Systems Engineering Concepts with Aid of Virtual Worlds and Open Source Software: Using Technology to Develop Learning Objects and Simulation Environments

Latina Davis, Morgan State University
Maurice Dawson
Marwan Omar

Available at: https://works.bepress.com/maurice_dawson/38/
Handbook of Research on 3–D Virtual Environments and Hypermedia for Ubiquitous Learning

Francisco Milton Mendes Neto  
*Federal Rural University of the Semiarid Region, Brazil*

Rafael de Souza  
*Federal Rural University of the Semiarid Region, Brazil*

Alex Sandro Gomes  
*Federal University of Pernambuco, Brazil*
Chapter 20

Systems Engineering Concepts with Aid of Virtual Worlds and Open Source Software: Using Technology to Develop Learning Objects and Simulation Environments

Latina Davis  
Morgan State University, USA

Maurice Dawson  
University of Missouri – St. Louis, USA

Marwan Omar  
Saint Leo University, USA

ABSTRACT

Technology is changing the landscape of learning and teaching in America. The use of virtual worlds enable engineering and technology programs to implement software programs such as Second Life and Open Simulator to enhance what they may currently already have. Additionally, virtual worlds can add a more dynamic environment in the online classroom for multiple platforms such as the Personal Computer (PC), wearables, and mobile devices. The purpose of this chapter is to provide a review of these programs to include how to implement these items into an engineering course. Further detailed in this submission is how to incorporate Institute of Electrical and Electronics Engineers (IEEE) documentation and other engineering guidelines into the projects. Included in this chapter is a detailed layout of a simulated environment as well as various approaches of structuring and organization for classroom activities.

DOI: 10.4018/978-1-5225-0125-1.ch020
**SIMULATION**

Simulation allows for the imitation of a real world scenario or systems. This can be accomplished using software technology such as virtual worlds. Simulation can come in the form of training, education, video games, modeling, low fidelity prototypes, and usability. Simulation can use learning objects and incorporate other modern day technologies such as Google Glass for increasing teaching effectiveness.

**UBIQUITOUS LEARNING**

Ubiquitous Learning (U-Learning), supported by the revolutionary and abundant digital resources, is viewed as an effective learning approach for situating students in real-life and relevant learning environments that supports and promotes a variety of learning needs. U-Learning involves applying ubiquitous technologies in the enhancement of education strategies and models. Embedded Internet-based devices that we use in our daily live can present a supportive environment for U-Learning. The rise in Internet availability and accessibility has truly made a significant number of learning resources and options available to today’s students at all levels of education. U-Learning has the unique power of providing educational resources in a manner that is flexible, calm, and seamless due to its pervasive and persistent model (Martinez-Maldonado, Clayphan, Muñoz-Cristóbal, Prieto, Rodríguez-Triana, & Kay, 2013); U-Learning aims at removing educational and learning physical barriers by utilizing the advancements in technology. The ubiquitous learning has become more than a technology phenomenon and a prominent vision that strives to revolutionize the educational landscape and present technology-driven educational settings, because it thrives on the concept and idea of making a variety of educational and learning assets available to students, creates new and varied learning environments, customized learning and enables the realization of a series of training activities from anywhere, anytime and from any device (Durán, Alvarez & Únzaga, 2014).

Ubiquitous and pervasive learning environments offer students unique possibilities for team work and collaboration both face-to-face and remotely. These environments include an array of modern and innovative technologies at different stages of adoption: interactive whiteboards are already available in many classrooms; interactive tabletops are just starting to be introduced in schools (Kharrufa et al, 2013), and handheld devices are already used by students and teachers in the form of smart-phones or tablets.

**U-Learning Space and Design**

Many studies in the past have investigated the effectiveness of deploying different learning and teaching styles with different U-Learning environments to determine which strategy produces the best learning outcomes for students with different learning needs. It’s important to note that developing a u-Learning space has to take into consideration the outcome of the existing learning theories in terms of best practices, such as a structured relationship between information and learners’ understanding in educational settings. This helps to prevent learning isolated from a meaningful context. For example, if a student understands why and how something happens rather than just being told that it is true, then the information is more relevant and, therefore, is more meaningful to the student. The rationale for this is that how is the inclusion of the pedagogical information; and why is the inclusion of interactive learning, allowing students to create knowledge from what they perceive (Ogata & Yano, 2012).
Colorado Technical University has successfully deployed an award-winning, revolutionary U-Learning system that is called “Intellipath”. Intellipath™ is adaptive learning software that personalizes learning based on a student’s pre-determined knowledge state on a particular subject or topic, and creates individual learning nodes, or steps for the student to ensure the student understands each element to a specific unit of work. Intellipath™ allows the instructor and students to work closely together to ensure transfer of learning is taking place. What is unique about this learning technology is that the learning lessons are connected throughout the entire course map. Work on one node influences learning on other nodes in the course map. Learning is a continuum throughout the course, assessment is constant, and by revising nodes students can continually improve their learning.

