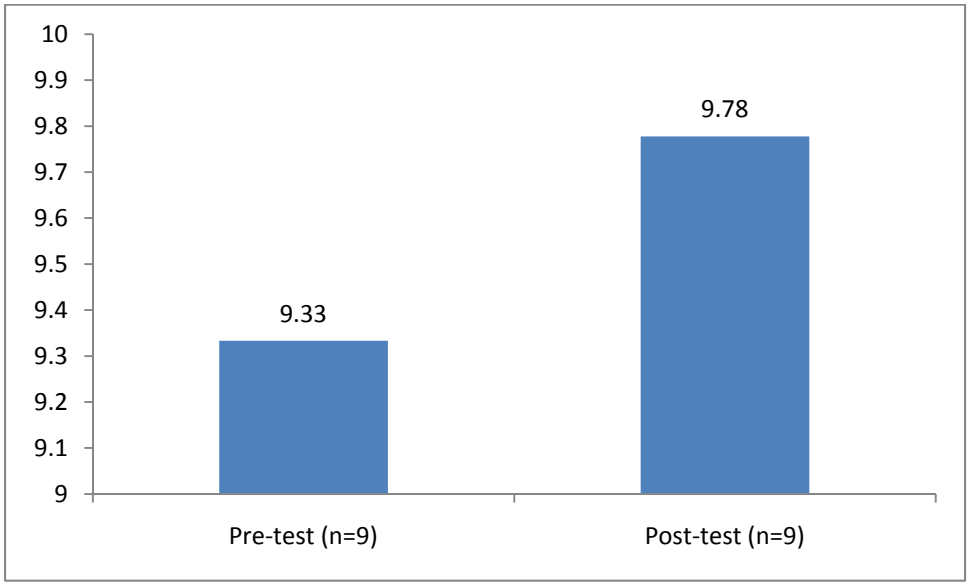

Figure 17. Number of questions students correctly answered (*School Two*, *Data provided by Dr. Teresa Franklin and Dr. LiWei Peng, 2009)*

Table 3: Statistics of students' performance (*School Two*, *Data provided by Dr. Teresa Franklin and Dr. LiWei Peng, 2009)*

	N	Mean	Std Dev	Std Error Mean
Pre-Test	9	2.47	0.05	0.017
Post-Test	9	2.5	0.087	0.029

Usability Testing

In order to evaluate the usability of the Financial Literacy Game, the game was deployed in a class session in *School One* and financial literacy student seminar in a *Midwestern University*. Deployment in the *School One* used the first development phase and deployment in a *Midwestern University* used the second development phase.

Procedure

Procedure was similar to financial literacy testing. Students went through the same activities such as buying a car, buying or renting a house, and investing in

retirement. Testing in *School One* was for the first development phase of the game. Feedback from *School One* was used to improve the game in the second development phase. After finishing the game session, participants were given survey forms to complete, the forms can be found in Appendix A.

User Feedback and Observations

Usability testing was completed with first phase of the game development in *School One* and second phase usability testing was done in a *Midwestern university*. Feedback was received from students about the technical or game play problems to improve the game further.

Among the feedback received for phase one development were, loading time of the game components, difficulties finding the starting position in order to get a HUD, and vehicles getting stuck in buildings. Some of the students were confused on navigating in the 3-D virtual world and indicated that the in-world map did not help.

The development team observed the students while they were playing the game. Some students did not read the game instructions or watched the educational videos but they enjoyed exploring the world and asked for help when they needed. After watching the students play the game and buy expensive vehicles and houses, it is a good idea to have better in-game indicators to let the players know that they are going bankrupt.

Results

The first usability testing was completed in *School One* in February 6, 2009, among 54 high school students, ranging in age from 15 to 18. Appendix A shows the survey form that was given to participants after playing the Financial Literacy Game. In

part 1, five point Likert Scale was used, 5 = ‘strongly agree’, 4 = ‘agree’, 3 = ‘neutral’, 2 = ‘disagree’ and 1 = ‘strongly disagree’. Figure 21 below shows the results for the first part of the survey.

It can be concluded that most participants believed that learning finance through a 3-D virtual world increased their finance knowledge and was useful from positive response to question, ‘I think playing the lesson in the simulation increased my content knowledge of finance’ and negative response to question, ‘I think the game is a waste of time for learning finance’.

Some of the students had problems with the navigating in the game from the response to question, ‘I had difficulty in navigating the game’. The development team felt that this issue was important and should be addressed in the second development phase. To address this, designers worked on a new island layout that is easy to navigate, especially with vehicles, and various important locations in the island were conveniently accessible through teleporter objects placed in-world.

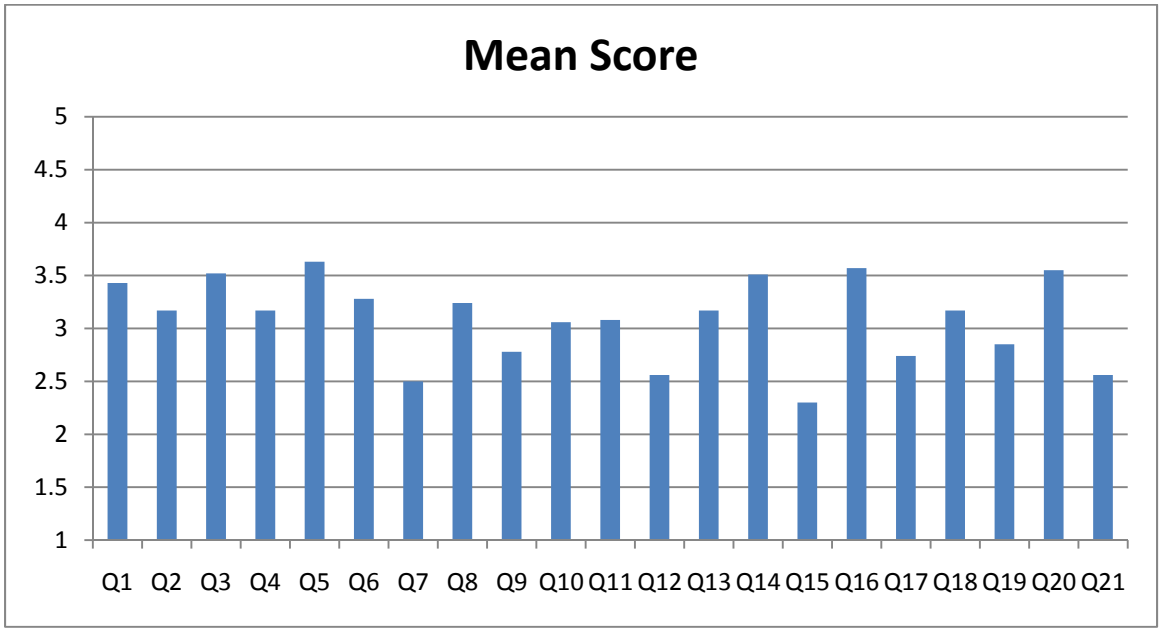

Figure 18. Mean scores for usability testing indicating agreement with game structure (*School One, Data provided by Dr. Teresa Franklin and Dr. LiWei Peng, 2009)*

