Developing Learning Objects for Engineering and Science Fields: Using Technology to Test System Usability and Interface Design

Maurice Dawson
Latina Davis, Morgan State University
Marwan Omar
Developing learning objects for engineering and science fields: using technology to test system usability and interface design

Maurice Dawson*
Illinois Institute of Technology,
IIT Tower, 10 West 35th Street,
Suite 14B, Chicago, IL 60616, USA
Email: maurice.dawson@ieee.org
*Corresponding author

Latina Davis
Morgan State University,
1700 E Cold Spring Ln,
Baltimore, MD 21251, USA
Email: latinaedavis@hotmail.com

Marwan Omar
Saint Leo University,
33701 FL-52, St. Leo,
FL 33574, USA
Email: marwan.omar@saintleo.edu

Abstract: Technology is changing the landscape of learning and teaching in US higher education. The use of virtual worlds enables engineering and technology programs to implement software programs such as second life, open simulator, or Lumiya 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), wearable’s, and mobile devices. The purpose of this paper is to provide a review of these programs to include how to implement these items into an engineering course. Further detailed in this article is how to incorporate Institute of Electrical and Electronics Engineers (IEEE) documentation and other design guidelines for the projects. Included in this document is a detailed layout of a simulated environment as well as various approaches to structuring and organisation for classroom activities.

Keywords: learning objects; technology; systems usability; interface design; limited fidelity prototyping; virtual worlds.

1 Introduction

Research has been conducted on design science in the discipline of information systems (ISs) to create and evaluate information technology (IT) artefacts to solve identified organisational problems (Peffers et al., 2007). Thus for technical instruction, this methodology can be used to develop learning objects that are simulated representations of the production software or system. Understanding a process is essential in understanding how systems can be designed, tested, and implemented. How design science could be used to address online management education was explored by North-Samardzic (2015). This paper provides how the use of design science for creating human-computer interfaces for learning objects that are to be used in engineering virtual educational labs.

2 Simulation

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

Ubiquitous learning (U-learning), supported by the revolutionary and abundant digital resources, is viewed as a practical learning approach for situating students in real-life and relevant learning environments that support 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 lives can present a supportive environment for U-learning. The rise in Internet availability and accessibility has indeed 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 et al., 2013); U-learning aims at removing educational and learning physical barriers by utilising the advancements in technology. The ubiquitous learning has become more than a technology phenomenon and a prominent vision that strives to revolutionise the educational landscape. Present technology driven educational settings thrive on the concept and idea of making a variety of educational and learning assets available to students. This creates new and varied learning environments, customised learning and enables the realisation of a series of training activities from anywhere, anytime and from any device (Durán et al., 2014).

Ubiquitous and pervasive learning environments offer students unique possibilities for teamwork and collaboration both face-to-face and remotely. These conditions include an array of modern and innovative technologies at different stages of adoption. Tools such as interactive whiteboards are already available in many classrooms. Interactive tabletops have been introduced in schools, and handheld devices are already used by students and teachers in the form of smart-phones or tablets (Kharrufa et al., 2013). These environments aid in the ability to deploy and create these virtual learning environments.

4 U-learning space and design

Many studies in the past have investigated the effectiveness of deploying different learning and teaching styles with various U-learning environments to determine which strategy produces the best learning outcomes for students with diverse learning needs. It’s important to note that developing a U-learning space has to take into consideration the results of the existing learning theories regarding best practices, such as a structured relationship between information and learners’ understanding in educational settings. This activity 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 and Yano, 2012).

5 Ubiquitous computing

As computers become ubiquitous, they capture our attention and daily activity, which allows 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 various digital embedded devices and services (Möller et al., 2013). Therefore, in ubiquitous learning settings or an environment enabled with ubiquitous computing options; students have the unique ability to explore the ubiquitous space built and powered by ubiquitous and mobile technology to interact with the various embedded digital devices 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 to decide which learning approaches best fit their learning needs and they can customise the environment to best fit their specifying situation (Martinez-Maldonado and 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 students can access, share, and distribute knowledge anytime and anywhere, and therefore we become a more dominant society by connecting people, ideas, and knowledge. With U-learning, we can create a learning environment and make it available and accessible to learning community by utilising mobile technology that makes learning attainable, traceable, and identifiable (Möller et al., 2013).