Ubiquitous Computing

As computers become ubiquitous, they capture our attention and daily activity, which allow them to infiltrate into the background. Ubiquitous computing, however, includes computing devices such as smartphones, tablets, cameras, and other digital gadgets. Integrating ubiquitous computing into ubiquitous learning promotes the interaction between students and their digital gadgets to become connected with the manifold digital embedded devices and/or services (Möller, Haas, & Vakilzadian, 2013). Therefore, in a ubiquitous learning settings or environment, students have the unique ability of exploring the ubiquitous space built and powered by ubiquitous and mobile technology to interact with the various embedded digital devices and/or services. Thus, ubiquitous learning has the potential to create a sustainable and persistent learning and education environment that has barrier free and adapts to varying students learning needs.

Students have the advantage and ability of deciding which learning approaches best fit their learning needs and they are able to customize the environment to best fit their specifying situation (Martinez-Maldonado & Kay, 2013). In the U-Learning space, sharing information and knowledge between learners and mobile devices becomes a reality and contributes to creating a learning environment where learners can access, share, and distribute knowledge anytime and anywhere and therefore we become a more powerful society by connecting people, ideas, and knowledge. With U-Learning, we are able to create and available and accessible learning community utilizing mobile technology that makes learning attainable, traceable, and identifiable (Möller, Haas, & Vakilzadian, 2013).

Learning Objects

Learning objects allow for educational content to be broken down into smaller pieces that can be reused in various learning environments (Boss & Krauss, 2007). Learning objects are grounded in the object oriented paradigm of computer science (Wiley, 2000). These are digital resources uniquely identified and metatagged that can be used to support learning. Provided is a new and innovative method to reuse technologies in the learning environment. Thus learning objects (LSTC, 2000a) leads other candidates for the next generation of instructional design.

The IEEE Learning Technology Standards Committee (LTSC) System Interoperability in Education and Training has a couple actively working on an augmented reality learning experience model. This new standard will include technologies such as wearables (LSTC, 2000b). In virtual worlds these objects can be given a 3 Dimensional (3-D) representation which allows users to interact with these objects. Also behavioral tasks and indicators can be observed with 3-D learning objects (Vincenti, 2010).
SLOODLE

SLOODLE is an open source project which integrates virtual environments with the Moodle Learning Management System (LMS). This application allows to connect to a chat room, present, obtain feedback, management in world assignments with an assignment drop box, give quizzes, track points, identity linking, and more (Kemp, Livingstone, & Bloomfield, 2009). As this software application is integrated with the Moodle LMS it allows for universities to implement new technologies to enhance online education. In technical fields such as systems engineering this will allow for the creation and reuse of design objects.

For example, an engineer professor could search for virtual objects by associated tags. Those tags would bring forward the appropriate virtual object to the instructor. These objects would save hours in design time for a new lab. Additionally, in a software engineering oriented environment this could be useful in teaching the principles of code reuse in an interactive environment.

OPEN SOURCE APPLICATIONS AND TECHNOLOGY EFFECTIVENESS

Open Source Software

Open source software (OSS), as the name suggests, refers to the software that is made available and can be accessed by any user without requiring any fees. Unlike commercial proprietary software, the source code of open source software can be accessed, developed, and improved by any person and without any limitations (Jacobs, Kussmaul, & Sabin, 2011). Open source software started becoming popular and widely used by academia during the last two decades (Rooij, 2009). The reason behind the popularity and importance of using Free Open Source Software (FOSS) is that many educational institutions lack the financial capability to purchase proprietary or closed software that is usually very expensive. Therefore FOSS owes much of its success and credibility to academia where students and faculty alike were the pioneers to participate in developing and improving FOSS. The basic idea behind the development of OSS is that if many people view the same code then we will have a better opportunity to improving that code faster because it’s from the community and to the community. Members of the open source community feel some kind of commitment towards continuously improving open source projects.

Open Source Software for Educational Institutions

There is a myriad of free, open source software applications that can be utilized to enhance the learning process for students in the areas of software engineering, project management, database development, and web development. Studies have shown that FOSS is increasing gaining ground and has grown in use at an exponential rate by educational organizations (Rowell. 2008) We will discuss some of the major open source tools that could be easily utilized in a classroom setting and are currently in use by enterprise software developers. Moreover; such tools will serve as an invaluable resource for students who are planning to enter the business enterprise environment and enable them to head start their professional career. OSS can be defined as software that is made available in source code form. This is important as this source code may fall under the General Public License (GPL) which is a widely used free software license that is managed under the GNU Not Linux (GNU) Project (GNU, 2007). Virtualization is impor-
tant as this is an effective method to reproduce system learning environments on the same systems the learner is using reducing the overall hardware footprint and need to for a massive lab. This paper will also cover various software applications that can be integrated into the university system.

