SYSTEMS ENGINEERING
Research Center

Design and Development Tools for the Systems Engineering Experience Accelerator – Volume I

Technical Report SERC 2015-TR-016-4

April 20, 2015

Principal Investigator: Dr. Jon Wade, Stevens Institute of Technology

Co-Investigator: Dr. Douglas Bodner, Georgia Institute of Technology

Research Team:

Dr. William Watson, Purdue University

Dr. Richard Turner, Stevens Institute of Technology

John Hinkel, Georgia Institute of Technology

Hao Kang, Stevens Institute of Technology

Alex Peizhu, Stevens Institute of Technology

Contract No. HQ0034-13-D-0004 Task Order: 0023, RT 123

Report No. SERC-2015-TR-016-4

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
20 APR 2015	**N/A**	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Design and Development Tools for the Systems Engineering Experience Accelerator â\" Volume I	**HQ0034-13-D-0004**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Bodner /Jon Wade Douglas**	5d. PROJECT NUMBER **RT 123**
	5e. TASK NUMBER **TO 023**
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Stevens Institute of Technology Georgia Institute of Technology**	8. PERFORMING ORGANIZATION REPORT NUMBER **SERC-2015-TR-016-4**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **DASD (SE)**	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Design and Development Tools for the Systems Engineering Experience Accelerator (SEEA) tools development efforts fall into four major categories â\" simulation tools for building and testing simulation models that mimic the behavior and results of acquisition programs, experience building tools that provide the structure for such system engineering experiences, learning assessment tools to measure the efficacy, and EA infrastructure changes to support this work. The simulation and experience building development projects achieved completion of the requirements and use cases, a review of existing tools to leverage, the development of a Prototype Sim Builder with GUI for model building and Phase Editor, Event Editor, Artifact Integrator respectively. Included capability was the manipulation of sub-models for modularity purposes and model archiving/curation and the development of a Prototype Sim Tuner that allows testing of model behavior drilled down to variables of interest. The tools are now at the stage where they are ready to be evaluated by external users for their use in Experience and Simulation development. An iterative development approach was quite successful at providing incremental functionality that was reviewed with its potential users throughout the research effort, prioritizing the most important features and delivering working prototypes throughout the effort.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**74**	

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract HQ0034-13-D-0004 (TO 0122).

Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense nor ASD(R&E).

TABLE OF CONTENTS

FIGURES

TABLES

EXECUTIVE SUMMARY

This document is a summary of the work that was completed in Part 1 of the SERC Research Topic DO1/TTO2/0123 "Design and Development Tools for the Systems Engineering Experience Accelerator (SEEA)" supported by Strategic Initiatives, Office of the DASD (Systems Engineering). The tools development efforts fall into four major categories – simulation tools for building and testing simulation models that mimic the behavior and results of acquisition programs that focus on system design and development, experience building tools that provide the structure for such system engineering experiences and the events that occur in them, learning assessment tools to measure the efficacy of the experience, and EA infrastructure changes to support this work.

The simulation tool development project achieved its Part 1 objectives with the completion of the requirements and use cases for simulation tools, a review of existing tools to leverage, the development of a Prototype Sim Builder with GUI for model building and capability to manipulate sub-models for purposes of modularity and model archiving/curation and the development of a Prototype Sim Tuner that allows testing of model behavior drilled down to variables of interest. The Experience building tools development project achieved its Part 1 objectives with the completion of the requirements, a review of existing tools to leverage, and the use cases and prototypes for the Phase Editor, Event Editor and Artifact Integrator. Progress was achieved in the learning tools area with a literature search, the exploration of an initial concept and the identification of data to be captured by the Experience Accelerator.

A successful demonstration of the tools has been given to the sponsor. The tools are now at the stage where they are ready to be evaluated by external users for their use in Experience and Simulation development. An iterative development approach was quite successful at providing incremental functionality that was reviewed with its potential users throughout the research effort, prioritizing the most important features and delivering working prototypes throughout the effort. This approach will be continued in Part 2 of this research program.

PROJECT DOCUMENTATION

The following are the documents that were produced by this research and may be referenced in this document:

RT123 Design and Development Tools for the SEEA Project documents:

- RT123 Technical and Management Work Plan (A009)
- RT123 Bi-Monthly Status Reports (A008)
- RT123 Design and Development Tools for the Systems Engineering Experience Accelerator, Part 1 proposal
- RT123 Design and Development Tools for the Systems Engineering Experience Accelerator, Part 2 proposal

Publications and Reports:

- Bodner, D., Wade, J. (2013) "Multi-Criteria Simulation of Program Outcomes," Proceedings of the 2013 IEEE International Systems Conference, 215-221, Orlando, FL, April 15-18, 2013.
- Bodner, D., Wade, J., Watson, W., Kamberov, G (2013) "Designing an Experiential Learning Environment for Logistics and Systems Engineering," Proceedings of the 2013 Conference on Systems Engineering Research (CSER), Atlanta, GA, March 20-22, 2013.
- Wade, J., Kamberov, G., Bodner, D., Squires, A. (2012) "The Architecture of the Systems Engineering Experience Accelerator", International Council on Systems Engineering (INCOSE) 2012 International Symposium/European Conference on Systems Engineering (EUSEC), Rome, Italy, July 9-12.
- Bodner, D., Wade, J., Squires, A., Reilly, R., Dominick, P., Kamberov, G., Watson, W. (2012), "Simulation-Based Decision Support for Systems Engineering Experience Acceleration", IEEE Systems Conference, Vancouver, BC, Canada, March 19-23.
- Squires, A., Wade, J., Watson, W., Bodner, D., Reilly, R., Dominick, P. (2012), "Year One of the Systems Engineering Experience Accelerator", Proceedings from the Ninth Annual Conference on Systems Engineering Research (CSER), Rolla, Missouri, March 19-22, 2012.
- Squires, A., Wade, J., Watson, B., Bodner, D., Okutsu, M., Ingold, D., Reilly, R., Dominick, P., Gelosh, D. (2011), "Investigating an Innovative Approach for Developing Systems Engineering Curriculum: The Systems Engineering Experience Accelerator", Proceedings of the 2011 American Society for Engineering Education (ASEE) Annual Conference and Exposition, Vancouver, BC, Canada, June 26-29, 2011.
- Squires, A., Wade, J., Dominick, P., Gelosh, D. (2011) "Building a Competency Taxonomy to Guide Experience Acceleration of Lead Program Systems Engineers", Proceedings from the Ninth Annual Conference on Systems Engineering Research (CSER), Redondo Beach, CA, April 14-16, 2011.
- Squires, A., Wade, J., Bodner, D., (2011), "Systems Engineering Experience Accelerator Workshop", National Defense Industrial Association (NDIA) 2011 14th Annual Systems Engineering Conference, San Diego, CA, October 24-27. (Presentation)
- Developing the Systems Engineering Experience Accelerator (SEEA) Prototype and Roadmap, Final Technical Report Year 1, SERC-2011-TR-16-1, May 31, 2011.
- Developing the Systems Engineering Experience Accelerator (SEEA) Prototype and Roadmap, Final Technical Report Year 1, SERC-2012-TR-16-2, October 24, 2012.
- Developing the Systems Engineering Experience Accelerator (SEEA) Prototype and Roadmap, Final Technical Report Year 1, SERC-2012-TR-16-3, December 31, 2013.

INTRODUCTION

Systems engineering is a multidisciplinary practice and is as much of an art as it is a science. While a waterfall model of education can provide a background of domain-centric knowledge, it is not until this knowledge is put into practice in an integrated, real world environment that a systems engineer can develop the necessary insights and wisdom to become proficient. In the workplace, these learning events are often distributed sparsely over time such that an engineer may only see a complete system life cycle over a period of several years. As a result, the maturation time from completion of formal studies to becoming seasoned systems engineer is unacceptably long, particularly when contrasted with the clock speeds of today's society in which career change is the norm rather than the exception, particularly among the young.

Clearly, there is a critical need to promote rapid skill development of systems engineers, in particular senior systems engineers, across the Department of Defense (DoD) and government workforces, as a large cohort of personnel is nearing retirement age. In addition, new systems engineering skills are needed to address important societal needs in national security, homeland security, airspace management and disaster recovery. These domains involve large-scale, systems-oriented solutions with increasingly limited budgets. Educational technologies hold the promise of providing customized learning exercises based on real-world situations to reduce the reliance on extensive on-the-job training that is the hallmark of current workforce development.

However, there are many challenges in developing effective educational technologies. Our prior work resulted in the development of the Systems Engineering Experience Accelerator (SEEA or EA); a first-generation educational technology set that provides experiential learning for systems engineers seeking technical leadership roles. This work also resulted in a learning experience based on design and development of an unmanned aerial vehicle (UAV) system for Department of Defense applications. What is of current interest is expanding the set of learning experiences to include different types of systems and development efforts. To do this successfully, we need to provide tools for the design and development of such experiences. This report addresses the first part of a three-part project that is creating such tools. The rest of the report is organized as follows:

- Reviews research literature on experiential learning
- Summarizes prior work on the SEEA
- Discusses progress on creating tools for designing and developing learning experiences for the SEEA
- Concludes and provides a summary of future work

LITERATURE REVIEW

In recent years, educational technology has received wide attention as a means to augment other forms of education and training (Aldrich, 2004; Aldrich, 2005; Cohen, 2006; Gibson et al., 2007; Herz et al., 2002; Jones, 1997; Prensky, 2001). In particular, we focus on the concept of an experiential interactive learning environment. An experiential interactive learning environment operates similarly to a role-based digital game, in that the learner assumes a role associated with a particular domain and engages in role-play within a digital environment representing the domain. The digital environment may assume different forms, from a simple text-based display to an immersive three-dimensional world. In an experiential interactive learning environment, of course, the role-play is aimed at furthering educational outcomes.

These types of environments often use Kolb's concept of experiential learning, which consists of four key elements through which a learner cycles in a continuous spiral of learning (Kolb, 1984). The four elements are (i) exposure to a concrete experience, (ii) reflection on that experience, (iii) generalization of the experience and formation of abstract concepts based on the generalization, and (iv) application of these concepts to the concrete experience. In an experiential interactive learning environment, the concrete experience is replaced with a simulated world, and the learner interacts with it, causing the simulated world to change.

Through these interactions and world changes, the learner can cycle through different experiences in the same domain or context, thus enabling a spiral learning process. Another concept typically utilized by these environments is constructivism, a theory that posits that learners construct their own knowledge rather than simply adopting pre-packaged knowledge. Goals for constructivist instruction include solving problems, reasoning, thinking critically and using knowledge both actively and reflectively, while conditions for effective constructivist learning include use of complex and realistic environments, use of social negotiation, multiple perspectives and learning modes for learners, encouragement for self-learning and self-awareness of knowledge construction (Driscoll, 2005). Yet another concept is that of role-playing educational experiences, which can be used to teach proper actions, vocabulary and ways of thinking in a particular domain (Shaffer, 2006; Van Ments, 1999). Educational technology can also be used to promote team or individual learning (O'Neil & Perez, 2008). While real social and physical environments have traditionally been used, researchers have concluded in recent years that computer-based environments are a suitable alternative (Brown et al., 1989; Herrington & Oliver, 1995).