6 Learning objects

A ‘learning object’ approach to building learning content allows for educational content to be broken down into smaller pieces that can be reused in various learning environments (Boss and Krauss, 2007). Learning objects are grounded in the object-oriented paradigm of computer science (Wiley, 2000). These are digital resources uniquely identified, and meta-tagged that can be used to support learning. Provided is a new and innovative method to reuse technologies in the learning environment. Thus learning objects (LTSC, 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 standard will include technologies such as wearable’s (LTSC, 2000b). In virtual worlds, these objects can be given a 3 dimensional (3D) representation which allows users to interact with these objects. Also, behavioural tasks and indicators can be observed with 3D learning objects (Vincenti, 2010).

7 Simulation-linked object-oriented dynamic learning environment

The 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 dropbox, give quizzes, track points, identity linking, and more (Kemp et al., 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 engineering 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.

8 Open source applications and technology effectiveness

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 U-learning can be obtained, developed, and improved by any person and without any limitations (Jacobs et al., 2011). U-learning 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 U-learning (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 developing that code faster because it’s from the community and to the community. Members of the open source community feel some commitment towards continuously improving open source projects.

9 U-learning for educational institutions

There is a myriad of free, U-learning applications that can be utilised 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 increasingly gaining ground and has grown in use at an exponential rate by educational organisations (Rowell, 2008). We will discuss some of the major open source tools that could be readily utilised 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 environment and enable them to jump-start their professional career. OSS is 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). Virtualisation is important 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 large lab. This paper also covers various software applications that can be integrated into the university system.
10 Virtualisation

Regarding virtualisation, there are available tools to create a virtual version of a system. Concerning 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 IT students such as networking, programming, system administration, and cyber security. There are multiple types of virtualisation 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 offer 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 uninstalling the base OS. As faculty members, we have used VMware software as well as Oracle Virtual box as practical tools to host Linux as well as Windows OSs; the results have been impressive in that students were able to grasp better 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 uninstalling 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, 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, etc. This virtual instance would allow an institution to have an image ready for every class to ensure consistency, and that the students have all required tools needed. In the case of 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, etc. preloaded. In considering virtual environments, the image can include the necessary installation software or preloaded software to start work in the U-learning environment immediately.

11 Why we need to consider Linux essential in higher education

Linux is a 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 code of origin to
replicate similar software builds. This exercise would be perfect for a developer to learn low-level coding techniques, design, integration, and implementation (Dawson et al., 2013).

Regarding associated cost, the majority of Linux distributions are free. However, some distributions require a fee for updates or assistance that related to specific needs such as OS modifications for server hosting (Dawson et al., 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.

12 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 et al., 2015). OSS provides the ability to do many technical items at a low cost and view source code of the software application (Dawson and 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 organisation to be competitive in instructions regardless of location. When thinking about U-learning, the virtual environment is crucial is the marketplace for low-fidelity prototyping.

13 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 an excellent baseline for understanding 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 conditions add to the success factor in software engineering (Hofmann and Lehner, 2001). The same could be said in systems engineering as software is a subset engineering field.

14 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.
Developing learning objects for engineering and science fields