Virtualization

In terms of virtualization there are available tools to create a virtual version of a system. In terms of educational resources this provides a method for institutions to train on Virtual Machines (VMs). This allows a university to teach students complex techniques to computer science, engineering, or Information Technology (IT) students such as networking, programming, system administration, and Information Assurance (IA). There are multiple types of virtualization such as hardware, desktop, memory, storage, data, and network.

For institutions that would like the opportunity to provide a cloud like environment tools such as Oracle Virtual Box and Vmware Player provide that ability. However it should be noted that new Linux distributions running that require GNOME 3 will have issues running on older hardware. With older hardware as a constrained there are bare minimal Linux distributions such as Puppy Linux and Damn Small Linux (DSL). VMs provide the ability for a student to experiment with hundreds of Operating Systems (OSs) without installing or un-installing the base OS. As faculty members, we have used VMware software as well as Oracle virtual box as effective tools to host Linux as well as Windows operating systems; the results have been impressive in that students were able to better grasp the theories and principles presented in class because they had the opportunity to tinker with all the inner workings of those OSs. This approach also helped us save invaluable time and resources that would have otherwise been needed for installing and un-installing all those OSs.

Additionally, this allows for the creation of baseline OS images for classes. For example, an engineering course would have an OS created with all the software, case studies, and etc. preloaded. This baseline OS for software engineering would have development tools, static code analysis tools, debugging tools, case studies, eBooks, links to online course management tool, and etc. This would allow an institution to have image ready for every class to ensure consistency, and that the students have all required tools needed. In the case for a more technical course such as software engineering the students would have a baseline OS image with all the programming software, the Integrated Development Environment (IDE), quality testing tools, and etc. preloaded. In considering virtual environments the image can include the necessary installation software or preloaded software to immediately start work in the U-Learning environment.

Why We Need to Consider Linux Essential in Higher Education

Linux is an Unix like OS that is built on the Linux kernel developed by Linus Torvalds with thousands of software engineers. As of 2012 there are over two hundred active Linux distributions. The majority of the kernel and associated packages are free and OSS. This type of software provides a license which allows users the right to use, copy, study, change, and improve the software as the source code is made available. Providing source code allows developers or engineers to understand the inner workings of development. Imagine being able to study Mac or Windows by viewing all the source code to replicate similar developments. This exercise would be great for a developer to learn low level coding techniques, design, integration, and implementation (Dawson, & Al Saeed, 2013).
In terms of associated cost the majority of Linux distributions are free. However some distributions require a cost for updates or assistance that related to specific needs such as OS modifications for server hosting (Dawson, & Al Saeed, 2013). In software, there is a packet management system that automates the process of installing, configuring, upgrading, and removing software packages from an OS. In the Linux OS builds the most common packet management systems are Debian, Red Hat Package Manager (RPM), Knoppix, and netpkg.

**ENHANCING THE STEM ENVIRONMENT**

When discussing teaching tools one must consider all the OSS applications that can be used to improve Science, Technology, Engineering, and Mathematics (STEM) fields such as systems engineering (Dawson, Al Saeed, Wright, & Onyegbula, 2015). OSS provides the ability to do many technical items at a low cost and view source code of the software application (Dawson & Al Saeed, 2012). It is essential to take advantage of these tools and applications as many institutions of learning are having budget problems. These items allow for any institution to be competitive in instructions regardless of location. When thinking about U-Learning the virtual environment is key is the marketplace for low fidelity prototyping.

**SYSTEMS ENGINEERING GUIDANCE**

IEEE provides guidance on software and systems engineering. The INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities provides a great baseline for understand the field of systems engineering (Haskins, 2007). Recently INCOSE released an updated version that brings in more relevant items (INCOSE, 2015). Requirements engineering is essential when starting any program (IEEE Computer Society, 1998). Proper derivation and management of requirements add to the success factor in software engineering (Hofmann & Lehner, 2001). The same could be said in systems engineering as software is a subset engineering field.

**Software Engineering Standards**

Table 1 represents some IEEE software standards that are essential for systems engineering. All of the items below can be applied to a virtual project to limit the virtual environment for development. These software standards can be used as conditions or requirements that must be met while developing the projects.