Part 2 used Likert scale from 1 to 5 with 5 = 'strongly satisfied', 4 = 'satisfied', 3 = 'no opinion', 2 = 'dissatisfied' and 1 = 'not satisfied'. It can be concluded that students enjoyed the game session and were satisfied with the 3-D virtual world from positive responses to question, 'How satisfied are you with this new way of learning as compared to when your teacher did not use it?'. Figure 22 below shows the results for the second part of the survey.

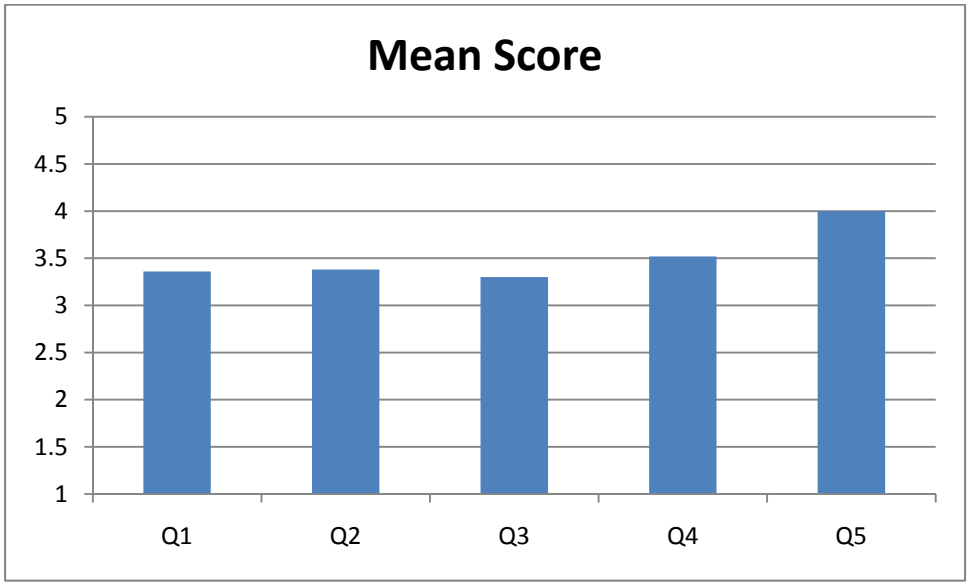

Figure 19. Mean scores for usability testing indicating satisfaction of game play (*School One, Data provided by Dr. Teresa Franklin and Dr. LiWei Peng, 2009)*

Second usability testing was done in a Midwestern university on March 12, 2009 for a financial literacy student seminar. There were 17 participants, including sophomores, juniors, seniors and professors. Appendix A shows the survey form that was given to participants after playing the Financial Literacy Game. In part 1, five point Likert Scale was used, 5 = 'strongly agree', 4 = 'agree', 3 = 'neutral', 2 = 'disagree' and 1 = 'strongly disagree'. Positive responses were received from most participants. Most participants believed that learning finance through a 3-D virtual world increased their finance knowledge and was useful from positive response to question, 'I think playing the lesson in the simulation increased my content knowledge of finance' and negative response to question, 'I think the game is a waste of time for learning finance'. Figure 23 below shows the results for the first part of the survey.

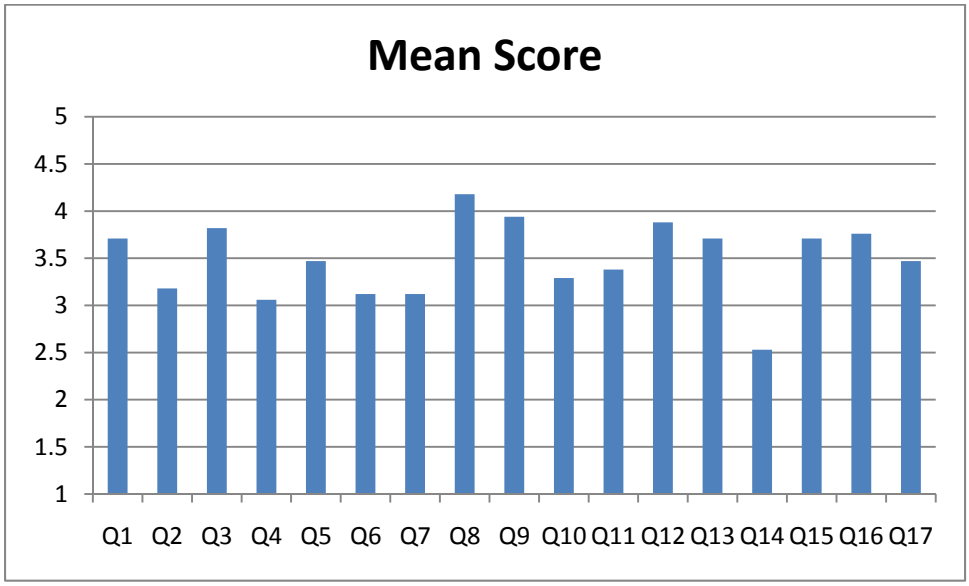

Figure 20. Mean scores for usability testing indicating agreement with game structure (*Midwestern University, Data provided by Dr. Teresa Franklin and Dr. LiWei Peng, 2009)*

Part 2 used Likert scale from 1 to 5 with 5 = 'strongly satisfied', 4 = 'satisfied', 3 = 'no opinion', 2 = 'dissatisfied' and 1 = 'not satisfied'. For part 2, similar positive responses were received. Most participants were satisfied with the 3-D virtual world, and gave positive responses to question, 'How satisfied are you with this new way of learning as compared to when your teacher did not use it?' Figure 24 below shows the results for the second part of the survey.