<table>
<thead>
<tr>
<th>IEEE standard</th>
<th>Name of standard</th>
<th>Additional info and citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE standard glossary of software engineering</td>
<td>IEEE standard glossary of software engineering</td>
<td>A glossary that contains the vocabulary for the software engineering domain (IEEE Standards Coordinating Committee, 1990)</td>
</tr>
<tr>
<td>terminology</td>
<td>terminology</td>
<td></td>
</tr>
<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 830-1998</td>
<td>IEEE recommended practice for software requirements</td>
<td>This document recommends the content and characteristics of a software requirements specification (IEEE Computer Society, 1998).</td>
</tr>
<tr>
<td></td>
<td>specifications</td>
<td></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-through 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 1220-2005 (ISO/IEC 26702)</td>
<td>IEEE standard for the application and management of the</td>
<td>(Doran, 2006) This standard is listed in a literature survey on international standards for systems requirements engineering (Schneider and Berenbach, 2013).</td>
</tr>
<tr>
<td>systems engineering process</td>
<td>systems engineering process</td>
<td></td>
</tr>
<tr>
<td>IEEE Std 1233-1998</td>
<td>IEEE guide for developing system requirements</td>
<td>This standard provides guidance on the development of a system requirements specification, covering the identification, organisation, 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></td>
<td>specifications</td>
<td></td>
</tr>
<tr>
<td>IEEE Std 1362-1998</td>
<td>IEEE Guide for Information</td>
<td>This document provides guidance on the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEEE Std 1362-1998 (Reaffirmed, 2007)</td>
<td>IEEE guide for information technology-system definition-</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></td>
<td>concept of operations (ConOps) document</td>
<td></td>
</tr>
<tr>
<td>IEEE Std 132-1998</td>
<td>IEEE guide-adoption of PMI standard – a guide to the</td>
<td>In the third edition of the PMBOK is recognised as an international standard which is the IEEE Std 132-1998 (Ahlemann et al., 2009).</td>
</tr>
<tr>
<td></td>
<td>project management body of knowledge</td>
<td></td>
</tr>
</tbody>
</table>
Table 1  List of relevant IEEE software standards (continued)

<table>
<thead>
<tr>
<th><strong>IEEE standard</strong></th>
<th><strong>Name of standard</strong></th>
<th><strong>Additional info and citation</strong></th>
</tr>
</thead>
<tbody>
<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 (Martínez-Costa et al., 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 (Gray, 1999). In this contract specified will be the requirements and the life cycle process model which will be used.</td>
</tr>
<tr>
<td>ISO/IEC 90003</td>
<td>Software and systems engineering – guidelines for the application of ISO 9001:2000 to computer software</td>
<td>This standard provides guidance for organisations in the application of ISO 9001:2000 to the acquisition, supply, development, operation and maintenance of computer software.</td>
</tr>
</tbody>
</table>

15 System usability scale

Usability is the effectiveness, efficiency, and satisfaction specified when users achieve specific goals in a particular environment (Jordan, 1998). The SUS is a reliable tool for measuring usability. It was created in 1986 by John Brooke (System Usability Scale, n.d.). This scale evaluates a broad range of products and services, including software, mobile devices, hardware, websites and applications (System Usability Scale, n.d.). Along with the research questions, the participants are asked to score the ten questions with one of five responses that range from strongly agree to strongly disagree. The scales consist of the following ten questions:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
Developing learning objects for engineering and science fields

5 I found the various functions in this system were well integrated.
6 I thought there was too much inconsistency in this system.
7 I would imagine that most people would learn to use this system very quickly.
8 I found the system very cumbersome to use.
9 I felt very confident using the system.
10 I needed to learn a lot of things before I could get going with this system.

16 Interpreting usability scores

Interpreting the usability scores can be complicated. The participant’s scores for each question are converted to a new number, added together, and then multiplied by 2.5 to convert the original scores of 0–40 to 0–100. Though the scores are 0–100, these are not percentages and should be considered only regarding their percentile ranking. Based on research, a SUS score above a 68 would be considered above average and anything below 68 is below average. However, the best way to interpret your results involves ‘normalising’ the scores to produce a percentile ranking.

17 System usability scale measures usability and learnability

SUS was only intended to measure perceived ease-of-use (a single dimension), recent research shows that it provides a global measure of system satisfaction and sub-scales of usability and learnability. Questions 4 and 10 provides the learnability dimension, and the other eight questions provide the usability dimension. This means you can track and report on both subscales and the global SUS score.