Educational technology has been applied via numerous efforts to engineering education (Chapman & Martin, 1995; Chen et al., 2008; Ebner & Holzinger, 2007; Jain & Boehm, 2006; Morsi & Jackson, 2007; Okutsu, 2009; Steif, 2006). The Systems Engineering Experience Accelerator is one of the few applications of educational technology specifically for systems engineering workforce development (Bodner et al., 2012; Bodner et al, 2013; Squires et al., 2011a; Squires et al., 2011b; Squires et al., 2012; Wade et al., 2012). The current prototype has been used in a pilot demonstration-targeting curriculum for lead systems engineer candidates at Defense Acquisition

University. This demonstration utilized the UAV learning experience. The main two lessons addressed in this experience are:

- Problem solving and recovery, and
- Results from pressures to cut corners to make short-term goals while ignoring long term outcomes.

Several teams completed the demonstration learning exercise. Despite a number of technical issues relating to networking capabilities and application stability and shortage of class time, the potential of the Experience Accelerator was validated through feedback from the students. A number of the targeted lessons were clearly represented in the team presentations, an indication of the effectiveness of the EA in promoting its targeted learning outcomes.

For example, for problem solving and recovery, the importance of small schedule delays and the use of additional staff to remediate the schedule problem was touched on by several teams. Three teams mentioned the need to hire staff early in their lessons learned, while another called for more dramatic staff shifts. One team highlighted the results of slipping the schedule too much, lamenting that they were fired for slipping CDR. Another example can be seen with "Cutting corners to make short term goals while ignoring long term outcomes" which stresses the need to make decisions early. The teams repeatedly touched on this in their lessons learned. A team mentioned hiring more staff early, while another called for shifting staff earlier. A third noted how their early emphasis on software worked, and a fourth reflected on the need to ramp up staff more quickly.

These examples demonstrate that for these two-targeted learning outcomes in particular, nearly all of the teams provided positive feedback on learning outcomes. The different teams also highlighted additional learning objectives. This feedback resulted despite the fact that the learners fully completed only the first two phases of the five-phase experience before speeding through the remaining phases in order to see their results. Furthermore, the EA was designed to be played multiple times by learners, so these results are indicative of impressive potential learning gains given the limited implementation of the demonstration.

Design and development of the current prototype, as with any complex first article, involved substantial trial-and-error and low-level development activities. These activities, aside from enhancements, are one-time affairs for the technology set. However, they would not be one-time affairs for the creation of different experiential learning modules. New modules would require substantial low-level development and programming to create, thus discouraging widespread adoption of the EA.

Many of the available game creation tools focus on the ability to create 3D models and their interactions. Tools supporting the creation of dialog, game play and systems dynamics simulations are much more limited. While a commercially available tool was found for dialog creation, adequate tools were not discovered in the other areas. This peaks to the need for tools to help developers interested in creating their own learning experiences.

EXPERIENCE ACCELERATOR RESEARCH PROGRAM

The RT-16 Systems Engineering Experience Accelerator (SEEA) project created a new approach to developing the systems engineering workforce which augments traditional, in-class education methods with educational technologies aimed at accelerating skills and experience with immersive simulated learning situations that engage learners with problems to be solved. Although educational technology is used in a variety of domains to support learning, the SEEA is one of the few such technologies that support development of the systems engineering workforce.

SUMMARY OF ORIGINAL PROTOTYPE REQUIREMENTS AND FUNCTIONS

The SEEA was developed to support a single-person role-playing experience in a digital environment, as well as a specific learning exercise in which a learner plays the role of a lead systems engineer for a DoD program developing a new unmanned aerial system. This exercise is based on the notion of experiential learning, and thus will be referenced as an experiential learning module. The learner engages with the experience (i.e., simulated world), makes decisions to solve problems, sees that results of those decisions, abstracts lessons learned from what was successful and what was unsuccessful, and then repeats the process in a series of cycles, simulating the evolution of the program over time.

The SEEA technology provides a graphical user interface allowing the learner to see the program status, interact with non-player characters to gain additional program information, and make technical decisions to correct problems. It also provides capability to simulate the program into the future, based on these learner decisions, so that outcomes can be shown to the learner. This cycle of decision and simulation-into-the-future supports the Kolb cycle of experiential learning; the Experience Accelerator uses multiple such cycles operating through the lifecycle of the program. In particular, this approach allows illustration of the effect of upstream decisions on downstream outcomes in the system lifecycle. The SEEA can support a wide variety of systems domains and areas of expertise through changes to the experience. Recently, additional multi-player technology is being developed to allow live player support for team-based learning, as well as for a mentor to provide advice and feedback.

This project seeks to create a set of tools to enhance capabilities to rapidly and cost-effectively create experience modules that can be used by the current generic Experience Accelerator technology set to deliver a variety of learning experiences tuned to the needs of particular workforces. The current set of Experience Accelerator technologies is shown in Figure 1, which illustrates the architecture of these technologies. In designing the Experience Accelerator, two principles are relevant. First, the architecture is highly modular to allow substitution of new different technologies, to accommodate new and better technologies as they become available. Second, there is a generic set of technologies that are used to support any experience module. The specifics of a particular experience module are housed in libraries and other data files separate from the generic EA technologies.

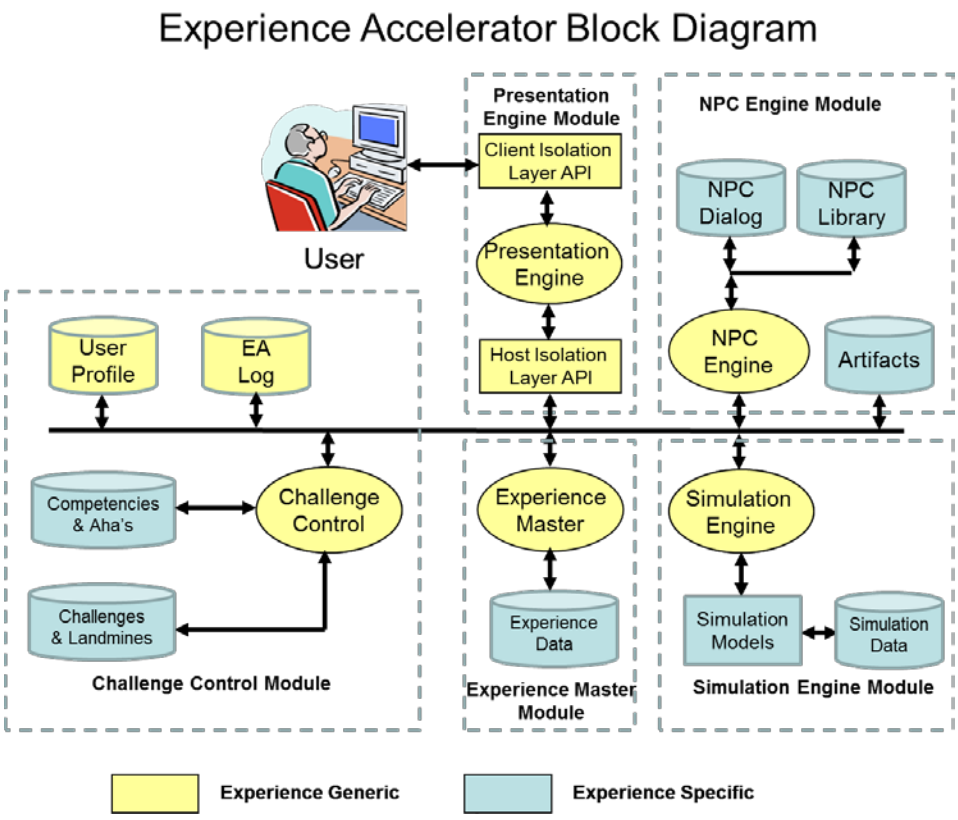

Figure 1. Experience Accelerator architecture

The modules currently included in the Experience Accelerator are:

- Presentation engine – Provides a user interface to the learner. This interface displays the status of the simulated program via a dashboard and status charts, and it lets the learner interact with a simulated program office. This interaction includes exploration of archived documentation on the program, querying of non-player characters about issues of interest regarding the program, and entering decisions about the program to correct perceived problems.
- Experience master – Maintains current value of state variables related to program status and performance and manages the invocation of other modules as needed. The experience master controls the flow of events and provides updated information from the other modules to the presentation engine for display. It also takes the learner inputs from the presentation engine and invokes internal functions or other modules as appropriate.
- Non-player character engine – Provides a number of non-player characters (NPCs) and associated dialog sessions based on the state of the program. The non-player characters have roles (e.g., program manager), and the learner is tasked with querying them about various aspects of program performance to determine the true state of the program and its underlying problems.
- Simulation engine – Maintains the detailed state of the program and advances this state via simulation models, incorporating learner decisions about the program. The simulation engine takes as input a base state of the program and then advances this state using any changes the learner may have entered. It also generates artifact charts detailing the program status

and performance over time (e.g., costs incurred, schedule and progress indicators, system performance estimates relative to targets, etc.).

- Challenge control engine – Maintains learner history and skill levels and provides parameter tuning to increase or decrease the difficulty of the experience based on learner knowledge and skills, and his/her success.

RESEARCH OBJECTIVES

While the existing technology infrastructure and experience content is useful, it is limited in its ability to support a community of educators and developers. The SEEA was developed with a goal of transitioning to an open-source sustainment model, which will provide long-term support for a community of educators and learners in creating learning exercises to address their specific needs. Currently, it is difficult to design and develop new educational content without significant knowledge of the SEEA design. Given the prototype nature of the SEEA with the primary goal of demonstrating its use as an effective education approach in pilot settings, this is an expected state for the SEEA technology at this stage of its evolution.

Problem Statement: *Traditional systems engineering education is not adequate to meet the emerging challenges faced by the Department of Defense and others in system acquisition and sustainment. New educational technologies such as the Systems Engineering Experience Accelerator hold the promise of facilitating a rapid skill and experience accumulation for the workforce to meet these challenges. However, to have scalable effect, such technologies cannot rely on extensive programming and low-level development to create a rich set of experiential learning modules needed to accelerate systems engineering workforce development.*

Hypothesis: *The Experience Accelerator technology will scale to support a community of developers engaged in creating modules for their organizations' use if tools are developed that allow educators and other non-programmers to create, maintain and evolve experiential learning modules.*

Measurable Outcomes: *The outcomes from this research will be measured in two main ways. First, educators and others interested in creating experiential learning modules will provide qualitative feedback on the effectiveness and efficiency of the Experience Accelerator toolset in creating experiential learning modules based on their use of the design and development tools. Second, the number of such developers who commit to create modules for their organizations' use will be tracked, as well as the number of organizations and variety of different application areas.*

Research Goal: *Validate the hypothesis through the creation of design and development tools for experiential learning modules that maximally leverage the current Experience Accelerator research, technology and content.*

IDENTIFICATION OF TOOLS NEEDED FOR FUTURE EXPERIENCE DESIGN AND DEVELOPMENT

This research task is focused on developing a set of tools specifically for educators and developers outside the SEEA research and development team, to support their designing and developing learning modules for their use. It concentrates on a subset of possible tools prioritized by the likelihood of having the most impact on facilitating module development:

1. Sim Builder - Simulation model builder using libraries/templates
2. Sim Tuner - Parameter tuner that automates the tuning of parameters to yield desired outputs via batch processing of different combinations of settings
3. Phase Editor - GUI-based tool for phase, cycle and event specification, with code generation
4. Event Editor - GUI or text-based tool to specify events and their triggers, with code generation
5. Artifact Integrator - Application that allows designer to take artifact files, such as design documents and enter it into EA application with automatic recompilation and re-linking
6. Learning Assessor - Assessment tool-suite that provides automated performance scoring and decision comparisons against proven baselines

Table 1 shows the limitations of designing and developing new experiences mapped to the tools above.