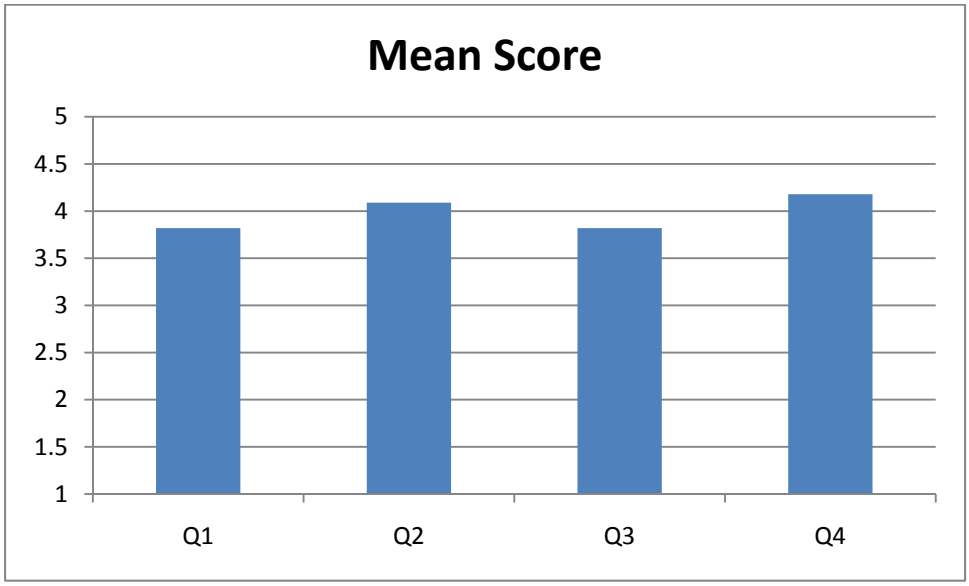

Figure 21. Mean scores for usability testing indicating satisfaction level of game play (*Midwestern University, Data provided by Dr. Teresa Franklin and Dr. LiWei Peng, 2009)*

Section 2: Second Life for Statistics Homework Sessions

The project was deployed in a *Midwestern University*, Industrial and Systems Engineering (ISE) course teaching engineering probability and statistics. The topics for the class include hypothesis testing, analysis of variance, statistical power, confidence intervals, types of error, and simple and multiple linear regression. To evaluate learning effectiveness, two sessions were conducted. There were 10 participants for the Second Life project. Appendix B shows the survey form that was given to participants after completing the session.

Results

After the students performed the homework sessions, students took the tests for the course. Figure 25 below shows the average test scores of students who performed the

Second Life homework sessions and regular homework. It can be seen that there were differences in exam performance between students who used the Second Life exercise and those who did not.

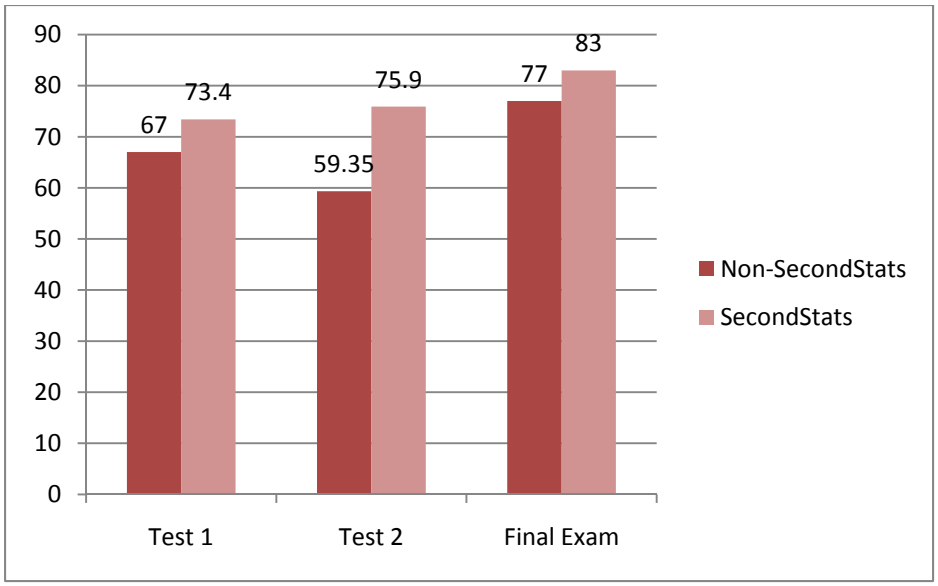

Figure 22. Test scores for students (*Data provided by Dr. Diana Schwerha, 2008).*

All students completed a survey after finishing the Second Life homework session. The survey can be found in Appendix A. In the survey, five point Likert Scale was used, 1 = 'strongly disagree', 4 = 'disagree', 3 = 'N/A', 2 = 'agree' and 1 = 'strongly agree'. Figure 26 shows the median values for the survey.

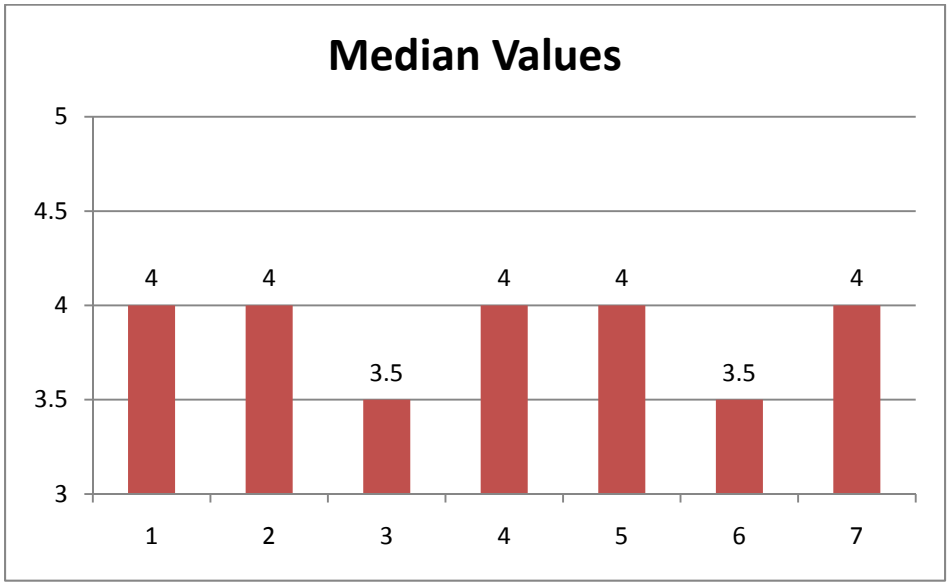

Figure 23. Median values for the survey (*Data provided by Dr. Diana Schwerha, 2008).*

According to the survey results, students think that feedback is important when doing homework exercises and Second Life exercise contributed to their learning of the material efficiently. Furthermore, students enjoyed working in teams and collaborating with their classmates.