18 System usability scale is reliable

Reliability refers to how consistently users respond to the items (the repeatability of the responses). SUS has been shown to be more reliable and detect differences at smaller sample sizes than home-grown questionnaires and other commercially available ones. Sample size and reliability are unrelated so that SUS can be used on minuscule sample sizes (as few as two users) and still generate reliable results. However, small sample sizes produce imprecise estimates of the unknown user-population SUS score. You should compute a confidence interval around your sample SUS score to understand the variability in your estimate. The SUS will provide a way of measuring the end users experiences in the environment. This can be done periodically to ensure that the end users are still finding the environment usable.
19 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 a design 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 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 fundamental concepts, contextual factors, and current practices in systems engineering.

20 High level systems analysis

To begin any analysis of the system, a HLSA must be performed in conjunction with communication with users. This HLSA 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 scheme a low-level systems diagram (LLSD). From the graphical representation models, a problem definition can be created to depict the issue 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 doing any task the stakeholder and users of the proposed system need to communicate with researchers and developers to understand what the users want; a HLSA takes place. From the HLSA a HLSD highlights the primary entities of the system’s goals and objectives graphically; 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 desired system ought to look like or function as also being well aware of the system is complex or not. This gives developers a better way to deciding how to approach a problem. The entities that make up the HLSD show in figure 1 in this research paper are human users, an ATM, and a Bank Network. These are what drive this system; there are dependent on each other to operate properly.

Figure 1 High level systems diagram (see online version for colours)
21 Low level systems diagram

LLSD shown in Figure 2 gives a more detailed graphical representation of the system. It allows intended users to visually recognise what exactly it takes to run the system including systems entities. The requirements within the LLSD will allow the user to select requested an account. LLSD shown in Figure 2 gives a more detailed graphical representation of the system. It allows intended users to visually recognise what exactly it takes to run the system including systems entities. The requirements within the LLSD will allow the user to select requested an account.

Figure 2  Low level systems diagram (see online version for colours)

22 Problem statement

The problem being addressed was that the ATMs are not reliable as developers want users to think that they are. The majority of ATMs charge users to access their personal banking accounts. As well as not having access to 24-hour self-cash services; that allows users to access their bank accounts to withdraw, deposit, transfer or check funds within their accounts. Existing ATMs 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.

23 Scope

The scope of this research can be expressed by the following statement: to analyse, model, implement and evaluate an ATM inside a virtual society. The primary focus is a user accessing their bank branch that they are associated to perform a 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.
24 Objectives and goals

The primary objective is to research and 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, analysing, designing, implementing, testing and evaluating phases of the development of a graphical user interface of an ATM model using the software second life virtual world. Plan the path in which a developer will take to follow for production. Analyse 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 want to perform a task.

25 Supporting tools and resources

The tools used in this research include enterprise architect (EA) software which is an advanced modelling 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.

26 Research project methodology

Three types of methods were used when developing the GUI for an ATM. The three are known as, first: the interface design process chart; which is a structured path in planning for development, second: the systems development life cycle (SDLC); which is a process path aiding analysis within a project, and third: scratch; which is a building block programming system allowing developers to write scripts to objects to perform a function or task.

27 Interface design process chart

The process path that was used to develop and design the GUI was the systems 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 ensure users’ requirements down to the summative evaluation shows how each step is repeated for proper development. The path is beneficial because it provided guidance throughout the different planning stages.
The 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 are not successfully meant by the developer. The first phase is the planning, where an individual identifies the need for a new or enhanced system. The second step is the analysis where the 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 analysis phase are modelled from a logical to physical design. The fourth phase consists of implementing the system from either documents or standards 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 OO particular approach to solving a problem. It is necessary to follow these procedures when approaching a complex system beginning with the initial users’ as it is a need 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 requirements. From the requirements based on the users’ needs and resources available, an HLSA was developed resulting in an HLSD to highlight all key components of the system, being able to model an LLSD; which gave a substantial focus on what the project goals and objectives were; also known as the project’s scope. With a diagram of the system, it was evident 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 are being addressed.