Table 1. Limitations and solutions

#	Limitation	Solution
1	Complex simulation models with limited reuse supported	**Sim Builder** - Simulation model builder utilizing libraries/templates
2	Complex simulation outputs dependent on hundreds of variables/parameters	**Sim Tuner** - Parameter tuner that automates the tuning of parameters to yield desired outputs via batch processing of different combinations
3	Manual nature of phase and cycle development	**Phase Editor** - GUI-based tool for phase, cycle and event specification with code generation
4	Manual nature of specifying events and their triggers in the Experience	**Event Editor** - GUI or text-based tool to specify events and their triggers with code generation
5	Manual nature of artifact integration involving re-linking and recompilation	**Artifact Integrator** - Artifact entry application that allows designer to take an artifact file and enter it into EA application with automatic recompilation and re-linking
6	Manual nature of assessment of learner performance	**Learning Assessor** - Assessment tool-suite that provides automated performance scoring, decision comparisons against proven baselines, etc.

It should be noted that the set of design and development tools noted above does the team identify a subset of the possible tools and template, and they are prioritized as the most important ones for inclusion in this work. Others include:

- Experience concept tools (storyboarding, learner profile creation,

- Context tools (project specification linked to learning outcomes, NPC roles/motivations/personalities), and
- Experience module events/flows (automated linkages between challenges/landmines and competencies/'aha's' and mitigating actions/effects)

The prioritization methodology and overall results are presented in Appendix A. Below is a short description of each tool.

Simulation Tools:

These two tools work interactively to allow the construction, test and tuning of systems dynamic models. Both the Sim Builder and Tuner tools have application outside of the Experience Accelerator in the development of system simulation models, particularly those in systems dynamics.

1. **Sim Builder** – This tool provides the ability for non-technical staff to build systems dynamics models based on existing templates in a GUI environment.
2. **Sim Tuner** – This tune provides the ability to analyze the system, determine the sensitivity of various parameters, and aid in the tuning of the system to achieve desired behaviors.

Experience Building Tools:

These three tools are used together with Chatmapper[1] to allow non-technical staff the ability to build and modify EA experiences without any programming. Chatmapper is an application that supports dialog creation. In most cases, this work will be accomplished through a GUI and a single workbench.

3. **Phase Editor** – This tool provides the ability to change the finite state machine that changes the phases within an EA experience. For example, the project phases can be customized to new domains and environments and can be constructed to represent state changes that are not affiliated with formal project states.
4. **Event Editor** – This tool provides the capability to create and edit events during an experience and the activities that may trigger them. For example, a phone call from the learner's supervisor can be triggered based on a decision made by the learner or the state of the project.
5. **Artifact Integrator** – This tool allows an experience builder the ability to quickly upload an experience change, be it a new artifact such as a new dialog or report, or a changed phase and or event, and test the results without having to do any programming.

Learning Analysis Tools:

[1] www.chatmapper.com

This tool has application outside of the Experience Accelerator and can be used as a tool for learning in a number of experiential environments.

6. **Learning Assessor** – This tool will analyze the subject's activities, decisions, project performance and self-assessments to determine the learning level achieved. This work will involve developing the logging ability to collect the necessary information, and an analysis tool for making the final predictions.

To design and implement the design and development tools for the Systems Engineering Experience Accelerator, work consists of the following major categories:

- Determine requirements for each tool described above through discussions with the user community.
- Review existing tools and technology for potential leverage and reuse.
- Develop and integrate the tools described above.
- Enhance the Experience Accelerator technology as necessary to support these tools.
- Integrate the tools with the Experience Accelerator (e.g., the artifact entry app spans modules including NPC dialog, project background document, simulation models and outputs, etc.).
- Validate the tools efficacy through their use in developing new prototype Experiences.

RESEARCH APPROACH FOR THIS TASK

Traditional systems engineering education is not adequate to meet the emerging challenges faced by the Department of Defense and others in system acquisition and sustainment. New educational technologies such as the Systems Engineering Experience Accelerator hold the promise of facilitating a rapid skill and experience accumulation for the workforce to meet these challenges. However, to have scalable effect, such technologies cannot rely on extensive programming and low-level development to create a rich set of experiential learning modules needed to accelerate systems engineering workforce development.

We are utilizing the following approach to scale the SEEA to include additional experiences.

Hypothesis: The Experience Accelerator technology will scale to support a community of developers engaged in creating modules for their organizations' use if tools are developed that allow educators and other non-programmers to create, maintain and evolve experiential learning modules.

Measurable Outcomes: The outcomes from this research will be measured in two main ways. First, educators and others interested in creating experiential learning modules will provide qualitative feedback on the effectiveness and efficiency of the Experience Accelerator toolset in creating experiential learning modules based on their use of the design and development tools. Second, the number of such developers who commit to create modules for their organizations' use will be tracked, as well as the number of organizations and variety of different application areas.

Research Goal: Validate the hypothesis through the creation of design and development tools for experiential learning modules that maximally leverage the current Experience Accelerator research, technology and content.

Research Approach: The tool developments will use an iterative process that will involve some or all of the following research activities in each Part:

Specify:

1. **Determine project goals & success metrics:** Determine goals and success metrics based on EA research team meetings with sponsors and EA developers based on their future EA design and development needs.
2. **Identify critical use cases**: Identify potential external Experience developers who can be define objectives, use and evaluate the toolset. Create test case scenarios based on Experience design and development objectives from these third parties.
3. **Review existing tools and technology:** Identify relevant existing tools and technologies and determine which to use and/or leverage.
4. **Specify initial tool design:** Specify requirements and features of the each of the desired new tools. Review this with the sponsor and identified external Experience developers.

Develop:

5. **Develop and test tools:** Develop and test tools in an agile development process with period feedback from the potential future users.
6. **Revisit design as needed:** Revisit the design and (re)assign feature priorities based on user input.
7. **Test:** Test the tools by having the potential users create portions of their prototype Experiences per the identified test case scenarios.

Evaluate:

8. **Perform usability analysis:** Record and analyze the results of the usage of the tools by the test case users. Determine the strengths and weaknesses of each of the tools.
9. **Refine tools and correct defects:** Based on the usability analysis, refine the tool set adding feature and correcting defects as required.
10. **Write final report:** The findings from the above research activities will be summarized in a final report. The final report will describe the additional work that will be necessary to fully deploy the system. In addition, opportunities for expanding the use of the technology will be described for further development. These findings will be presented and reviewed with the sponsor and translated into a plan for the work in following year.

SIMULATION TOOLS

The Sim Builder and Sim Tuner are the focus of this work with respect to simulation tools. The Sim Builder facilitates the development of simulation models within the Experience Accelerator. The Sim Tuner allows the modeler to adjust parameters and variable values so that the model behaves as intended. This section reports on the Part One work of the project on these tools.

In the Experience Accelerator, simulation models are used to represent the state over time of the system being developed in the experience. They track technical performance measures, key performance parameters and quality. They are also used to represent the state over time of programmatic elements of system development, including cost (Earned Value Management), schedules (milestones such as entrance criteria for review meetings) and staffing (including assignment of technical staff to tasks). The simulator executes a model to advance these states over simulated time via behavioral models and relationships. General phenomena modeled include feedback loops, lags and non-linear behavior. Simulator outputs provide the learner with graphical system/program status information for decision support and status data for NPC dialog support. The simulator is implemented in Java™ and XML using the system dynamics formalism.

Simulation models present substantial development and curation challenges. Typically, both expert domain knowledge and simulation modeling knowledge are required. Thus, it is difficult for a novice to create a mode that would apply to an experience of interest to the novice. Additionally, once a model is developed, it must be validated. In the Experience Accelerator, such validation addresses authentic and compelling experience targets and trades them against development time and effort. This requires experimentation with the model to determine that it behaves as intended and in a realistic manner. These issues have been known in simulation and have remained largely unsolved for many years. The intent here is to provide the Sim Builder as a tool to aid an experience developer, with the following features:

- Graphical user interface (GUI) support for model development,
- Support for model curation and documentation via libraries, templates and sub-models, and
- GUI support for output chart design.

At the same time, simulation models have complex parameter and variable relations, making desired output behavior potentially difficult to achieve. In the currently modeled experience involving development of an unmanned aerial vehicle, for instance, there are 185 variables and 172 parameters (constants). The learner can modify a number of variables as decisions entered into the experience to solve problems. There are complex inter-relations among these variables, and these may be complicated by the learner input. The intent of the Sim Tuner is to aid the experience developer in achieving the desired model behavior and outputs, with the following features:

- Graphical output displays for parameter tuning to get desired behavior,
- Graphical output displays to facilitate understanding of learner inputs and their effects on the simulation output, and
- Support for setting difficulty levels for different learners.

SUCCESS METRICS

At the outset of this increment of work, the following success metrics were formulated.

- The simulation tools will provide similar functionality to existing commercially available tools, with the understanding that functionality will be prioritized based on importance subject to funding availability. This includes intuitive support for model building constructs:
    - Dynamics,
    - Dependency,
    - Feedback, and
    - Lags.
- The simulation tools will provide methods for model and sub-model library creation and archiving. A limited library of sub-models will be provided.
- The simulation tools will conform to generally accepted usability principles.
- The simulation tools will be designed to facilitate model building by experience developers who are not simulation experts.
- The tools will support model integration with the Experience Accelerator application.

CRITICAL USE CASES

In analyzing the usage of the simulation tools, we determined that critical use cases fall into the following categories.

1. Conceptual model design and development – Here, the developer is tasked with designing and developing the overall simulation model to be used for a cycle or phase of an experience. This model may have many different components, reflecting technical system behavior, cost and schedule, and workforce activities.
2. Sub-model design and development – The overall model typically can be decomposed into sub-models. This is useful for the sake of modularity and also for facilitating the creation of sub-model libraries that can be used in developing new simulation models.
3. Model element interaction – The simulation tools provide a graphical user interface for model building. The developer thus interacts with different model elements in building a model.
4. Model output specification – The developer is interested in outputs from the model to support learner decisions in the experience. The numeric values of these outputs and their behavior over time are important. The developer also must be able to specify the visualization format to be provided to learners.

Within the category of conceptual model design and development, the following use cases were derived:

1. Designing a new model. The developer starts with a verbal description of the simulated behavior of the program and system under development, as well as the learner decisions that can impact program and system behavior and outcomes.

    a. The developer specifies the different sub-models that will be needed and the relationships between them.

    b. The developer specifies the structure of model elements needed to support each learner decision set.

    c. The developer tests the model, verifies its internal consistency and validates its outcomes against the desired experience/scenario.

2. Evolving an existing model. The developer starts with an existing model that is to be used in an experience that is modified in some manner from the original experience for which the model was designed.

    a. The developer adds sub-models and/or model elements to an existing model.

    b. The developer changes existing sub-models or model elements to make an experience either more difficult or less difficult for a learner.

    c. The developer tests the model, verifies its internal consistency and validates its outcomes against the desired experience/scenario.

3. Deriving a model from an existing model. The developer starts with an existing model and derives a new model that corresponds to a phase in the experience that either precedes or follows the phase to which the existing model corresponds.

    a. The developer determines the common variables, sub-models and other model elements, as well as the new features that the new model must support.

    b. The developer implements the new model.

    c. The developer tests the model, verifies its internal consistency and validates its outcomes against the desired experience/scenario.

Sub-model design and development yielded the following use cases:

1. Create a sub-model. The developer creates a sub-model by adding and linking model elements. Specific model elements subject to learner decisions are developed.