**SYSTEMS ENGINEERING COURSE PROJECT: ATM MACHINE**

Detailed in the following pages is the required project for graduate systems engineering course. This project was designed to simulate an ATM while being developed with an engineering methodology. Through the design of this test environment multiple virtual objects were created as a result of this project through object oriented design techniques. The literature that follows provides the detailed design
**Table 1. List of IEEE Software Standards**

<table>
<thead>
<tr>
<th>IEEE Standard</th>
<th>Name of Standard</th>
<th>Additional Info &amp; Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE Std 730-2002</td>
<td>IEEE Standard for Software Quality Assurance Plans</td>
<td>This particular standard specifies the format and content of Software Quality Assurance plans (Lee et al, 2005).</td>
</tr>
<tr>
<td>IEEE Std 1028-2008</td>
<td>IEEE Standard for Software Reviews</td>
<td>This standard defines five types of software reviews and procedures for their execution. The five review types include management reviews, technical reviews, inspections, walk-throughs and audits (Westfall, 2008).</td>
</tr>
<tr>
<td>IEEE Std 1062-1998</td>
<td>IEEE Recommended Practice for Software Acquisition</td>
<td>This document recommends a set of useful practices that can be selected and applied during software acquisition (IEEE Standards Association, 1998).</td>
</tr>
<tr>
<td>IEEE Std 1233-1998</td>
<td>IEEE Guide for Developing System Requirements Specifications</td>
<td>This standard provides guidance on the development of a System Requirements Specification, covering the identification, organization, presentation, and modification of requirements (Moore, 1998). It also provides guidance on the characteristics and qualities of requirements such as objective or threshold requirements specification.</td>
</tr>
<tr>
<td>IEEE Std 1362-1998 (Reaffirmed 2007)</td>
<td>IEEE Guide for Information Technology-- System Definition-- Concept of Operations (ConOps) Document</td>
<td>This document provides guidance on the format and content of a Concept of Operations (ConOps) document, describing characteristics of a proposed system from the users’ viewpoint.</td>
</tr>
<tr>
<td>ISO 9001:2000</td>
<td>Quality Management Systems— Requirements</td>
<td>This standard has been debated upon in relation to the impact of quality management (Martinez-Costa, 2009).</td>
</tr>
<tr>
<td>IEEE/EIA 12207-2008</td>
<td>Systems and Software Engineering - Software Life Cycle Processes</td>
<td>An international standard to establish common framework for software life cycle processes. This is applicable to software products and the acquisition of systems.</td>
</tr>
<tr>
<td>IEEE/EIA 12207.1-1996</td>
<td>Industry Implementation of International Standard ISO/IEC 12207:1995, Standard for Information Technology-- Software Life Cycle Processes--Life Cycle Data</td>
<td>It is essential to know the basic relation between primary parties in the form of something that is binding (Gary, 1999). In this contract specified will be the requirements and the life cycle process model which will be used.</td>
</tr>
</tbody>
</table>
of a systems engineering project in a virtual environment with the systems engineering life cycle being used as the applied methodology. Displayed is an example of U-Learning that supports key concepts, contextual factors, and current practices in systems engineering.

**High Level Systems Analysis**

To begin any analysis of the system, a High Level Systems Analysis (HLSA) must be performed while in conjunction of communication with the users. This is the structure of the system, defining the essential core design features and elements providing the framework for required components and for ones that may follow resulting in future adjustments. A High Level Systems Diagram (HLSD) is constructed from the analysis that provides an engineering view of the users’ vision for what the system needs to be and do; including the paths that’s are required for the system to function properly. The HLSD is an approach in the methodology of SDLC following behind with a more in-depth model of the system a Low Level Systems Diagram (LLSD). From the graphical representation models a problem definition can be created to depict the problem of the system and what steps can be performed to begin the design and implementation of the system to be evaluated to fit requirements of the user.

Before performing any task the Stakeholder and Users of the proposed system, needs to communicate with researchers and developers to understand exactly what the Users wants; a High Level Systems Analysis (HLSA) takes place. From the HLSA a High Level System Diagram (HLSD) highlights graphically the main entities of the systems goals and objectives; also known as the scope. A problem definition will be determined from the analysis and JAD sessions, which then results in the analysis and design phase to implementing the project. Developers can perceive what the desire system ought to look like or function as, also being well aware if the system is complex or not. This gives developers a better way in deciding how to approach a problem. The entities that make up the HLSD shown in Figure 1 in this research paper are human users, an ATM Machine, and a Bank Network. These are what drive this system; there are dependent on each other to properly operate.

**Low Level Systems Diagram**

Low Level System Diagram (LLSD) shown in Figure 2 gives a more detailed graphical representation of the system. It allows intended users to visually recognize what exactly it takes to run the system including systems entities. The requirements within the LLSD will allow user to select requested account and expect task to be achieved.