Feedback

In Second Life homework session, students were provided feedback in various forms. Feedback was provided after answering team questions in the first part. All team members were provided with the explanations with the correct answer for the question, which team members can discuss in their team chat. Feedback was also provided via statistics glossary and definitions provided in their team area. Students were able to look up relevant statistical terms in the game.

After the game, students provided feedback about the game. They found it hard to communicate in teams of four people, since not all of the team members knew each other. Teams of two people could have worked better.

Another aspect that students did not like was the two hour limit on the homework session, especially when compared to days where they can work to complete the assignment. It might be a good idea to allow for larger time span for homework sessions.

CHAPTER 6: CONCLUSION

This thesis has described a research project that uses feature-driven development, an agile software development methodology, in development of educational games in 3-D virtual worlds. Feature-driven development brings improvements to agile software development when developing educational games in a development environment with frequent changes and small, multi-disciplinary team.

By adapting feature-driven methodology, the researcher was able to design and develop educational games with a small, multi-disciplinary team. The educational games developed increases students' motivation for learning and improves learning effectiveness. The experiment research was presented using two educational games that were developed in 3-D virtual worlds.

The first project, a financial literacy game, aims to teach financial literacy to teenagers by using a 3-D virtual environment to simulate their personal finance. The game takes players from high school graduation to retirement and teaches them about the real life financial decisions. The game was deployed to high schools for testing and evaluation. Participants of the study were presented with usability surveys and financial literacy test.

The second project, Second Life for Statistics Homework Sessions, is aimed at university students to understand the abstract statistical concepts by providing real life examples, and using collaboration to motivate students to work on their homework problems. The virtual homework session was deployed in a *Midwestern University* classroom. Participants of the study were presented with a survey form.

Test results indicate improvements in understanding for the domain concepts and improvements in test scores. Furthermore, survey results show that students believe these educational games can motivate students and contribute to learning.

Future work in this area of research is to continue testing and evaluation with larger sample sizes. In addition to this, comparison of traditional and feature-driven methodologies with separate development teams should be researched to understand the improvements in software development.

REFERENCES

Abrahamsson, P. (2005). *Agile software development: Introduction, current status and future.* Retrieved June 28, 2009 from http://www.mit.jyu.fi/opetus/kurssit/jot/2005/kalvot/agile%20sw%20development.pdf

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods - review and analysis. Espoo, Finland:* VTT Publications.

Agile. (2009). In *Merriam-Webster Online Dictionary*. Retrieved December 17, 2009, from http://www.merriam-webster.com/dictionary/agile

Ambler, S. (2009). *The XP project lifecycle.* Retrieved Jan 6, 2010, from http://www.agilemodeling.com/essays/agileModelingXPLifecycle.htm

Anderson, D.J. (2004) *Agile Management for Software Engineering - Applying the Theory of Constraints for Business Results*, New Jersey, USA: Prentice Hall.

Bain, S. L. (2008). *Emergent design: The evolutionary nature of professional software development* (1st ed.) Boston, MA: Addison-Wesley Professional.

Baskerville, R., Travis, J., & Truex, D. P. (1992). Systems without method: The impact of new technologies on information systems development projects. *Proceedings of the IFIP WG8.2 Working Conference on the Impact of Computer Supported Technologies in Information Systems Development,* 241-269.

Beck, K., & Andres, C. (2004). *Extreme programming explained: Embrace change* (2nd ed.) Boston, MA: Addison-Wesley Professional.

Bethke, E. (2003). *Game development and production.* Wordware Publishing, Inc.

Bizzocchi, J., & Paras, B. (2005). Game, motivation, and effective learning: An integrated model for educational game design. *The International DiGRA Conference, June 16th - 20th, 2005, Vancouver, British Columbia, Canada.*

Boehm, B. (2002). Get ready for agile methods, with care. *Computer, 35*(1), 64-69.

Building a better bug-trap (2003, June). *The Economist*, *367*, 18.

Chan, T. S., & Ahern, T. C. (1999). Targeting motivation - adapting flow theory to instructional design. *Journal of Educational Computing Research, 21 (2)*, 151-164.

Charette, R. N (2005). Why software fails. *IEEE Spectrum,* 42, 42-49.

Coad P. (1988). DOD-STD-2167, defense system software development: Point, counterpoint, and revision A. *Proceedings of the Computer Standards Conference,* Washington, DC. 47-50.

Coad, P., Luca, J. d., & Lefebvre, E. (1999). *Java modeling color with uml: Enterprise components and process with cdrom*. Upper Saddle River, NJ: Prentice Hall PTR.

Cockburn, A., & Highsmith, J. (2001). Agile software development: The people factor. *Computer, 34*(11), 131-133.

Constantine, L. L. (1995). *Constantine on Peopleware*, Yourdon Press.

De Luca, J. (2003). *Issue 4 - Q&A about feature milestones.* Retrieved August 28, 2009, from http://www.featuredrivendevelopment.com/node/548

Deci, E.L. & Ryan, R.M. (1985). *Intrinsic motivation and self-determination in human behavior (perspectives in social psychology).* New York: Plenum.

Dede, C., Clarke, J., Ketelhut, D., Nelson, B., & Bowman, C. (2005). Fostering motivation, learning, and transfer in multi-user virtual environments. *American Educational Research Association Conference,* Montreal.

Delwiche, A. (2006). Massively multiplayer online games (MMOs) in the new media classroom. *Educational Technology & Society, 9*, 160-172.

Second Life Destination Guide. (n.d.). Retrieved Feb 12, 2010 from http://secondlife.com/destinations

Fowler, M. The new methodology (2005). Retrieved June 24, 2009, from http://www.martinfowler.com/articles/newMethodology.html

Gibson, A., & Sherba, A. S. (2008). Agile game development and fun. (Bachelor's thesis. University of Colorado).

Glass, R. L. (2006). The Standish report: Does it really describe a software crisis? *Communications of the ACM, 49*(8), 15-16.

Good, J. M. (2003). A pragmatic approach to the implementation of agile software development methodologies in plan-driven organisations. (Honours Dissertation, Lincoln University).

Hibbs, C., Jewett, S., & Sullivan, M. (2009). *The art of lean software development: A practical and incremental approach.* O'Reilly Media, Inc.

Highsmith, J. (2002). *Agile software development ecosystems*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Highsmith, J., & Fowler, M. (2001). The agile manifesto. *Software Development Magazine, 9*(8), 29-30.