2. Design sub-model interfaces. The developer specifies the interfaces between sub-models.

3. Validate a sub-model. The developer verifies that a sub-model behaves as intended and validates that it accurately reflects the portion of the experience/scenario being modeled.

4. Archive a sub-model. The developer saves a sub-model in an archive where it can be retrieved by other developers who may wish to use it (or a modified version of it) in other experiences.

5. Tune an existing sub-model. The developer determines the combination of variables and parameters that yield the detailed values over time for sub-model outputs.

    a. The developer considers internal parameters for the sub-model's endogenous behavior.

    b. The developer also considers external relations (i.e., exogenous inputs to the sub-model).

6. Add an archived sub-model to a new model. The developer selects an existing sub-model from an archive and inserts it into a model being developed. The developer understands the behavior and function of the sub-model from documentation. The developer modifies its internal workings as needed and links it to sub-models in the current model as appropriate.

The use cases for model element interaction focus heavily on usability.

1. Detailed model and sub-model creation. The developer creates a model or sub-model using graphical elements from a menu.

    a. The developer places elements on the screen in the desired location.

    b. The developer connects those elements to represent linkages and relationships.

    c. The developer manipulates those elements to add or modify names, change locations, etc.

    d. The developer marks the model elements subject to learner influence.

2. Equation specification. The developer specifies math equations for various model elements (rates and auxiliary functions) to model the dynamic behavior and relationships in the simulated program and system.

3. Use of standard application functions. The developer uses standard application functions including:
    a. Save

    b. Edit

    c. Cut-copy-paste

    d. Zoom

Finally, the use cases for model output specification include the following:

1. Chart design. The developer uses a graphical user interface to design charts by specifying:
    a. Variables
    b. Plans (i.e., how critical program/system metrics should unfold during the experience)
    c. Units
    d. Chart labels
    e. Time horizon (amount of time shown on the chart)
2. General output consistency checking. The developer tests the consistency of the actual output with the desired output of the simulation model. The developer adjusts variable and parameter values as needed to get the desired outcomes.
3. Learner-influenced output consistency checking. The developer tests the consistency of the actual learner-influenced output vs. the desired learner-influenced output. The developer tests various values of learner decisions, taking care to study the interaction effects from multiple decisions. The developer identifies ranges of valid learner decision inputs (e.g., new values of simulation variables).

REVIEW OF EXISTING TOOLS

The Experience Accelerator uses the system dynamics simulation paradigm. A number of existing tools use this paradigm. Three of the most prominent commercial tools include iThink/Stella™ [2], Vensim™ [3] and AnyLogic™ [4]. iThink/Stella was selected as the main reference point due to its prominence in the systems dynamics simulation software market and its recent initiative to standardize an XML schema for system dynamics models.

For this increment of work, the way in which iThink/Stella handles sub-models was reviewed and included the following:

- Graphical user interface functionality
- Sub-model structures and interfaces
- XML implementation

In the future, we will investigate the way in which it addresses dependencies of variables.

[2] www.iseesystems.com
[3] www.vensim.com
[4] www.anylogic .com

REQUIREMENTS

Requirements are divided into three main categories – graphical user interface, sub-models and parameter tuning. The GUI requirements are the following:

1. The graphical user interface will be similar to Microsoft Visio™.
2. It will support the functionality expected from a graph builder application, including
    a. The ability to create a new graph,
    b. The ability to insert specific nodes,
    c. The ability to specify properties that belong to those nodes,
    d. The ability to zoom the overall graph,
    e. The ability to verify that a graph is correct and to provide feedback if the graph has not been created correctly,
    f. Facilities for creating charts from the simulation output, and
    g. The ability to execute all standard program functions (open/save/create new files, cut, copy, paste, and changing language)

The sub-model requirements are as follows:

1. The user will be able to section a graph into sub-models.
2. The GUI will show how sub-models are linked.
3. The GUI will allow the manipulation of each sub-model on screen.
4. Sub-models will be able to be saved for use in other models

Finally, requirements related to parameter tuning in the Sim Tuner include the following.

1. The tools will have the ability for the user to change multiple variables and have the corresponding graphs automatically updated based on the changes.
2. The graphs must be readable on the screen.
3. The tools will display multiple graphs to illustrate how different variables change over time.

TOOL DESIGN AND DEVELOPMENT

This effort started with an existing simulation model-building tool with a graphical user interface of limited functionality. The focus of this increment has been to improve the usability of the Sim Builder tool, to add sub-model support, and to perform design work for the Sim Tuner.

Since this work references the building blocks of system dynamics simulation models, a summary of these building blocks is provided here.

- Level nodes (variables or stocks). These variables change over time by various rate functions during a simulation run.
- Constant nodes. A constant node contains a value that does not change during a simulation run.
- Rate node. A rate node specifies an equation resulting in a rate of change (either increase or decrease) of a level node. This rate can be conceptualized as flow into or out of the level node "stock."
- Auxiliary node. An auxiliary node contains an intermediate function for use elsewhere in the model.
- Source/sink node. Flows from level nodes may go to or from other level nodes. They may also go to a sink node or emanate from a source node.

USABILITY

We focused on three usability heuristics from Nielson[5] for the work associated with usability of the Sim Builder and Sim Tuner:

- User control and freedom (e.g., making sure user can exit unintended state without having to go through an extended dialog),
- Consistency and standards (e.g., platform-specific standards for design), and
- Error prevention.

The first step involved a usability assessment and associated improvements of the existing Sim Builder tool. The GUI for the Sim Builder is shown in Figure 2.

[5] http://www.nngroup.com/articles/ten-usability-heuristics/

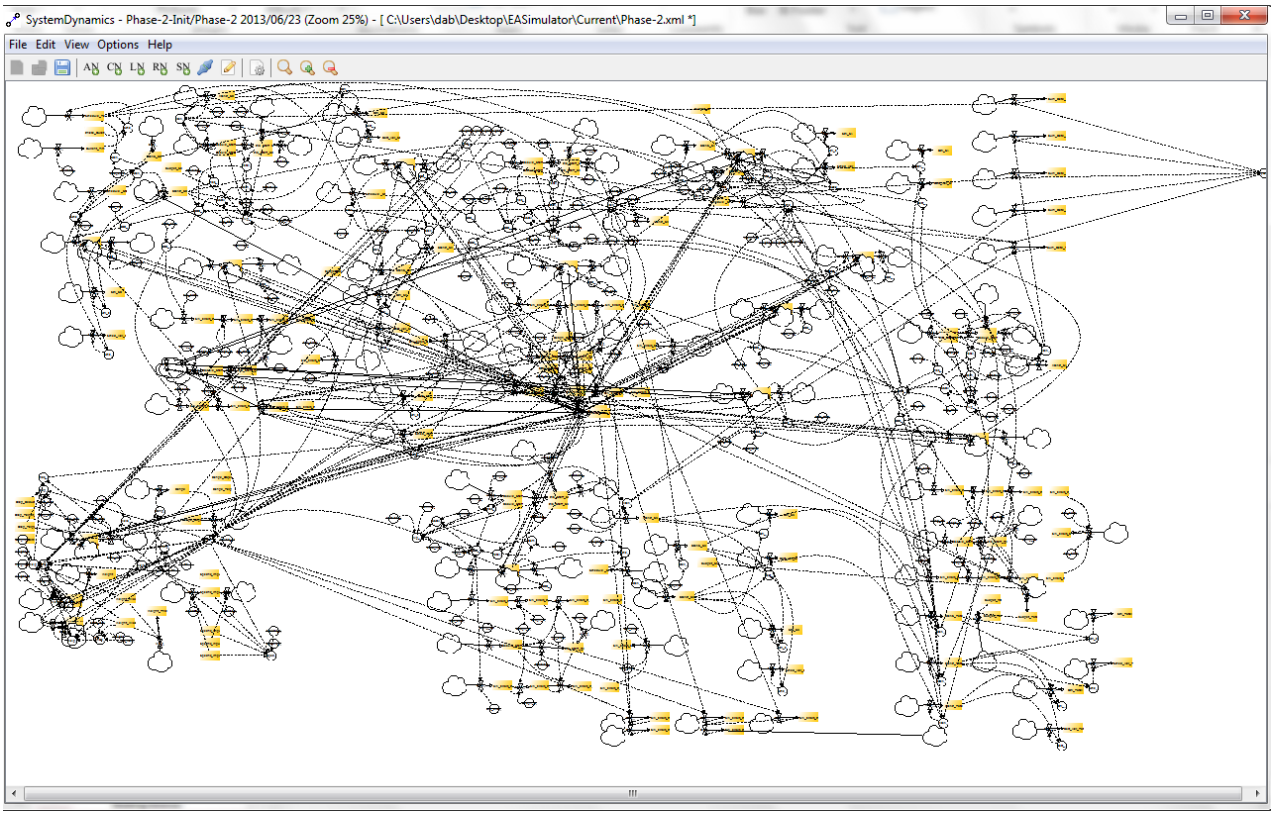

Figure 2. Sim Builder interface

The result of the assessment, as well as work to correct issues, is discussed below. Table 2 addresses issues involving the file menu of the interface. Table 3 shows issues and solutions associated with other menu items. Finally, Table 4 presents issues and solutions for the graph elements.

Table 2. Usability of file menu

Problem	Details and Solution
Deleting files and folder in "open" option	Details. If the "new folder" button is clicked by mistake, and the user then wants to remove the folder created by mistake, there is no option for deleting the folder from within the dialog. The delete key did not work, and "delete folder" was not an option when the user right clicks. Solution. Implement both a delete key and a delete option in the context menu that allows user to delete a file.
Exception when using "save" for invalid graph not communicated to user	Details. If the user tries to save a graph having only a source sink and a level node (variable) not connected with each other, an exception occurs due to an invalid graph. However, there is not message to the user. The application simply does not save the model. Solution. Isolate where this error is coming from and trap it so that it shows in an error message.
Verify and save in "save"	Details. The application allows the user to save a model only if it is "valid" (i.e., the model is executable with all nodes being connected properly). It does not allow the user to save an intermediate model, which could potentially result in lost work. Solution. Separate "save" and "verify."

Interaction with other aspects of application	Details. Usually, if the user tries to open another file with one currently open, an application will prompt to save any unsaved changes in the current file. After the user chooses whether to save changes or not, the user will be redirected to the other file. However, the open option is currently greyed out as long as a new or existing model is displayed on the screen. The user has to close a file before opening another. Solution. Enable file "open" option when an existing graph is displayed. Implement "save unsaved changes" dialog box before opening other graph.

Table 3. Usability of other menus

Problem	Details and Solution
Cut/copy/paste	Details. The existing application does not have cut, copy and paste functionality. This is a major drawback to its usability. Solution. Implement functionality for cut, copy and paste.
Switching from Sim Builder to Sim Tuner	Details. If the user clicks "execute model" in the Sim Builder to go to the Sim Tuner, the application does not allow return from the Sim Tuner to the Sim Builder. All menu options are greyed out except for "save," "save as," and "exit" on the file menu. Solution. Implement "cancel" button that allows user to return to the Sim Builder from the Sim Tuner.
"Add flow" mode	Details. The "add flow" mode works properly when the user wishes to connect a source/sink and level nodes, but the menu choice icon is not intuitive. Solution. Change the icon so that when this mode is on, it displays a connected "plug," and when not in this mode, it shows an unconnected "plug."
Zoom in "view" menu	Details. Granularity of zoom in/out is too coarse (only allows 2x and 0.5x). Solution. Provide more granularity.
Spanish language option	Details. The letter "ñ" shows up as "?" when the Spanish language option is selected. Solution. Display correct letter.