*Figure 1. High Level Systems Diagram (HLSD)*
Problem Statement

The problem being addressed can be defined as the ATM machines are not reliable as developers want users to think that they are. Majority of ATM machines charge users to access their own personal accounts. As well as not having access to a 24 hour self cash services; that allows users to access their bank accounts to withdraw, deposit, transfer or check funds within their accounts. Existing ATM machines are not benefiting users at a hundred percent reliability. From the problem definition a scope is formed to focus primarily on the goal of the system.

Scope

The scope of this research can be expressed by the following statement. To analyze, model, implement, and evaluate an ATM machine inside a virtually society. The primary focus is a user accessing their bank branch that they are associated to perform task such as; check balance, deposit, transfer and or withdraw funds at any desired time. Outside of the scope is the interaction between bank network employees and the potential users of the system will not be discussed in depth.

Objectives and Goals

The main objective is to research and clearly understand the requirements of an ATM system, as well as understand the application model requirements for integration with an ATM system simulation. The goal of this research paper is to discuss the planning, analyzing, designing, implementing, testing and evaluating phases of the development of a graphical user interface of an ATM machine model using the software Second Life; virtual world.

Plan the path in which a developer will take to follow for production. Analyze requirements and literature review to understand the entities within the system.

Use OOAD to graphically model users, use cases and scenarios, data and flow diagrams.

Implement the OO model into Second Life by constructing an environment in which the system will possibly be able to operate in. Program objects to function when virtual users wants to perform a task.
Supporting Tools and Resources

The tools used in this research include Enterprise Architect (EA) software which is an advance modeling and design tool formatted in UML; the Interface Design Process Chart to follow a plane in developing a graphical user interface. Scratch building block coding to program objects for functionality as well as Second Life software to design and implement the proposed system in focus.

Microsoft Project 2007 was used to plan out the development of the system. The planner kept a detailed list of task that must be completed as well as providing a calendar to show graphically how far along the developer is on the project as well as keeping aware of deadlines.

Research Project Methodology

Three types of methods were used when developing the GUI for an ATM Machine. The three are known as the Interface Design Process Chart; which is a structured path in planning for development. The Systems Development Life Cycle; which is a process path aiding analysis within a project. As well as Scratch; which is a building block programming system allowing developers to write scripts to objects to perform a function or task.

Interface Design Process Chart

The process path that was used to develop and design the GUI was the Systems Development Life Cycle (SDLC) and Interface Design Process chart shown in Figure 3. The development of the ATM and environment was designed within Second Life which is a part of the design and implementation phases. The structured path of the development of a successful GUI aids developers. Beginning from the Needs Analysis and continuing to insure users’ requirements down to the summarative evaluation shows how each step is repeated for proper development. The path is beneficial because it provided guidance throughout the different planning stages.

Systems Development Life Cycle

The Systems Development Life Cycle (SDLC) consist of steps that will lead a developer to an effective and efficient system; planning, analysis, design, implementation, and maintenance are phases that are required for a successful system shown in Figure 4. The end of one process of a system begins another process. That being said a developer cannot proceed on within a development of a system without completing each process, but a system can repeat a phase if requirements aren’t successfully meant by the developer. The first phase is the planning, where an individual identifies the need for a new or enhanced system. Second phase is the analysis where study of the requirements of a system is identified by potential users and current systems. The third phase is the design stage, where the results from the analysts phase is modeled from a logical to a physical design. The fourth phase consist of implementing the system from either documents or models to be coded, tested and or installed into a real life application which the user involvement is an essential to the development of the system in focus. The last phase consists of maintenance where a problem of the developed system has aroused and a better solution has been
discussed for the system to perform better. This usually occurs after a potential user has tested the system and as mentioned before an error is found.

The diagram that is represented provides the procedures one will pursue in an object oriented (OO) particular approach to solving a problem. It is necessary to follow these procedures when approaching a complex system; beginning with the initial users’; it is necessary to coordinate and communicate with the users of a system to understand the system itself and what the users’ want which will take place within a Joint Application Development (JAD) Session where collaboration occurs.

For this particular research, literature review and communication between human users’ determined the ATM machine requirements. From the requirements based on the users’ needs and resources available, a HLSA was developed resulting in a HLSD to highlight all key components of the system, being able to model a LLSD; which gave a solid focus on what the project goals and objectives were; also known as the projects scope. With a diagram of the system, it was clear to see how complex the system is. A problem definition was determined from the analysis and JAD sessions, which resulted in the analysis and design phase to implementing the project. Within this paper only particular aspects of the OOAD design phase will be addressed.
Object Oriented Analysis and Design

Object-Oriented Analysis and Design (OOAD) follows a structure, in which systems can be planned, analyzed, designed, implemented, and tested using models as a graphical representation of the system. A very familiar structure that OOAD follows is known as the System Development Life-Cycle (SDLC) mentioned previously. Object-Oriented Analysis and Design (OOAD) is distributed amongst multiple diagrams in several analysis techniques. Object-Oriented Analysis (OOA) focuses on the techniques on analyzing the requirements for a system. Object-Oriented Design (OOD) focuses on the implementation of the system. “OOA focuses on what the system does, OOD on how the system does it”. Examples of how a system should work are made up of diagrams are used to further enhance the design of a system and the properties within a system. Within the design phase models are constructed from the analysis of the system in focus. Models consist of the Use Case Model as well as their Scenarios. From the Use Case Models, Class diagrams can be constructed as well as Object, Sequence and Activity Diagrams. An OOAD detailed description of the case study of the ATM simulation will be discussed within the Case Study section of this paper.