Jeffries, R. E., Anderson, A., & Hendrickson, C. (2000). *Extreme programming installed*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Khramthchenko, S. (2005). A project management application for feature driven development (FDDPMA). (Master's Thesis, Harvard University).

Lindvall, M., Basili, V. R., Boehm, B. W., Costa, P., Dangle, K., Shull, F., et al. (2002). Empirical findings in agile methods. *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - XP/Agile Universe 2002,* 197-207.

Long, J. (2003). Why not have fun while learning: Using programming games in software programming education. Dallas, 23.

LSL Portal. (n.d.). Retrieved from the Second Life Wiki: http://wiki.secondlife.com/wiki/LSL_Portal

LSL Script Memory. (n.d.). Retrieved from the Second Life Wiki: http://wiki.secondlife.com/wiki/LSL_Script_Memory

Mandel, T. (1997). *The elements of user interface design*. New York, NY: John Wiley & Sons, Inc.

McFarland, I. (2006). *Agile practices on real-world projects.* Retrieved May 19, 2009, from http://javamug.org/mainpages/presentations/AgileDevelopmentatGoogle-DallasJUG.pdf

Michael, D. R., & Chen, S. L. (2005). *Serious games: Games that educate, train, and inform.* Muska & Lipman/Premier-Trade.

Moreno-Ger, P., Burgos, D., Martinez-Ortiz, I., Sierra, J. L., & Fernandez-Manjon, B. (2008). Educational game design for online education. *Comput.Hum.Behav., 24*(6), 2530-2540.

Nacke, L. (2005). Facilitating the education of game development. (Master's thesis, University of Magdeburg).

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Commun.ACM, 48*(5), 72-78.

Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to feature-driven development.* Prentice Hall PTR.

Petrillo, F., Pimenta, M., Trindade, F., & Dietrich, C. (2008). Houston, we have a problem...: A survey of actual problems in computer games development. *SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing,* Fortaleza, Ceara, Brazil. 707-711.

Reel, J. S. (1999). Critical success factors in software projects. *IEEE Softw., 16*(3), 18-23.

Reifer, D. J. (2002). How good are agile methods? *IEEE Software, 19*(4), 16-18.

Royce, W. W. (1987). Managing the development of large software systems: Concepts and techniques. *ICSE '87: Proceedings of the 9th International Conference on Software Engineering,* Monterey, California, United States. 328-338.

Russ, M. L., & McGregor, J. D. (2000). A software development process for small projects. *IEEE Software, 17*(5), 96-101.

Salo, O., & Abrahamsson, P. (2007). An iterative improvement process for agile software development. *Software Process: Improvement and Practice, 12*(1), 81-100.

Schwaber, K. (2004). *Agile project management with scrum*. Microsoft Press.

Schwaber, K., & Beedle, M. (2001). *Agile software development with scrum*. Upper Saddle River, NJ: Prentice Hall PTR.

Schwerha, D., Liu, C., Ozercan, S., Vadlamani T., & Neiman, L. (2008). Second life as a pedagogical tool for improving statistics homework sessions. *Proceedings of the American Association of Engineering Education (Zone 1) Conference,* West Point, NY.

Shine Technologies. (2003). *Agile methodologies survey results.* Retrieved June 24, 2009, from http://www.agilealliance.org/system/article/file/1121/file.pdf

Standish Group. (2004). *2004 chaos report.* Retrieved June 28, 2009, from http://www.standishgroup.com/sample_research/PDFpages/q3-spotlight.pdf

Theunissen, W. (2003). A case-study based assessment of agile software development. (Master's thesis, University of Pretoria).

Tran, M. Q., & Biddle, R. (2008). Collaboration in serious game development: A case study. *Future Play '08: Proceedings of the 2008 Conference on Future Play,* Toronto, Ontario, Canada. 49-56.

Vinekar, V., Slinkman, C. W., & Nerur, S. (2006). Can agile and traditional systems development approaches coexist? an ambidextrous view. *IS Management, 23*(3), 31-42.

What is Second Life?. (n.d.). Retrieved Feb 12, 2010 from http://secondlife.com/whatis

West, D. (2009). *Case study: Microsoft speeds tool delivery with agile development.* Retrieved May 16, 2009, from

http://www.microsoft.com/presspass/itanalyst/docs/03-27-09CSMicrosoftSpeedsToolDelivery.aspx

Williams, L., & Kessler, R. (2002). *Pair programming illuminated*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

APPENDIX A: QUESTION SETS FOR FINANCIAL LITERACY GAME

Usability Testing

Please write your answer in the corresponding column on the right.
Name
Grade
School
Gender male female
Date

We would like your opinion about the financial simulation! You may choose whether or not you fill out this survey. Your individual responses will be kept confidential. Your teachers, parents, and the other students will NOT see your survey answers. Completing the survey has NO effect on your grade. Thank you for giving us your opinion!

Part I.

Please circle ONE response for each statement.

(Strongly Agree/Agree/Neutral/Disagree/Strongly Disagree)

1.I think playing the lesson in the simulation increased my content knowledge of finance.
2.I think the different functions of the simulation were hard to play.
3.I understand the instructions in the lesson.
4.I am frustrated in playing the simulation.
5.The videos in the simulation helped my understanding of finance.
6.I had difficulty in navigating in the simulation.
8.I do NOT like playing the simulation.
9.The way the simulation looked appealed to me.
10.I found the lesson very complex.
11.Playing the lesson in the simulation motivated me to learn finance.
12.I did NOT learn anything from the lesson in the simulation.
13.This lesson took an appropriate amount of time.
14.I think that I need help in using this game.
15.I think that I would like to play this simulation again.
16.I think the lesson in the simulation is a waste of time for learning finance.
17.Overall, this lesson went well in the simulation.

Part II.

Please indicate how satisfied you were with the simulation that you played. Answer the following questions by crossing out the numbered box that most closely identifies how satisfied you were with playing the simulation:

Please mark an "X" for your response to each item. (Strongly Satisfied/Satisfied/No Opinion or Not Sure/Dissatisfied/Strongly Dissatisfied)

1.How satisfied are you that the simulation improved your financial skill?
2.How satisfied are you that the instruction you received in the simulation helped your understanding of finance.
3.How satisfied are you that the simulation helped you enjoy learning more about finance?
4.How satisfied are you that your finance grade will improve by playing the simulation?
5.How satisfied are you with this new way of learning as compared to when your teacher did not use it?

Part III

Please write your answers in the space provided following each question.