Table 4. Graph symbol issues

Problem	Details and Solution
Symbol usage	Details. The cloud shape for source/sink nodes is too large as compared to the rest of the shapes on the graph. This clutters the interface. The constant symbol and auxiliary node symbol are the same, so it is difficult to see at a glance which circle corresponds to a constant and which corresponds to an auxiliary node. Solution. Shrink the cloud shape to be more proportional with rest of graph. Make the constant symbol a circle within a square.
Text on symbols	Details. The text on the rate node is unreadable because there is a flow arrow over top of it. On some of the auxiliary nodes, only the first three letters of the text are displayed with "…" afterwards. This makes it difficult to understand what the node represents. For example, in the current UAV experience, there are numerous auxiliary nodes that have the first three letters "CDR." Even though they are labeled differently after the first three letters, they look the same on the interface. Solution. Remove the text completely. Provide all descriptions in the tooltips. Or put the text to one side of the symbol. This could make the interface more cluttered. (This item is under development.)

Position of shapes	Details. In the current experience model, many symbol nodes are placed on top of one another. This is an artifact of the model being developed outside the previous Sim Builder due to its usability problems. Solution. Investigate whether the graph representation can be changed to automatically space symbols better. (This item is under development.)

SUB-MODELS

One of the major tasks in this increment was to implement capability for sub-models. These are needed to promote modularity and re-use of models. The existing Sim Builder and simulation execution did not provide for sub-models, only for one main model. This is problematic when the user is developing a large complex model, as shown in Figure 2. This work consisted of two main components:

- GUI design and development, and
- Changing the XML schema to allow sub-models, as well as modifying the back-end code to allow reading and savings of files in this new schema.

The GUI design shows different sub-models in different sections of the screen. The user can create a new sub-model by clicking a new "add sub-model" button on the menu. The application automatically sub-divides the screen to accommodate the new sub-model, and it automatically generates a different color for the boarder of each sub-model as its reference color. In addition, a number is assigned to each sub-model. The number displays on the toolbar when the user mouses over that sub-model. An example model with six sub-models is shown in Figure 3. In this display, the sub-models are numbered from 1 – 6.

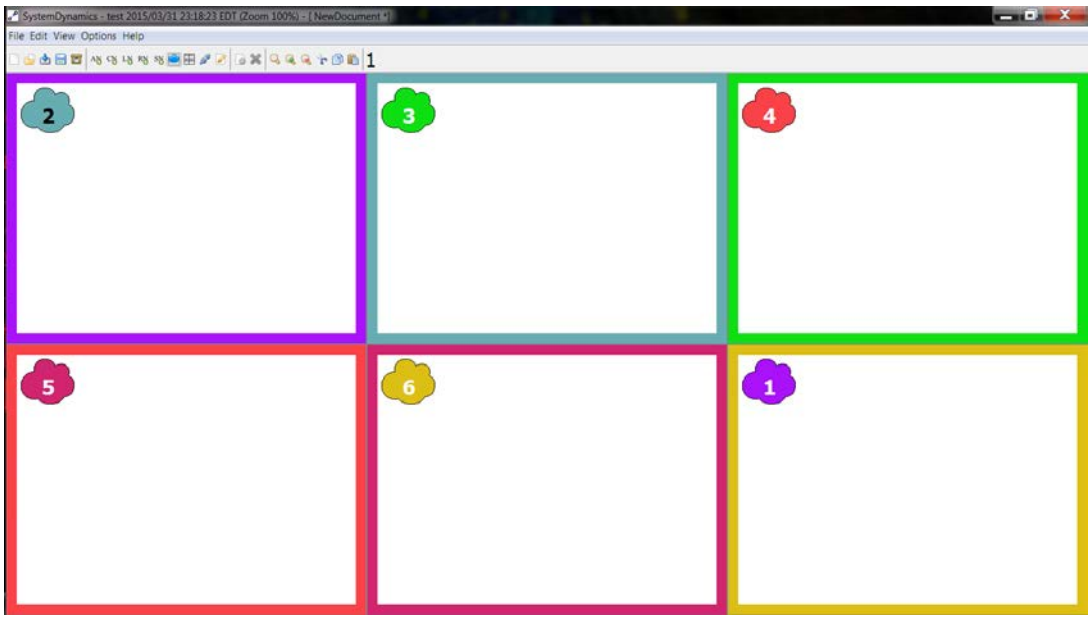

Figure 3. Sub-model interface

Within each of the sub-model panes, the user can create a sub-model consisting of the various model elements linked to one another. When inserting a model element, the application

prompts the user for which sub-model should receive the element. The element is then placed in that sub-model. Additionally, the user can place a shared source/sink node in a sub-model. This shared source/sink node transfers flow between sub-models, and it is characterized by a color corresponding to the linked sub-model. The figure shows a shared source/sink node in each of the sub-model panes. The shared source/sink node in sub-model 1 (upper left plane with purple border) directs flow between sub-model 1 and sub-model 2. Hence, it is shown in green, corresponding to the border color of sub-model 2. Thus, the numbers in the shared source/sink nodes do not reflect the sub-model in which they are located, but rather the sub-model to which they connect.

The GUI is designed such that one can easily determine sub-models since it associates a different color with each sub-model. Numbers were also added (rightmost toolbar item and on top of the colored source/sink nodes) in order to design for colorblind individuals.

A major task involved designing the underlying XML schema to facilitate creation of sub-models, saving a group of sub-models as a model, and archiving individual sub-models to be imported later. This involved creating an intermediate sub-model tag underneath the overall model tag, and done before the content in each model. The sub-model tag has attributes dictating its color so that it would be appropriately colored when the whole model was opened up again. Additionally, each colored source/sink node has its respective color as an attribute of its tag as well.

In the future, shared nodes will be added for other model elements such as variables and constants so that a variable, for instance, can be defined in one sub-model but referenced in another. Note that the variable would be changeable only in the sub-model in which it is defined.

SIM TUNER DESIGN

The current Sim Tuner graphs each of the variables (level nodes) from a simulation model on one main chart. For large models (e.g., Figure 2), this is quite cumbersome. Figure 4 shows the Sim Builder interface for a simpler model that addresses weight and range of the unmanned aerial vehicle in the current experience.

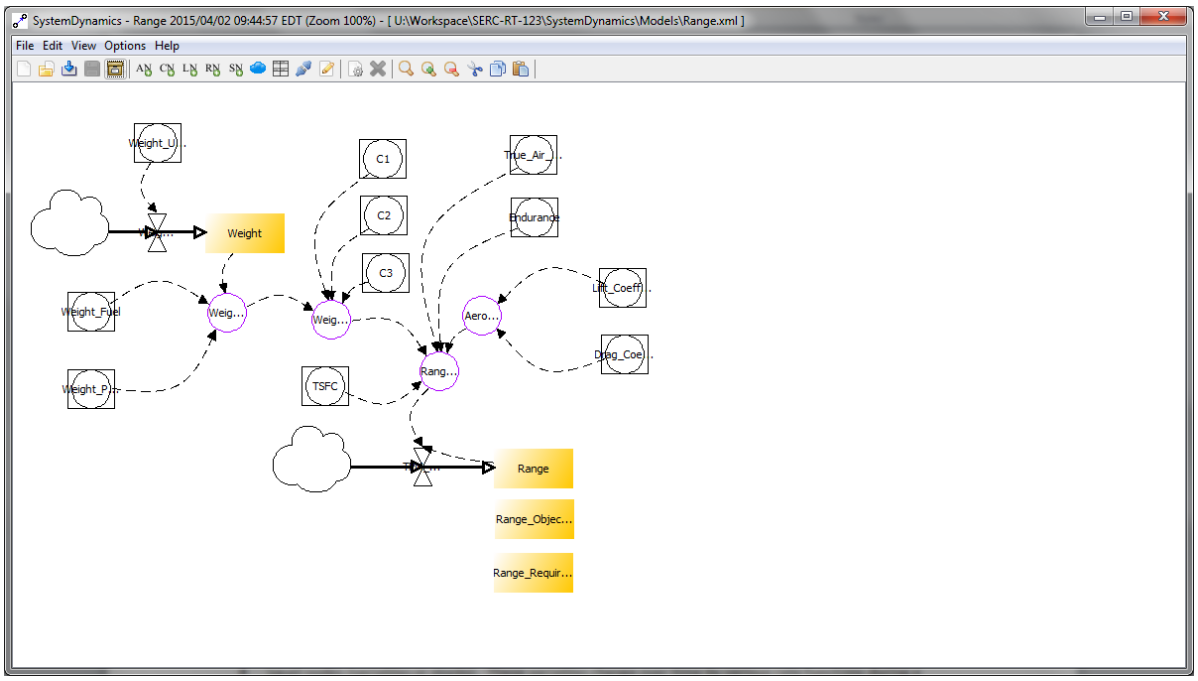

Figure 4. Range model

Figure 5 shows the Sim Tuner interface for this model. There are three range-related variables – the range (estimate for the range), the range objective (desired range) and the range requirement (minimally acceptable range). The weight is also a variable, and as shown in the output chart, it is increasing over time. Thus, the range decreases, since weight and range are inversely related. Other technical performance measures that affect range are held constant in this model (e.g., drag, propulsion efficiency, lift, etc.). The main y-axis on the left references the range-related variables. There is an option to include additional y-axes for other variables that scale differently, such as weight. These are displayed on the right, as shown in the figure. While this feature is useful, future work will involve having more charts with options to include only certain variables.

In addition, some features of the chart output generation from the simulator module in the SEEA execution platform can be used here, such as the scaling of the y-axis such that only the bottom and top 10% of chart real estate is unused. Finally, the Sim Tuner needs more of an experimental functionality so that different variable and parameter values can be explored in different combinations. Figure 6 shows a design for a future version of the Sim Tuner. It presents an interface similar to Microsoft Excel™ for manipulating variable and parameter values, with different graphs shown using a tabbing approach. Different graphs could potentially correspond to sub-models or to output charts for the Experience Accelerator.