Use Case and Use Case Scenario

As stated before to have a sufficient outcome of a complex problem the Object-Oriented Analysis and Design approach is considered an efficient approach. Following the Systems Development Life Cycle diagram in Figure 4, the first step in analysis and design would consist of the Use Case Model. In this
project, the goal is to observe the requirements of the ten instruments of a flight deck system. Within the use case modeling, scenarios will be created. The scenarios describe the requirements of this system and their subsystems.

The Use Case Model focuses only on the Actor(s) and requirements of the system. Figure 5 illustrates the use case diagram for the ATM Machine System. The ATM machine is composed of seven use cases; that interact to provide the users access to their bank accounts. Use Case scenarios are depicted from the use case requirements, which would allow one to construct a Class Diagram Figure 5 illustrates a close view of a Use Case and its Actors with use case scenarios provided.

Users are represented by Actors because it is an entity that interacts with the system. They are the systems external use because they provide the information that is processed within the system. An Actor can be a person, computer hardware, or device just to name a few for example. The use cases are requirements that must fulfill the environment that it is in. The environment is what surrounds the use cases and separates the users. The particular environment within this project is the ATM Machine system. Built within use cases are scenarios; which are the procedure it takes to fulfill the requirement.

Below are three Case Scenarios from the ATM Machine Case Study.

*Figure 5. Use case diagram*
Login

1. User inserts and remove bank card to be read by ATM Machine.
2. ATM Machine prompts User to enter pin number.
3. User enters pin number.
4. ATM Machine prompts User to either select Checking or Saving Account if correct pin number is entered.
   a. If incorrect pin number is entered the User has 2 more attempts to try to access account before system locks out User and then the User will need to contact their bank branch for further assistance.
5. User selects desired account.
6. ATM Machine prompts User to select a desired task.
7. User either selects: Check Balance, Withdraw Funds, Deposit Funds, or Transfer Funds.
8. ATM Machine sends for request.

Withdraw

1. User selects “Withdraw Funds” task.
2. ATM Machine request selected task.
3. ATM Machine sends message back “Select Amount”, only $500 maximum can be withdrawn from the machine.
4. User selects desired amount:
   a. If User has enough funds within their account, the system will process the request.
   b. If User does not have enough funds within their account requested, the system will send a message back “Insufficient Funds” and logs the User out of the system.
5. If request processes, the ATM Machine sends message back “Would You Like A Receipt, Yes or No.”
6. If User either selects “Yes”, the amount requested and balance of the desired account will be printed on a receipt with desired amount of cash requested.
   a. If User selects “No”, ATM Machine request amount of cash desired and sends message to screen “Would You Like Another Transaction, Yes or No.”
      i. If User selects “Yes”, the system will prompt the User to select another task.
      ii. If User selects “No”, the system will ask the User to log out of the system.

Transfer

1. User selects “Transfer Funds” task ATM Machine request selected task.
2. ATM Machine sends message back “Enter Amount Transfer to desired account”, No maximum limit can be transfer between accounts.
3. User enters desired amount to transfer, then selects the “Done” option.
4. ATM Machine calculates transfer and sends message back “Would You Like A Receipt, Yes or No.”
   a. If User either selects “Yes”; the balance of the desired account will be printed on a receipt.
   b. If User selects “No”; ATM Machine sends message to screen “Would You Like Another Transaction, Yes or No.”
      i. If User selects “Yes”, the system will prompt the User to select another task.
      ii. If User selects “No”, the system will ask the User to log out of the system.

Class Diagram

From the analysis of a use case diagram and its scenario, forms a class diagram. Attributes and operations are expressed which consist of the instruments functions. As well as how the instruments relate and interact with one another. The class diagram gives one a whole visual understanding of how other systems relate to a particular system. A Class is a generic definition for a set of similar objects. It captures and specifies the properties and behaviors that are essential to the system. The Class determines the structure and capabilities of its objects. As stated before, Classes have has attributes and operations.