1.Please describe any problems you experienced related to the use of simulation.
2.Please describe any problems you experienced related to understanding of the content.
(Mortgage/House/Car/401K)
3.Please describe your impression of this lesson.
4.How could this lesson be improved?
5.Other comments?

Financial Literacy Testing

Pre/Post-Test Financial Literacy

First Name: _____

Class Period____ School_____

Please place the correct letter in the blank which best answers the question.

_____1. Which is these is NOT a financial institution?
A. Bank
B. Credit Union
C. Lawyer
D. Savings and Loan

_____2. Credit means _____.
A. Defending yourself in court
B. Signing a lease for an apartment

C. Borrowing money
D. Filing your income taxes

_____3. Financing means _____.
A. Making arrangements for a loan to buy something
B. Obtaining insurance for a car
C. Having the car's title changed to your name
D. Registering your car to get license plates

_____4. When a business or person does not have enough money to pay their bills, this is called_____?
A. Bailout
B. Bankruptcy
C. Stimulus
D. Credit

_____5. Which of these jobs requires a college degree?
A. Waiter
B. Construction Worker
C. Teacher
D. Janitor

_____6. Responsible young adults should buy the things they want and need using:
A. Money their parents continue to give them throughout their lives
B. Credit Cards.
C. Money they earn from working
D. Money from the government

_____7. Which of these means of purchasing involves borrowed money?
A. Checks
B. Cash
C. Credit Cards
D. Gift Cards

_____8. If you borrow money to buy a car, how does the lender know he will get back the money you borrowed?
A. Because you promised
B. If you don't pay, he will take your car
C. Your license will be revoked
D. He will send a gang member to beat you up

_____9. If you use a credit card (which is a loan) to make purchases, _____.
A. You must repay the loan on the next weekend
B. The money is taken from your paycheck
C. The money becomes a gift to you after 60 days
D. You must make a payment when you receive a bill

_____10. When you buy a car, payment is made
A. By saving enough money to pay for the car

B. Asking you family to give you their old car
C. Borrowing the money
D. Trading things that you already own

Car Quest Questions:

1. A cosigner is
   A. Someone who helps you find what car to purchase.
   B. A person who signs his or her name to a loan agreement, if the primary debtor does not pay, the cosigner is fully responsible for the loan.
   C. A company that assist you in your payments
   D. None of the above

2. Who typically offers the lowest APR rate when financing your car?
   A. The car dealer
   B. A commercial bank
   C. A credit union
   D. They always offer the same

3. All else equal what financing term would have the lowest monthly payments?
   A. 3 years
   B. 4 years
   C. 5 years
   D. They are all the same

4. How does a down payment when purchasing a car affect your monthly payments all else equal?
   A. It decreases the monthly payments
   B. It increases the monthly payments
   C. It does not change the monthly payments
   D. It is different for each bank

5. If you can afford a total of $500 a month for transportation what does that entail?
   A. Car payment
   B. Gas and Insurance
   C. Maintenance and repair
   D. All of the above

6. What are some reason(s) people finance their car?
   A. They do not have enough saved up
   B. They are saving their money for possible emergencies
   C. They can only afford to pay monthly payments
   D. All of the above

7. If you were financing a $20,000 car, for 5 years at 5.9% APR, how much would your monthly payments be?
    A. $285
    B. $385
    C. $485
    D. $585

House Quest Questions:

1. What percent is a typical down payment when buying a home?
    A. 0 %-9%
    B. 10%-20%
    C. 25%-35%
    D. 50%-60%

2. What do banks look at in order to determine whether to approve a loan?
    A. Income
    B. Credit History
    C. Property Value
    D. All of the Above

3. True or False Interest rates vary from year to year.
    A. True
    B. False

4. If you have lower credit scores you will have lower interest rates.
    A. True
    B. False

5. Which one of these will have the highest interest rate?
    A. Savings Account
    B. Certificate of Deposit (CD)
    C. Home Loan
    D. Checking Account

6. Which one of these is a fee associated with mortgages?
    A. Appraisal Fees
    B. Closing Costs
    C. Underwriting and processing Fees
    D. All of the Above

7. If you are unsure about how long you will live in an area, it may be wiser to rent (rather than buy) a house.
    A. True

B. False

8. Which one of these are factors in deciding whether to buy or rent a house
   A. How much savings you have
   B. Your credit score
   C. If you can afford a down payment
   D. All of the Above

9. Does the number of persons living with you in a home affect the decision to buy or rent?
   A. Yes
   B. No

10. Which of these are places you might apply for a home mortgage?
   A. Mortgage Company
   B. Savings and Loan
   C. Credit Union
   D. All of the Above

APPENDIX B: QUESTION SETS FOR SECOND LIFE FOR STATISTICS

HOMEWORK SESSIONS

Game Questions for the First Homework Assignment

Question 1:

The AllStar cell phone company has recently hired your team to help with statistical decision making. Their first question for you focuses on the time it takes an older population (age 65+) to **retrieve an address** from the address book. They want to know if an address can be retrieved from their new design (the large phone with flat buttons) **in less than 15.2 seconds**. They believe that if they can show this type of performance, that they will get a larger portion of the market. Please answer the answers as they are presented to you. Use 0.05 as a significance level.

To begin, click on the phone with the large flat buttons to get the sample data (n = 30).

1. What type of test would you use?
    a. Statistic test
    b. Hypothesis test
    c. Usability test

2. What would be the null hypothesis (H_o)?
    a. Time is < 15.2 sec
    b. Time = 15.2 sec
    c. Time is better than competitor's

3. What's the alternate hypothesis (H_1)?
    a. Time is equal to 15.2 sec
    b. Time is less than 15.2 sec
    c. Time is faster than competitor's

4. What type of test is this?
    a. One sided test
    b. Two sided test

5. What is the mean of the sample?

    a. 15.25 sec
    b. 14.98 sec
    c. 16.2 sec

6. What is the sample standard deviation of the sample?

    a. 0.17 sec
    b. 0.24 sec
    c. 0.67 sec

7. What type of test would you use?

    a. Z test
    b. t test

8. What is the P-value of the test?

    a. 0.06
    b. 0.00
    c. 0.10

9. What do these results mean?

    a. This is significant and we can reject the null hypothesis
    b. This is not significant and we cannot reject the null hypothesis

10. What is α error?

    a. Rejecting the null hypothesis when it is true
    b. Failing to reject the null hypothesis when it is false

11. What is β error?

    a. Rejecting the null hypothesis when it is true
    b. Failing to reject the null hypothesis when it is false

12. What's the relationship between β and power?

    a. Power is the negative of type II error
    b. Power is 1 – β
    c. Power is β + 1

13. If you increased the sample size, what would happen to β?

    a. β would increase
    b. β would decrease

14. What would you conclude to your supervisors?

a. There is strong evidence that it would take users less than 15.2 seconds to retrieve an address from the phone
b. There is not strong evidence that it would take users less than 15.2 seconds to retrieve an address from the phone

Question 2:

The All Star Cell Phone Company has recently hired your team to help with statistical decision making. The second question focuses on the time taken for the older population to **dial a number** with their new design-the **phone with small flat buttons**. Following are the tabulated observations. They want to know if the time taken to dial the number exceeds 15.8 seconds. Please answer the following questions. Use significance level of 0.05.