29 Object-oriented analysis and design

OOAD follows a structure, in which systems can be planned, analysed, designed, implemented, and tested using models as a graphical representation of the system. A very familiar structure that OOAD follows is known as the SDLC mentioned previously. OOAD are distributed amongst multiple diagrams in several analysis techniques. Object-oriented analysis (OOA) focuses on the techniques for analysing the requirements for a system. Object-oriented design (OOD) concentrates 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
Developing learning objects for engineering and science fields

enhance the design of a system and the properties of 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 in the Case Study section of this paper.

30 Use case and use case scenario

As stated before to have a satisfactory outcome of a complex problem the OOAD approach is considered an efficient approach. Following the SDL C 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 modelling, 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. The ATM is composed of seven use cases; that interact to provide the user’s access to their bank accounts. Use case scenarios are depicted from the use case requirements, which would allow one to construct a class diagram. Users are represented by actors because it is an entity that interacts with the system. They are the system’s external use because they provide the information that is processed by 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 fulfil 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 system. Built within use cases are scenarios; which are the procedures it takes to fulfil the requirement.

Below are three case scenarios from the ATM machine case study.

30.1 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 two 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.
30.2 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.
   c 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”.
      • If user selects ‘yes’, the system will prompt the user to select another task.
      • If user selects ‘no’, the system will ask the user to log out of the system.

30.3 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”.
      • If user selects ‘yes’, the system will prompt the user to select another task.
      • If user selects ‘no’, the system will ask the user to log out of the system.

31 Class diagram

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

An attribute is 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 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 on the analysis of use case diagrams. The ATM use cases are illustrated in classes with attributes expressed regarding characteristics, and transactions are based on the system’s functionality. This graph consists of two entities and an interface to work amongst each other within the scope of this research. The ATM is the most important instrument in use about the image. Users must first gain access from the ATM before entering their account.

### 32 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 behaviours and states instead of attributes and operations. An object diagram shows the data of return types for actions 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 exists in. The states are represented by the values of properties. For example, when a human is in an awake state; they have behaviours such as standing, walking or running. The behaviour of an object is related to how an object acts and reacts. Objects’ behaviours 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 interchangeable such as a bank account and a 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 ‘login’. The use case scenario for the ‘log in’ is provided in detail below.

#### 32.1 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.
If incorrect pin number is entered the user has two 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 with the resulting attributes and operations. These are called behaviours and states. The object diagram shows the requested access and data return types of the behaviours to detect the ATM and Bank Networks for activation. The ATM system acts as a dependent to access uses accounts and communicates with Bank Networks.

### 33 Sequence diagram

A technique to model various interacting diagrams would consist of a sequence diagram. The sequence shows the passage of time, the interaction of objects, and sending messages between the objects. 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 desired task in favour.

### 34 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 behaviours and states of the objects as they interact within the system.

This particular type of diagram addresses the activities within a use case. The operation 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 original 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 behaviour of the objects as they interact within a system. The activity diagram is designed for modelling the performance of actions of a procedure or operation 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 users; 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 can 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.
35 Conclusions

As technology is continuously changing the landscape of learning and teaching in US U-learning has to be taken seriously. It is essential that education institutions increase interaction, and productivity to ensure survival while enhancing 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 to integrate with modern technologies such as wearable’s. As institutions look to makes leaps in instructional technology, the changes will occur in distance education. The contexts of the use of virtual learning environments need to be further explored from a perspective of design science. Tasks such as how to securely ‘log out’ of the system; which is another use case of its own allows for metrics to be captured regarding usability. This, in turn, allows the learning object to be enhanced that will directly impact the physical environment that is being simulated. And any improvements to the learning objects regarding usability will provide the learner to navigate with minimal problems. The use of this virtual technology with the Linux OS could provide an inexpensive to create a new learning environment in open source with taking in account of students’ perceptions of desktop OSs such as Ubuntu (Dawson et al., 2016). This concept is the future of engineering education.

References