An attribute are properties or more often things in the real world with name and values. An attribute captures the characteristics of an object. While an operation performs a function in order to provide services to the system also known as a method. Within the model there are indicators that relate to properties and operations. (-) indicate the properties and (+) indicates the operations

Class Diagrams are based off of the analysis of Use Case Diagrams. The ATM Machine Use Cases are illustrated in classes with attributes expressed in terms of characteristics and the operations are based on the systems functionality. Figure 7 illustrates the ATM Machine class diagram. This diagram consists of two entities and an interface to work amongst each other within the scope of this research. The ATM Machine is the main instrument in use pertaining to the image. Users must first gain access from the ATM machine before entering their account.
Object Diagram

Object diagrams originate from class diagrams as well as use case scenarios. The nouns or objects a use case scenario has, represents the potential class diagram for the use cases. Where then an object diagram is constructed with its behaviors and states instead of attributes and operations. An object diagram shows the data of return types for behaviors within a system. Objects are self-contained with well-defined characteristics. Objects may have many states but cannot act in multiple states at one time. A state would consist of which an object exist in. The states are represented by the values of properties. For example, when a human is in an awake state; they have behaviors such as standing, walking or running. The behavior of an object is related to how an object acts and reacts. Objects behaviors are known as methods and or functions and can be either physical or conceptual.

Physical objects are tangible as well as being visible and touchable for example an automobile. Conceptual objects are intangible such as a bank account and a time schedule. Within the research project there will be conceptual objects discussed. An Object Diagram represents Class diagrams in depth detail from the use case scenarios. From the use case diagram for the ATM System, there is a use case named “Log In”. The use case scenario for the “Log In” is provided in detailed below:

Login

1. User inserts and remove bank card to be read by ATM Machine.
2. ATM Machine prompts User to enter pin number.
3. User enters pin number.
4. ATM Machine prompts User to either select Checking or Saving Account if correct pin number is entered.
   a. If incorrect pin number is entered the User has 2 more attempts to try to access account before system locks out User and then the User will need to contact their bank branch for further assistance.
5. User selects desired account.
6. ATM Machine prompts User to select a desired task.
7. User either selects: Check Balance, Withdraw Funds, Deposit Funds, or Transfer Funds.
8. ATM Machine sends for request.

   From the use case scenarios, nouns and or objects can be depicted to construct an Object Diagram. The Class Diagram is a template for objects; the Object diagram can be expressed in Figure 8 with the resulting attributes and operations. These are called behaviors and states.

   The object diagram shows the requested access and data return types of the behaviors to detect the ATM and Bank Networks for activation. The ATM system acts as a dependent to access uses accounts and communicate with Bank Networks.

**Sequence Diagram**

A technique to model various interacting diagrams would consist of a sequence diagram. The sequence shows the passage of time, interaction of objects, and sending messages between the objects. Sequence diagrams are constructed from object diagrams [6].

The sequence diagram shown in Figure 9 illustrates the user activating the ATM system to connect to Bank networks to access accounts. The ATM system connects to Bank Networks and validates users’ access code to access their account. The user is now able to select which account they would like to access as well as the desire task in favor.

*Figure 8. ATM systems object diagram*
Figure 9. ATM systems sequence diagram
Activity Diagram

To further enhance the design of the ATM system, an Activity Diagram was constructed to observe activities within the system of use case “Log In”. This is another form of illustrating the behaviors and states of the objects as they interact within the system. Figure 10 illustrates the activity diagram for “Log In”.

This particular type of diagram addresses the activities within a use case. The activity blocks are classified as action states. They are used to model a single step throughout the procedure. An activity diagram cannot be decomposed any further. Transitions are represented by an arrow connecting the two action state nodes. A black circle corresponds to the initial start of the process or procedure of the system. The system is complete when there is a circle with an X signifying the process is final. This is another form of illustrating the behavior of the objects as they interact within a system. The Activity Diagram is designed for modeling the performance of actions of a procedure or action within a system.

In the first activity the user inserts their bank card to begin operation, the ATM system sends a signal to the Bank Networks to verify the user’s card number. The ATM receives confirmation to allow the user to enter their pin code; if the pin code is valid the ATM system and Bank Network allows the user to have access to their account. The user is now able to select one out of the four options available, one at a time. When the user is done they are able to notice the ATM system that they have completed their transaction and wish to “Log Out” of the system; which is another use case of its own.

*Figure 10. ATM system activity diagram*
ATM Second Life Model

Second Life is an online virtual world, where people can socialize and connect with other people all over the world. It has been used to teach classes, concerts, and stores to name a few situations. People are able to buy, trade, learn and create their own world. Second Life Market was used to buy objects to write scripts to for the objects within the system will operate; such as furniture, screens, boats, and kiosk to name a few. The software can be used to virtually represent a system or an environment of its proposed operations. In Figure 11 the Second Life environment described can be observed. The furniture, pay phone, plants, ATM Machine objects and building was bought from the Second Life Market and the scratch software was implemented into the scripts of the ATM Machine objects to satisfy the development of the GUI in focus.