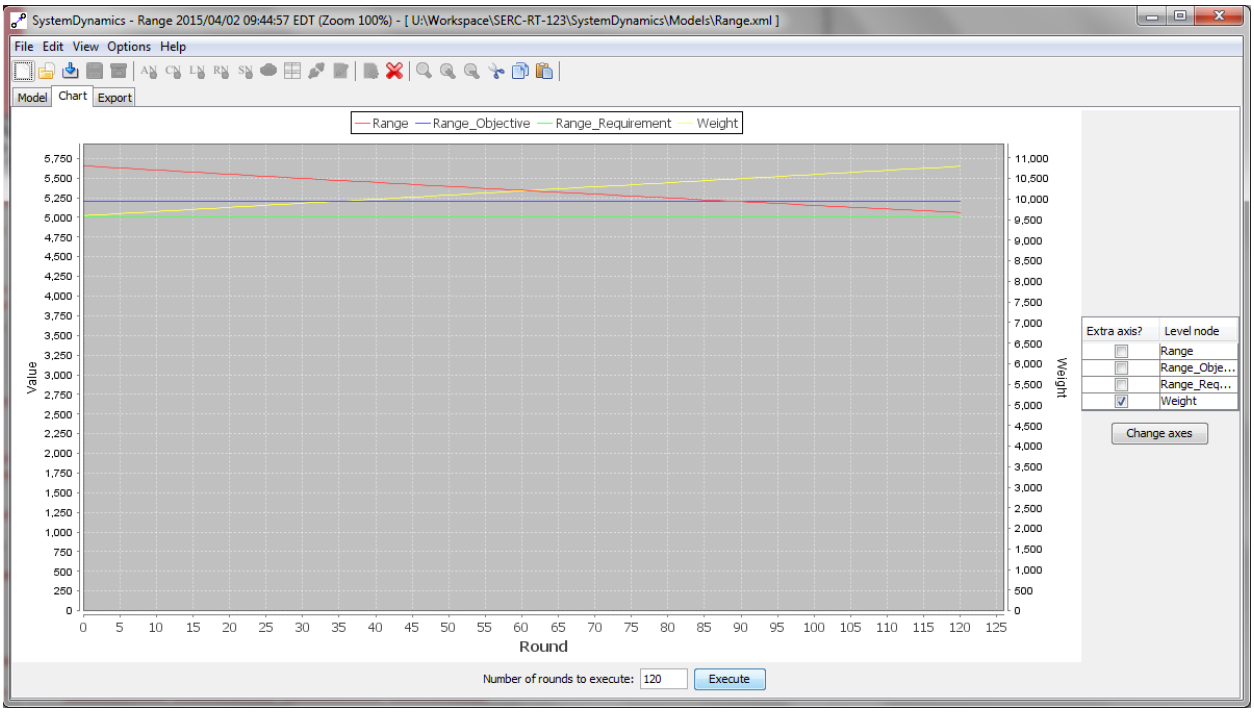

Figure 5. Sim Tuner interface

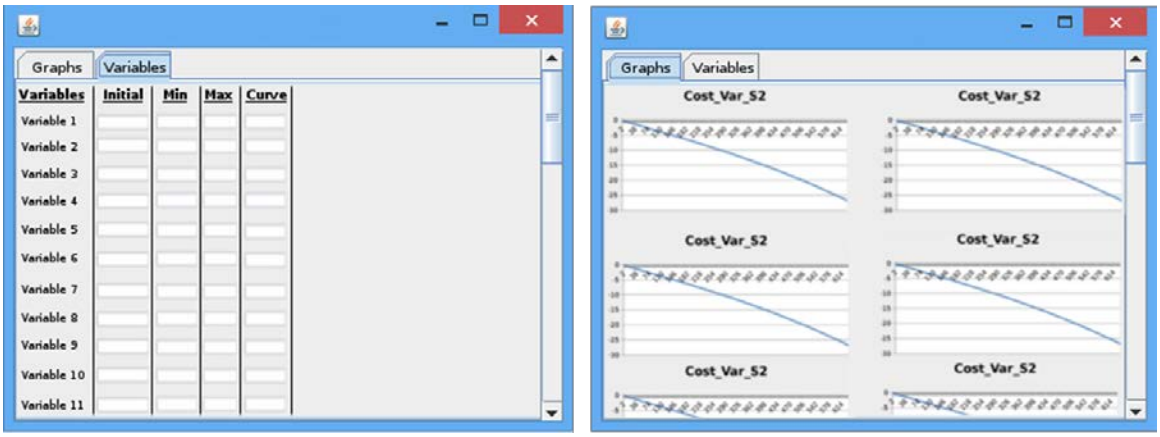

Figure 6. Design for future Sim Tuner

EVALUATION

Evaluation of the Sim Builder and Sim Tuner has been conducted internally and has identified several issues that were addressed. Future evaluation work will test usability and effectiveness of the tools as potential experience developers use them.

- The application would not validate a model that has an empty sub-model.
- Cut-copy-paste did not work for shared source/sink nodes.
- Flow could not be added to shared source/sink nodes.
- For models having sub-models, the switch back function from the Sim Tuner to the Sim Builder did not bring up the whole model. It methodologically displayed only the first.
- Various error messages were cryptic.

EXPERIENCE BUILDING TOOLS

The following is a description of the Experience Building tools, namely the Phase and Event Editors, and Artifact Integrator.

UAV PROJECT AND SYSTEM LIMITATIONS

The SEEA provides a simulated experience of training system engineers based on an UAV project. All the phases are designed to practice managing the UAV project. We can see that all the data entries are set specifically for the UAV in Figure 7.

KPPs and TPMs	Range	Drag Coeff.	Propul. Effcy.	APS Weight	CCS Weight	Total Weight
	5205.69 ↓	2.64 →	0.00 →	7278.91 ↓	2600.93 ↓	9877.19 ↓
Contract Performance EVM ($M)	**Cur. Burn Rate**	**BAC (Phase2)**	**EAC (Phase2)**	**Difference**	**Result (End of Phase 2)**	
	6.45	195.00	166.83	-28.17	-------	-14%
	Sched(SV)	**Sched(SPI ratio)**		**Budget(CV)**	**Budget(CPI ratio)**	
Overall	8.28 ↑	1.13 ↓		10.47 ↑	1.17 ↑	
UAVenture (Prime)	5.65 ↑	1.43 ↓		4.04 ↑	1.27 ↑	
AirGo (Airframe and Propulsion)	2.03 ↑	1.15 ↓		1.09 ↑	1.08 ↑	
FlyWire (Command and Control)	-0.53 ↓	0.98 ↓		4.57 ↑	1.16 ↑	
Terra Firma (Ground System, Launch/Retrieval)	1.14 ↑	1.23 ↓		0.81 ↑	1.15 ↑	
CDR Requirements (% complete)	**Test Reqts**	**Digital Data Files**	**SW Dsgn Descs**	**Struct. Lds.**	**SS Dsgn&Ver**	**V&V of SIL**
	33%	43%	32%	34%	32%	47%
FlyWire SW Quality (# of Defects)	**Outstanding**	**Detection Rate**	**Removal Rate**	**Projected Time To Remove**		
Critical Defects	14 ↓	12.0 def/mon ↑	8.7 def/mon ↑	1.6 Months ↓		
Non-critical Defects	215 ↓	194.3 def/mon ↑	168.7 def/mon ↑	1.3 Months ↓		
Red - beyond Threshold or > 15% worse than plan, Yellow - 10-15% worse than plan, Green - within 10% of plan						
Change from last cycle: ↑-better →-same ↓-worse						

Figure 7: Dashboard specified for UAV project

While the SEEA is useful for training system engineers through this UAV project, it has limitations in changing experience content. The structure of the system is customized for the UAV project; it is difficult to replace the experience contents or create a new project without touching the code on both the server and the client sides. The SEEA was developed with a goal of transitioning to an open-source sustainment model which will provide long-term support for a community of educators and learners in creating learning exercises to address their specific needs (Wade, 2014). The legacy system does not allow for this flexibility. It is necessary to improve the expandability of the system, to make it more usable and reusable. In order to make improvements to the system, the current mechanism of storing and processing experience contents must be addressed. The following sections will demonstrate how the legacy system works in the handling of experience content.

PHASE

Similar to how a game story is divided into levels, the SEEA experience is divided into phases. A phase may contain sub-phases which may consist of several cycles. How an experience progresses through phases is shown in Figure 8.

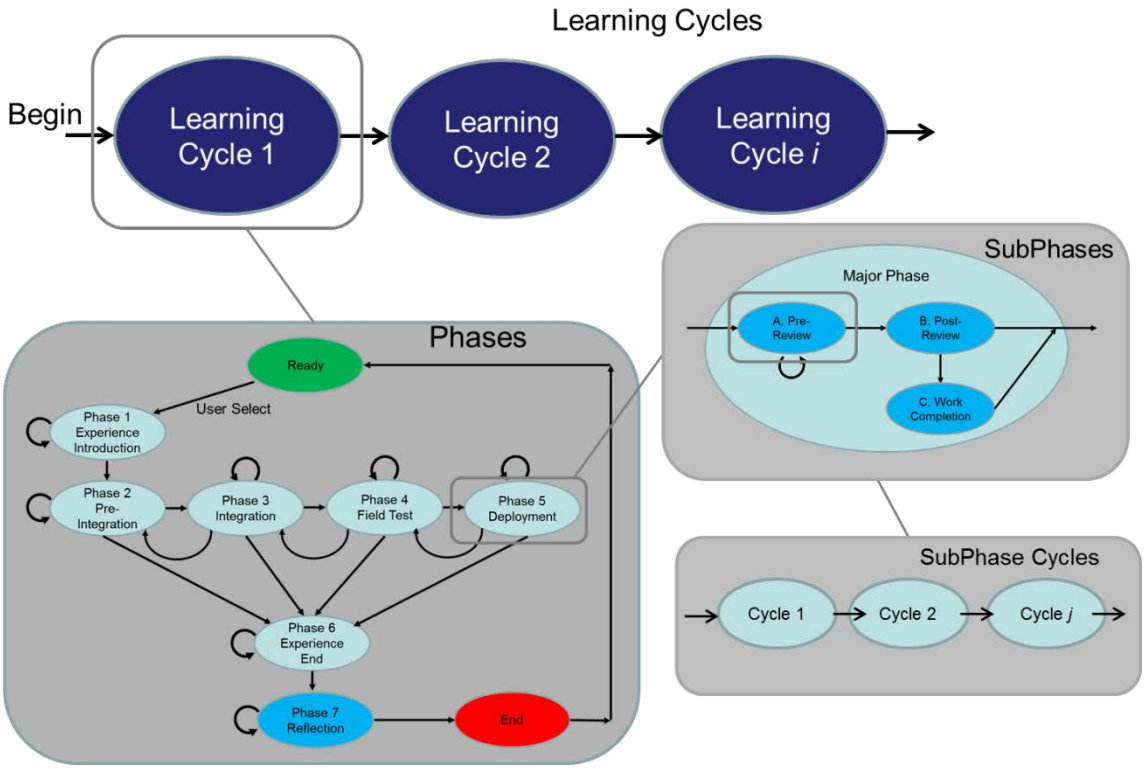

Figure 8: An experience progresses through phases

The current UAV experience is composed of the eight phases listed below:

- Phase 0 is where the user is introduced to the SEEA and must take a survey to advance to the next phase.
- Phase 1 is an introduction to the experience.
- Phases 2 through 5 are the main body of the experience where the user must make decisions that affect the success of the simulated project.
- Phase 6 is the end phase.
- Phase 7 is the last phase in the experience where final feedback and overall outcomes are presented.

The user can progress to a new phase when the current phase transition condition is achieved. Usually a recommendation form is required in order to move forward. An event can also trigger transition, as can open specific documents, or assign certain values to state variables. A phase can also be skipped if the demanded condition is not met. For example, if the user fails to perform well in a phase and cannot fulfill the transition condition, he or she will be put directly to the final phase with a feedback of project fail.

The sequence of phases with the transition conditions is hard-coded on the server side. There is a class called *ExperienceMaster* that controls this, by using a method *doPhaseTransition*, which handles the order and condition of phases for the whole UAV project. Phases switch in an inflexible way under such design. For each phase, there is an XML file holding its information on the server. The system loads and parses these XML files for setting up the corresponding phases.