To begin, click on the phone with the small flat buttons to get the sample data (n = 10).

1. What type of test would you use?

a. Z Test
b. t Test

2. What's the null hypothesis?

a. Time =15.8 sec
b. Time < 15.8 sec
c. Time >15.8 sec

3. What's the alternative hypothesis?

a. Time = 15.8 sec
b. Time < 15.8 sec
c. Time > 15.8 sec

4. What's the value of the test statistic (t_0)?

a. 4.00
b. 3.18
c. 5.29

5. What's the P value for this Test?

a. 0.002 (estimated from the tables; it's between 0.001 and 0.0025)
b. 0.063
c. 0.089

6. What would you conclude to your supervisors?

a. There is strong evidence that it would take more than 15.8 seconds for the users to dial a number
b. There is no strong evidence that it would take more than 15.8 seconds for the users to dial a number.

7. What is the 95% confidence interval on the time needed to dial a number?

a. $15.92 \leq \mu$
b. $14.89 \leq \mu$
c. $15.92 \geq \mu$

Question 3:

The All Star Cell Phone Company has recently hired your team to help with statistical decision making. The second question focuses on the time taken for the older population to **retrieve an address** from the address book with their new design-**the phone with large pop-up buttons**. Following are the tabulated observations. They want to know if the time taken to dial the number is not equal to 12.8 seconds. Please answer the following questions. Use significance level of 0.05.

To begin, click on the phone with large pop-up buttons to get the sample data (n = 10).

1. What type of test would you use?

a. Z Test
b. t Test

2. What's the value of the test statistic (t_0)?

a. 4.20
b. 3.89
c. 2.34

3. What's the P value for this Test?

a. 0.006
b. 0.002 (estimated from tables)
c. 0.923

4. What's the 95% two sided confidence interval on mean time?

a. $9.52 \leq \mu \leq 11.23$
b. $12.90 \leq \mu \leq 13.12$
c. $15.90 \leq \mu \leq 16.12$

5. What would you conclude to your supervisors?

a. There is strong evidence that the mean for the users to retrieve a number is not equal to 12.8 seconds.
b. There is no strong evidence that the mean for the users to retrieve an address is not equal to 12.8 seconds.

Question 4:

The All Star Cell Phone Company has recently hired your team to help with statistical decision making. The second question focuses on the time taken for the older population to **dial a number** with their new design-**the Statsberry**. Following are the tabulated observations .They want to know if the time taken to dial the number is equal to 9.0 seconds or not. Please answer the following questions. Use significance level of 0.05.

To begin, click on the Statsberry to get the sample data (n = 30).

1. What type of test would you use?

a. Z Test
b. t Test

2. What's the value of the test statistic (Z_o)?

a. 0.10
b. -0.10
c. 0.56

3. What's the P value for this Test?

a. 0.825
b. 0.917
c. 0.256

4. Given a P-value of 0.917, would you reject the null hypothesis?

a. yes
b. no

5. What is the probability of type II error if the true mean was 9.05?

a. 0.65
b. 0.53
c. 0.24

6. What is the power for this test?

a. 0.53
b. 0.35
c. 0.47

7. If you increased your sample size to 50, what would be the value for beta of the test?

a. 0.77
b. 0.48
c. 0.22

8. What is the power for this test when the sample size is 50?

a. 0.52
b. 0.68
c. 0.78

9. What sample size would you need if you wanted a power of 0.90?

a. 152
b. 126
c. 214

Game Questions for the Second Homework Assignment

This assignment focuses on two-sample hypothesis tests, confidence intervals, and sample size determinations. Please answer the following questions by circling the correct answer. If not given, assume that alpha = 0.05.

Question 1: A company is trying to determine if the design of phone interfaces affects the time it takes to retrieve an address. They want to market the new phone to the 65+ market, but they need some statistics to substantiate their claim. Test the phone with the large buttons against the phone with the small buttons to determine if users can retrieve an address faster with the phone with the large buttons. Assume that the phone with the small buttons is sample one and the phone with the large buttons is sample 2. Assume that the population variances are unequal. Use n = 10. (sample one mean = 16.95 sec with sd = 0.158 sec and sample two mean = 14.99 sec with sd = 0.185 sec)

1. What is the null hypothesis?

    a) $\mu_1 = \mu_2$
    b) $\mu_1 < \mu_2$
    c) $\mu_1 > \mu_2$

2. What is the alternative hypothesis?

    a) $\mu_1 = \mu_2$
    b) $\mu_1 < \mu_2$
    c) $\mu_1 > \mu_2$

3. What assumption did you make to answer question 2?

    a) I assumed that the larger buttons would make the phone easier to use.
    b) I assumed that the color contrast would make the phone easier to use.
    c) I just like the look of phone from sample 1 better.

4. What type of test would you use?

    a) Z test
    b) t-test with $n_1 + n_2 - 2$ degrees of freedom
    c) t_0* test with degrees of freedom calculated from equation (degrees of freedom will be less than in answer b)
    d) t_0* test with degrees of freedom calculated from equation (degrees of freedom will be more than in answer b)

5. What is t_0*?

    a) t_0* = 2.60
    b) t_0* = 25.45
    c) t_0* = 34.2
    d) t_0* = 4.56

6. What is your conclusion?

a) There is strong evidence to reject the null hypothesis
b) I cannot reject the null hypothesis
c) I cannot reject the null hypothesis and believe that people can retrieve an address faster with phone 2 than with phone 1.

7. What is the one-sided bound for this test?

a) $\mu_1 - \mu_2 \geq 1.826$
b) $\mu_1 - \mu_2 \leq 1.826$
c) $\mu_1 - \mu_2 \equiv 1.826$

Question 2: You just started a new job in the usability group for a cell phone manufacturer. Your new supervisor isn't so sure of your capabilities, so he asks you some questions. Please answer the questions below.