The primary purpose to this system is to develop a test plan to implement an analyzed system into a virtual world. The GUI system designed to be the ATM system will be developed within in the Second Life software. The users within Second Life will be able to touch the ATM system and the scripts will show within the chat log. The ATM system will “Welcome” the users and prompt the users to enter their pin number and select if they either would want to access their “Checking or Saving Account”. The system will then prompt the users to check their balance, withdraw, deposit or transfer funds within their account. Finally the system will notify the users that they have successfully completed their transaction and are logged out of the system with a notice displaying “Thank You” which completes the process. The use of scripts was used to generate when users “touch” the ATM, the time frame between each statement is 1 second. Total time it should take for users to complete the process within Second Life should be about 8 to 10 seconds long. Recording time started when subjects touched and activated the systems script.

Figure 11. Second Life ATM machine simulation
SECOND LIFE RESULTS

Subject Model 1

The first subject is a female ranging between the ages of 25 – 30. The subject decided out of the two ATM Machines she wanted to explore. The subject decided to interact with the ATM Machine closer to the entrance and exit of the environment; shown in Figure 12.

Subject 1 Results

1. **Time to complete test module**: 8 seconds
2. **Time to select and touch ATM object and run script**: 1 second
3. **Time to Complete Script**: Approximately 1 second per line

Subject Model 2

The second subject is a female ranging between the ages of 25 – 30. The subject decided out of the two ATM Machines she wanted to explore. The subject decided to interact with the ATM Machine further from the entrance and exit of the environment; shown in Figure 13.

Subject 2 Results

1. **Time to complete test module**: 12 seconds
2. **Time to select and touch ATM object and run script**: 3 second
3. **Time to Complete Script**: Approximately 1 second per line

*Figure 12. Subject 1 module*
Subject Model 3

The third subject is a male ranging between the ages of 25 – 30. The subject decided out of the two ATM Machines he wanted to explore. The subject decided to interact with the ATM Machine closer to the entrance and exit of the environment; shown in Figure 14.

Subject 3 Results

1. **Time to complete test module**: 10 seconds
2. **Time to select and touch ATM object and run script**: 2 second
3. **Time to Complete Script**: Approximately 1 second per line

Project Conclusion

In conclusion a GUI of an ATM Machine has been produced and integrated with the scratch software to program scripts within the objects of the system. Three types of methodologies were used to plan, analyze, design, implement, test and evaluate the developed system within this paper. Beginning from the initial problem definition and users’ requirements a HLSA was proposed resulting into graphically modeling the system with HL and LL systems diagrams. This allowed the developers to capture the main important entities within this project.
Once analyzing the problem and system a plan for design was implemented into Enterprise Architect (EA) using the SDLC OOAD methodology. The idea then lead to purchasing objects from the Second Life Market to use as prototypes within the environment of Second Life. Difficulties aroused while using the software Second Life. Difficulties such as programming objects as well as receiving objects from the market; some objects were unable to be modified and required to purchase other objects that would cooperate with the proposed system and environment. The overall experience was interesting in learning to plan a development of a GUI.

Future work will be to present the development process of the research project as well as further enhance knowledge within Second Life to use an effective tool in simulation work.

CONCLUSION

As technology is continuously changing the landscape of learning and teaching in America U-Learning has to be taken seriously. It is essential that learning institutions increase interaction, and productivity to ensure survival which increasing overall participation. The creation of learning objects for virtual worlds will decrease the course and lab development time for professors. Additionally, it will provide instructors the ability to teach interactive concepts of object oriented design and code reuse. The use of virtual learning environments helps enable systems engineering and technology programs integrate with modern technologies such as wearables. As learning occurs more in distance education the use of virtual learning environments needs to be further explored.
REFERENCES


**KEY TERMS AND DEFINITIONS**

**Linux:** An open source version of the UNIX OS (Perens, 2009).

**Open Source Software:** Software that allows the original source code to be free available which may be freely redistributed or modified (Perens, 2009).

**Requirements Engineering:** Per IEEE this refers to the process of defining, documenting, and maintaining requirements for a program. This can be applied to systems and software engineering (IEEE Computer Society, 1998).

**Quality Management:** A process that ensures products or systems meet a specific standard (Westfall, 2008).

**Second Life:** An online virtual world, where people can socialize and connect with other people all over the world (Boulos, Hetherington, & Wheeler, 2007).
**Software Engineering:** A discipline that focuses on the application of engineering to the design, development, integration, and maintenance of software (Ghezzi et al, 2002).

**Systems Engineering:** An interdisciplinary field within engineering that is focused on complex systems and managing their associated life cycle. (Shishko & Aster, 1995).