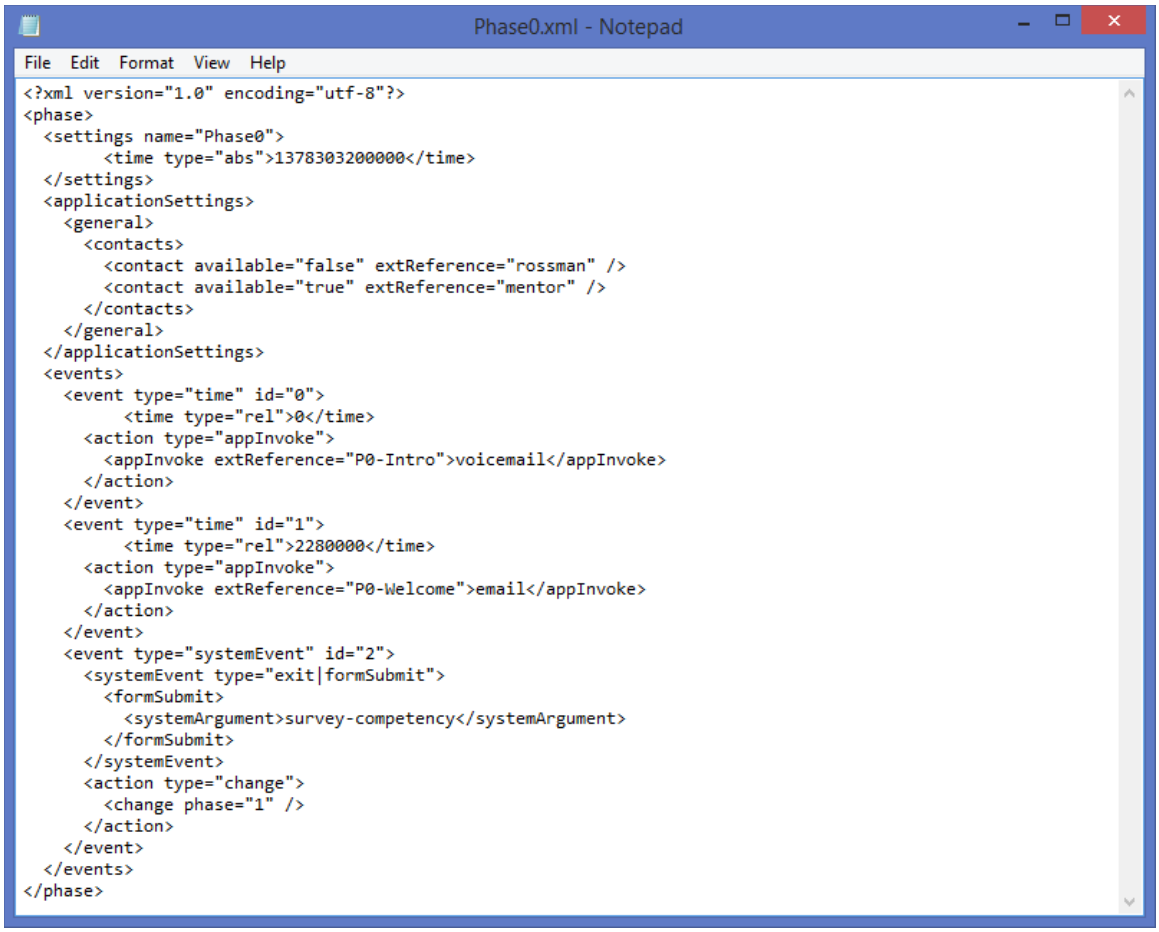

Figure 9: An example of phase XML file

The content of a phase XML file is shown in Figure 9. The file contains attributes and settings for the phase, for example, it has the events, which are going to be triggered during the phase. With the current phase XML file, the structure of each phase can be modified outside the system.

EVENTS

Events are specified in XML files on the server that contain the conditions necessary to trigger the event and the actions that take place when triggered. Conditions that can trigger events include:

- Experience time
- Opening documents
- Submitting forms
- State values such as simulator outputs or success status

The following are some of the actions that can happen as a result of an event triggering:

- Sending an email to the user from an NPC
- User receiving a voicemail from an NPC

- Initiating a call (dialog) to the user from the NPC
- Sending pop-up messages to the user about experience status
- Transitioning to other phases/sub-phases/cycles
- Making state variable changes usable by other parts of the system

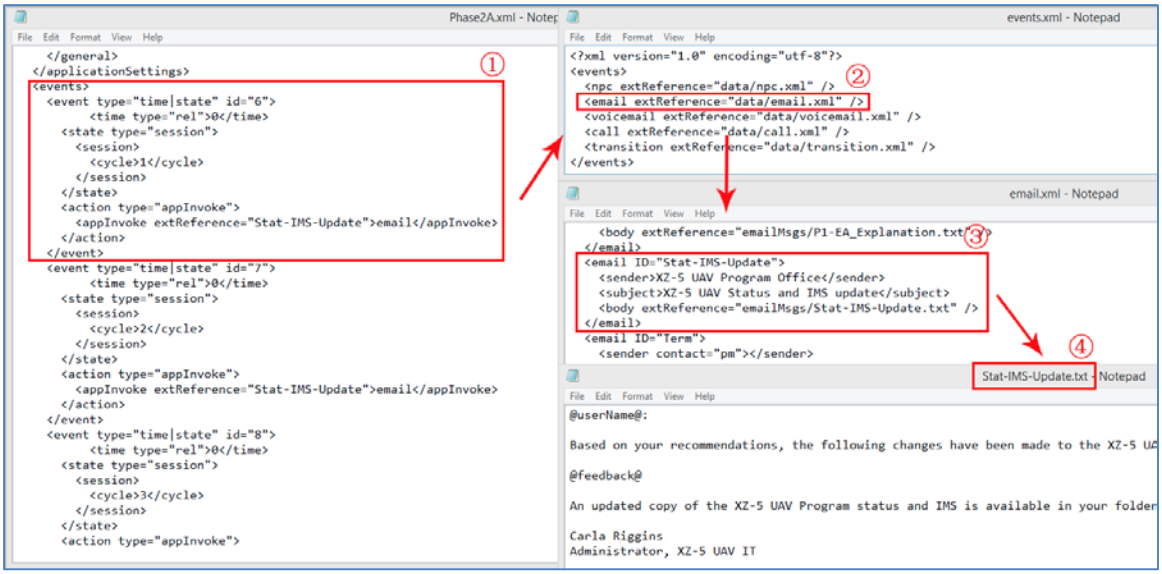

Figure 10: Event and action

Figure 10 shows how an action is taken place when an event is triggered. Some events are shown in Phase2A.xml (which is a sub-phase of Phase2); one of them is “email event” (Part 1 in Figure 10) with an action “appInvoke”. This event can be found in events.xml (Part 2 in Figure 10). The external reference of this event has a value “Stat-IMS-Update” which points to email.xml (Part 3 in Figure 10). The external reference of “Stat-IMS-Update” in email.xml indicates that the system can finally find and invoke an email with the content stored in Stat-IMS-Update.txt (Part 4 in Figure 10). Therefore when the email event is triggered in Phase2A, an action will be executed which sends an email from NPC Carla Riggins to the user.

The mechanism of storing event information in XML files allows modifying events outside the system. However since one event relates to several different files, this process would be tedious and fallible.

ARTIFACTS

‘Artifact’ is a general term for the files that can be accessed by the user within the SEEA virtual file system for project information, as well as the files that are loaded into the server to create experience context. There are two kinds of artifacts in SEEA server file storage distinguished by functions, which are static artifacts and configuration artifacts.

As shown above in Phase and Event, some of the experience content files are stored on the server. These files are either read in and parsed by the Experience Master on the server or transferred to the client for viewing. The following is a list of content files held on the server:

- PDFs for static artifacts
- Xml files for configuration, events, and dialog
- Txt files for email and voicemail content

The PDFs for static artifacts are the files in the system, from which the user will read, and learn and gather information. These artifacts contain descriptions about the experience, project, and background information that may be useful to the user. They may also contain images or graphs if desired by the content creators. Because these documents provide information that does not change throughout the experience, they exist as PDFs that get sent to the client for viewing when requested. Figure 11 shows the content of a static artifact.

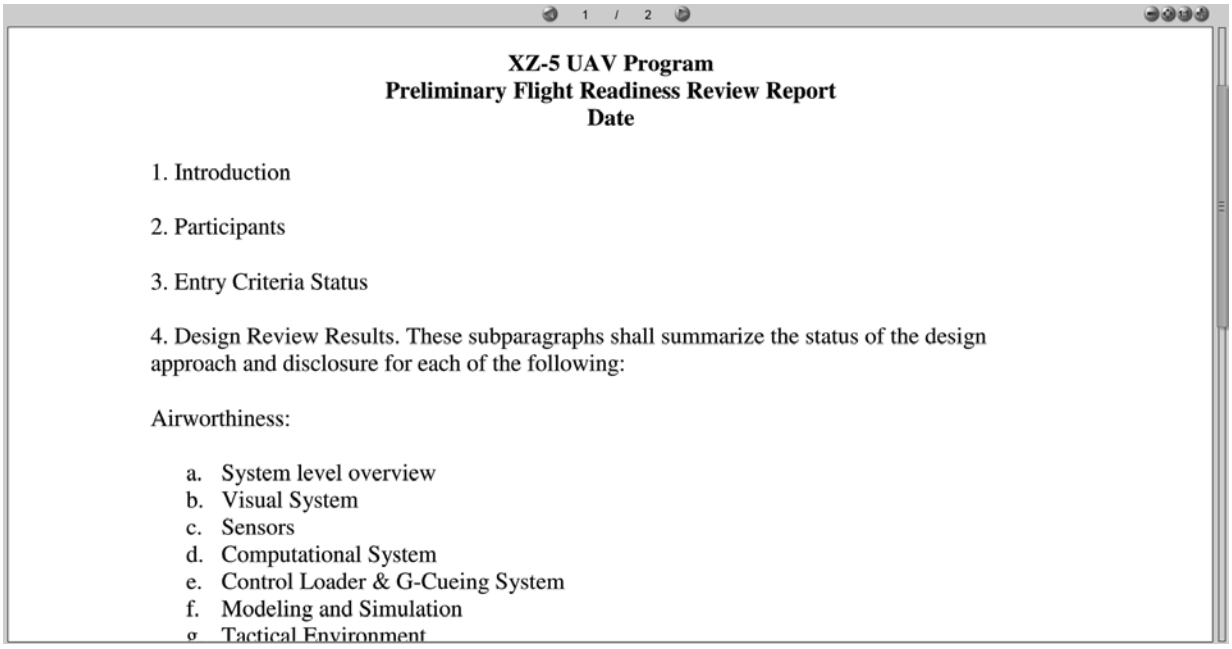

Figure 11: The content of a static artifact

All the static artifacts have a suffix of SWF. This is because that PDF files cannot be shown directly in Flash client. So these files must be converted to Flash compatible versions with an external tool. The tool pdf2swf is used for the conversion in current system. Since the number of static artifacts may be large, the process would become a heavy workload.

Most of the Xml and Txt files are configuration artifacts. As we have seen from Figure 10, triggering one event may relate to four different configuration artifacts. This leads to a deficiency on managing artifacts in that it is very complex to change or create new configuration artifacts, because files may be linked with each other. This means, relevant configuration artifacts should be changed or created together as a pack.

PROPOSED EXPERIENCE DEVELOPMENT TOOLS

The current pattern of experience contents handling is designed for a specific project. Replacement for the experience contents relies primarily on code changes. Some items related to experience contents are file based, which can be managed outside the system, but the work

would either be tedious and fallible or complex and heavy. It is necessary to have tools to help handle experience contents more efficiently. Based on the previous work, three tools are designed and developed in this project. These three tools are used together to provide non-technical staff with the ability to build and modify SEEA experiences without any programming.

The Phase Editor is a tool that provides the ability to alter the finite state machine that changes the phases within an EA experience. For example, the project phases can be customized to new domains and environments and can be constructed to represent state changes that are not affiliated with formal project states. Figure 12 shows a screenshot of the Phase Editor.