1. You need to test whether a training intervention is effective. You decide to use a paired-t test. What is your justification for the use of this test rather than a regular t test?

a) You believe that the data is correlated
b) There are more than 500 data points
c) The variance differs between samples

2. If you use the pooled value for the variance, you are assuming:

a) The population variances are equal
b) The sample size is more than 500
c) The population variances are not equal

3. If you use the t_0* test, you are assuming:

a) The population variances are equal
b) The sample size is more than 500
c) The population variances are not equal

4. For a normal two sample t test (population variances assumed to be equal) the degrees of freedom are calculated the following way:

a) $n_1 + n_2$ -2 degrees of freedom
b) $n_1 + n_2$ -1 degrees of freedom
c) $n_1 + n_2$ + 2 degrees of freedom

5. If I want to compare two variances, I will be looking up critical values in the:

a) t distribution table
b) Z distribution table
c) Chi-square distribution table

6. I can only create a two-sided confidence interval on the difference between means when I perform a hypothesis test to:

a) Test whether the means are equal or not
b) Test whether mean 1 is greater than mean 2
c) Test whether mean 1 is less than mean 2

7. If you create a two-sided confidence interval for the difference in means and find that zero is contained within that interval, you would have done the following in the hypothesis test:

a) I would have rejected the null hypothesis
b) I would have failed to reject the null hypothesis
c) I wouldn't have done a hypothesis test

8. Can you have a negative chi-square random variable?

a) Yes
b) No

9. Go to page 469 in your book. If you are performing a two-sided t-test and want your power = 0.8 and your d = 1, what sample size for each group would you need?

a) 10
b) 6
c) 21

Question 3: Your company wants to determine if the Statsberry design leads to faster dialing of phone numbers than the phone with the large pop up numbers. Using your data, answer the following questions. Use n= 10 and assume that the population variances are equal. Assume that the statsberry is sample one and the phone with the large pop up numbers is sample 2. The mean time for the statsberry (sample one) is 8.99 sec and the sd = 0.166 sec and the mean time for the large pop up phone (sample two) is 11 sec and the sd = 0.149 sec.

1. What is your test statistic?

a) t = -28.46

b) t = 28.46
c) z = 1.96

2. What is the alternative hypothesis?

a) $\mu_1 = \mu_2$
b) $\mu_1 < \mu_2$
c) $\mu_1 > \mu_2$

3. What conclusions would you make?

a) There is strong evidence to reject the null hypothesis
b) I cannot reject the null hypothesis
c) I cannot reject the null hypothesis and believe that people can dial a number faster with the Statsberry.

Survey Questions

1. Before completing this assignment, I believed that Second Life was mostly an entertainment site.

2. Sometimes I have difficulty doing traditional homework assignments because I can't get feedback when I'm trying to complete the assignment.

3. I believe that the homework assignment in Second Life contributed to the learning of the material.

4. I believe that the feedback provided during the Second Life sessions helped me to more efficiently learn the material.

5. I enjoyed working with a team during the Second Life sessions.

6. I believe that I learned more about the material during the Second Life sessions than through traditional homework methods.

7. After completing the assignment, I believe that Second Life can be successfully used for educational purposes.

APPENDIX C: PERMISSIONS

From: stephen.palmer@step-10.com <stephen.palmer@step-10.com>
Date: Fri, Jan 8, 2010 at 17:28
Subject: Re: Image permission request
To: Sertac Ozercan <sozercan@gmail.com>

As author, I have no objection to your use of that image in your thesis ... and would enjoy reading your thesis if published on the Internet.

Best regards

Steve

On Jan 8, 2010, at 3:17 PM, Sertac Ozercan <sozercan@gmail.com> wrote:

> Dear Mr. Palmer,
>
> I am completing a thesis at Ohio University entitled "Adapting
> Feature-Driven Software Development Methodology to Design and Develop
> Educational Games in 3-D Virtual Worlds". In this research project, I
> analyze the educational games we developed with Feature-Driven
> Development and discuss the results and feedback received from
> students when deployed in classrooms.
>
> I would like to use an image from your book, "A practical guide to
> feature-driven development", in giving background on feature-driven
> development. The image is the five processes of FDD with their outputs
> (Figure 4-3).
>
> Please let me know this information ASAP, as the thesis document's due
> date is fast approaching.
>
> Thanks.
>
> Best Regards,
>
> Sertac Ozercan

From: Scott W. Ambler <sambler@ambysoft.com>
Date: Thu, Jan 14, 2010 at 06:50
Subject: Re: Image permission request
To: Sertac Ozercan <sozercan@gmail.com>
Cc: sambler@ambysoft.com

You have my permission.

- Scott
On Fri, January 8, 2010 3:17 pm, Sertac Ozercan said:
> Dear Mr. Ambler,
>
> I am completing a thesis at Ohio University entitled "Adapting
> Feature-Driven Software Development Methodology to Design and Develop
> Educational Games in 3-D Virtual Worlds". In this research project, I
> analyze the educational games we developed with Feature-Driven
> Development and discuss the results and feedback received from
> students when deployed in classrooms.
>
> I would like to use an image from your website "Agile Modeling (AM)
> Home Page", in giving background on agile software development
> methodologies. The image is the XP process in page
> http://www.agilemodeling.com/essays/agileModelingXPLifecycle.htm
>
> Please let me know this information ASAP, as the thesis document's due
> date is fast approaching.
>
> Thanks.
>
> Best Regards,
>
> Sertac Ozercan
>

Scott W. Ambler
Chief Methodologist/Agile, IBM Rational
http://www.ibm.com/developerworks/blogs/page/ambler

From: Ken Schwaber <ken.schwaber@verizon.net>
Date: Tue, Jan 12, 2010 at 16:04
Subject: Re: Image permission request
To: Sertac Ozercan <sozercan@gmail.com>

You have my permission,
Ken

On Jan 8, 2010, at 3:15 PM, Sertac Ozercan wrote:

> Dear Mr. Schwaber,
>
> I am completing a thesis at Ohio University entitled "Adapting
> Feature-Driven Software Development Methodology to Design and Develop
> Educational Games in 3-D Virtual Worlds". In this research project, I
> analyze the educational games we developed with Feature-Driven
> Development and discuss the results and feedback received from
> students when deployed in classrooms.
>
> I would like to use an image from your book, "Agile project management
> with scrum", in giving background on agile software development. The
> image is titled "The Overall Scrum Process" (Figure 1-3).
>
> Please let me know this information ASAP, as the thesis document's due
> date is fast approaching.
>
> Thanks.
>
> Best Regards,
>
> Sertac Ozercan