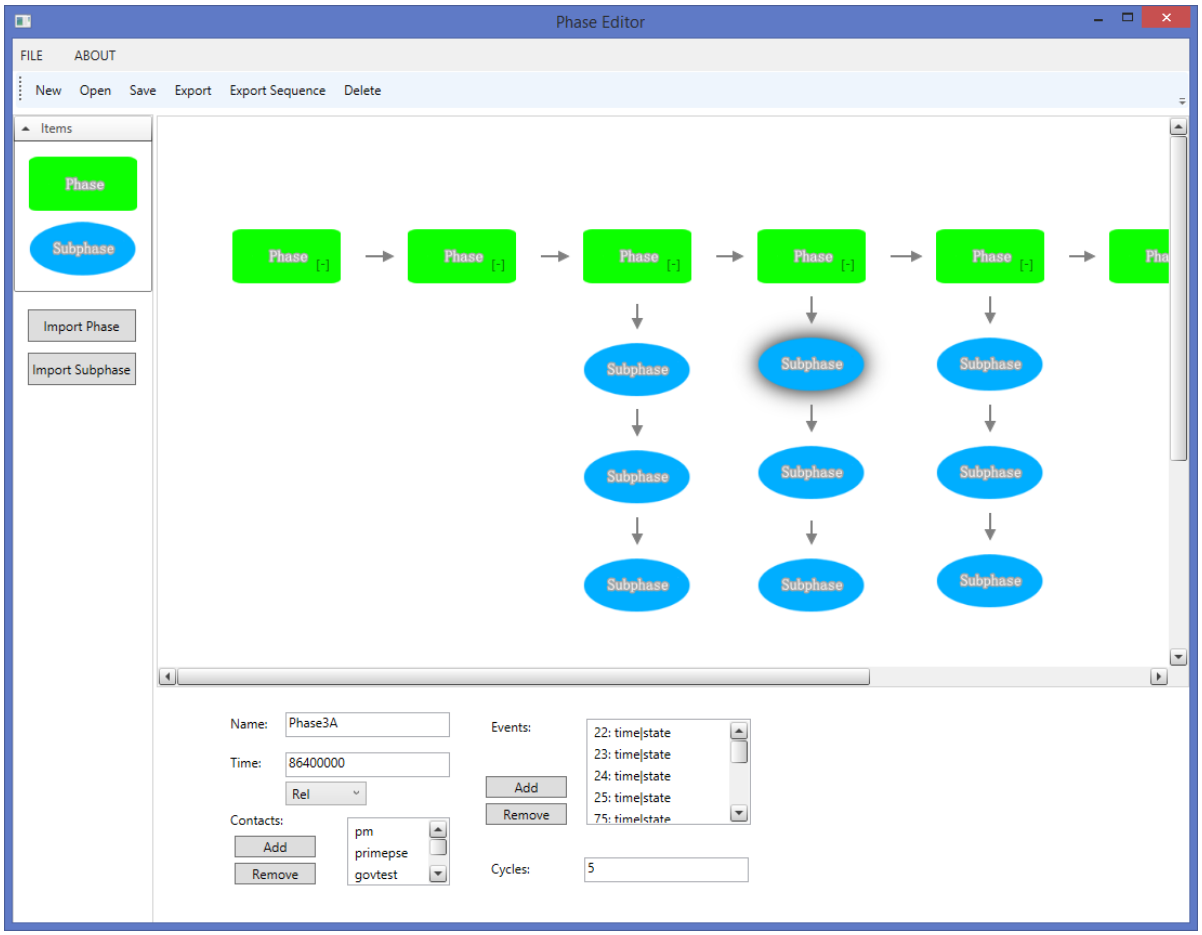

Figure 12: A screenshot of Phase Editor

The Event Editor is a tool that provides the capability to create and edit events during an experience and the activities that may trigger them. For example, a phone call from the learner's supervisor can be triggered based on a decision made by the learner or the state of the project. Figure 13 shows a screenshot of the Event Editor.

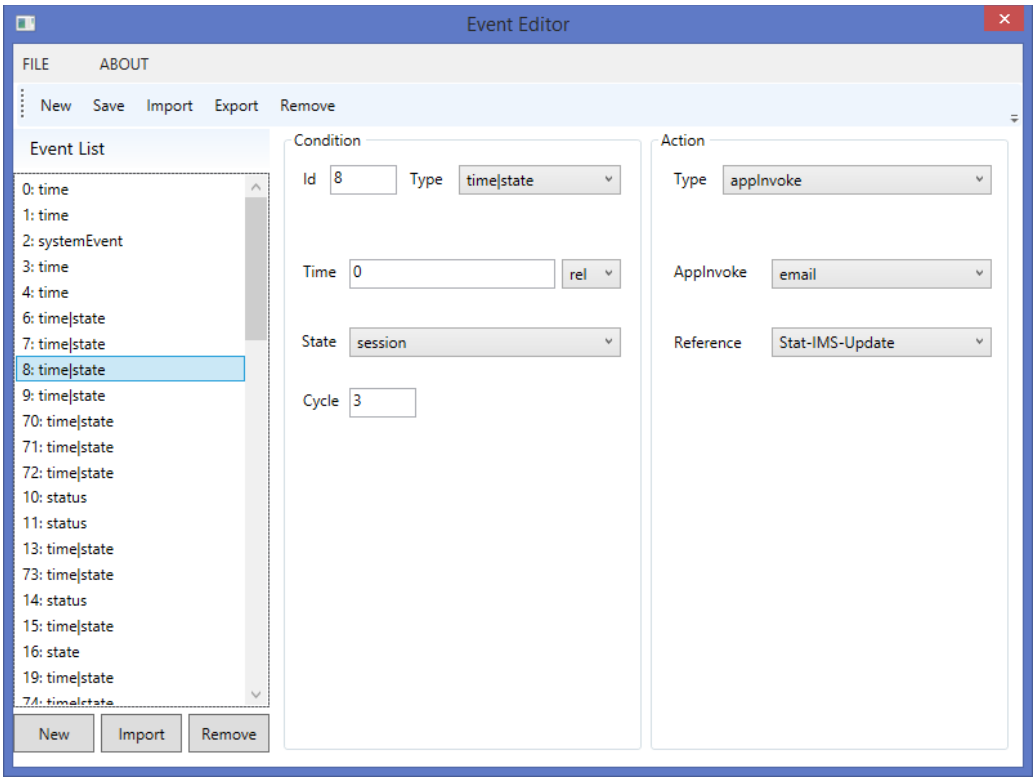

Figure 13: A screenshot of Event Editor

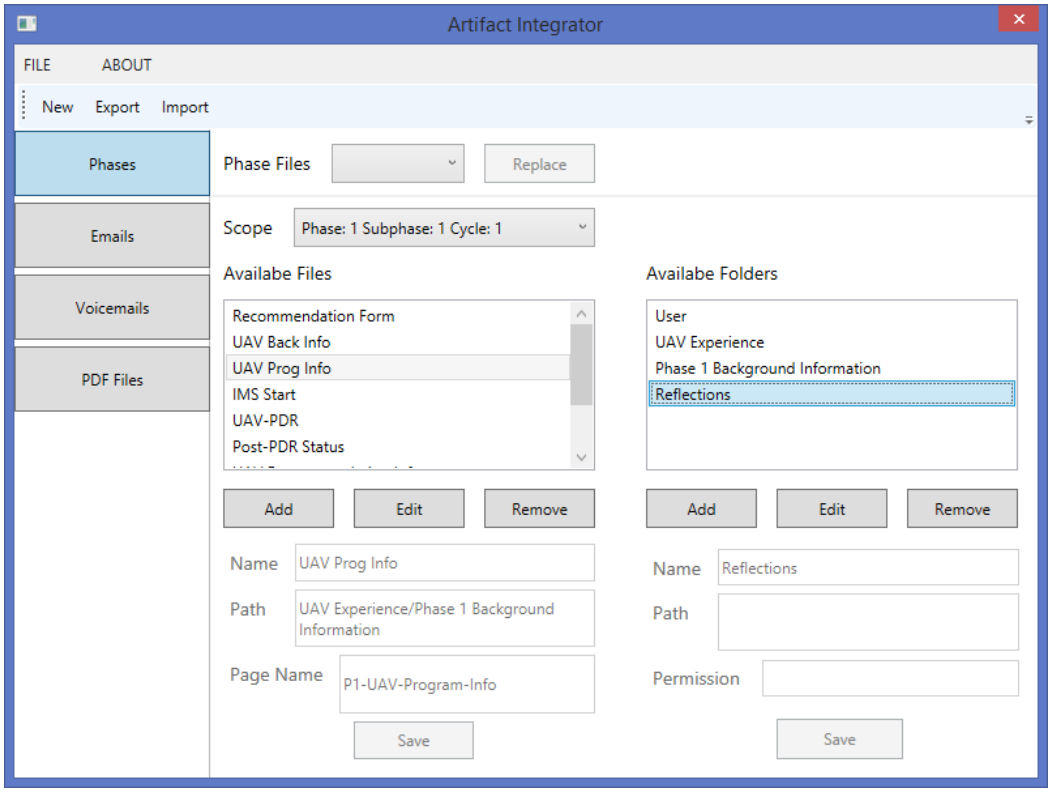

Figure 14: A screenshot of Artifact Integrator

The Artifact Integrator is a tool that provides an experience builder with the ability to quickly upload an experience change, be it a new artifact such as a new dialog or report, or a changed phase and or event, and test the results without having to do any programming. Figure 14 shows a screenshot of the Artifact Integrator.

REVIEW EXISTING TOOLS

The following is a review of existing tools similar in purpose to the proposed experience development tools.

GAME-BASED LEARNING

Game-based learning (GBL) is a type of game play that has defined learning outcomes. Generally, game-based learning is designed to balance subject matter with gameplay and the ability of the player to retain, and apply said subject matter to the real world. There are a large number of research papers on GBL topics. Many of them are related to System Engineering, Software Engineering and Civil Engineering, but most of those systems are designed as fixed-content games. Tools (or engines) for developing educational games have been discussed in several references (Morsi et al, 2007). Few of them mention building tools for content expansion on existing systems.

The Emergency Management Staff Trainer (EMST) (http://www.train-emst.com), is a simulation-based training system, which is sponsored by the National Guard Bureau and contains emergency management scenarios for National Guard, state, and local responders, which allows individuals or teams to make decisions in realistic situations and see the outcomes. In EMST, a Scenario Builder is utilized. This Scenario Builder contains a few editors, such as *Scenario Properties Editor*, *Time Segment Wizard*, *Inject Wizard*, and *Email Response Editor*.

The Scenario Builder gives users the capability to design EMST training exercises that are completely authentic to the user's organization. With access to the Scenario Builder, a user can author training exercises to mimic specific incidents that the user's organization has faced in the past and design incidents that gauge training for the user's organization's response to future threats. Since EMST exercises are geographically distributed, authoring or tailoring exercises in the Scenario Builder allows a user's organization to utilize the intelligence gained in the response to a real emergency in any city or state and apply it to training exercises hosted anywhere in the country. This increases not only the scalability, but the lifespan of training exercises as a user can update scenarios at any time with new lessons learned.

With this tool, it is very easy to apply new scenarios to a virtual exercise, which shows the benefit of using a powerful tool to extend the learning experience. Figure 15 shows the User Interface of this tool. It is similar to the design concept for experience tools developed in this research. This provides support for the efficacy of developing such tools. In our design, we are going to build a tool set similar to the Scenario Builder, with which educators and developers outside the SEEA research and development team may design and develop educational content. Tools in our

toolset are used together to build and modify the major factors of SEEA experience, in order to provide diversified content to SEEA.

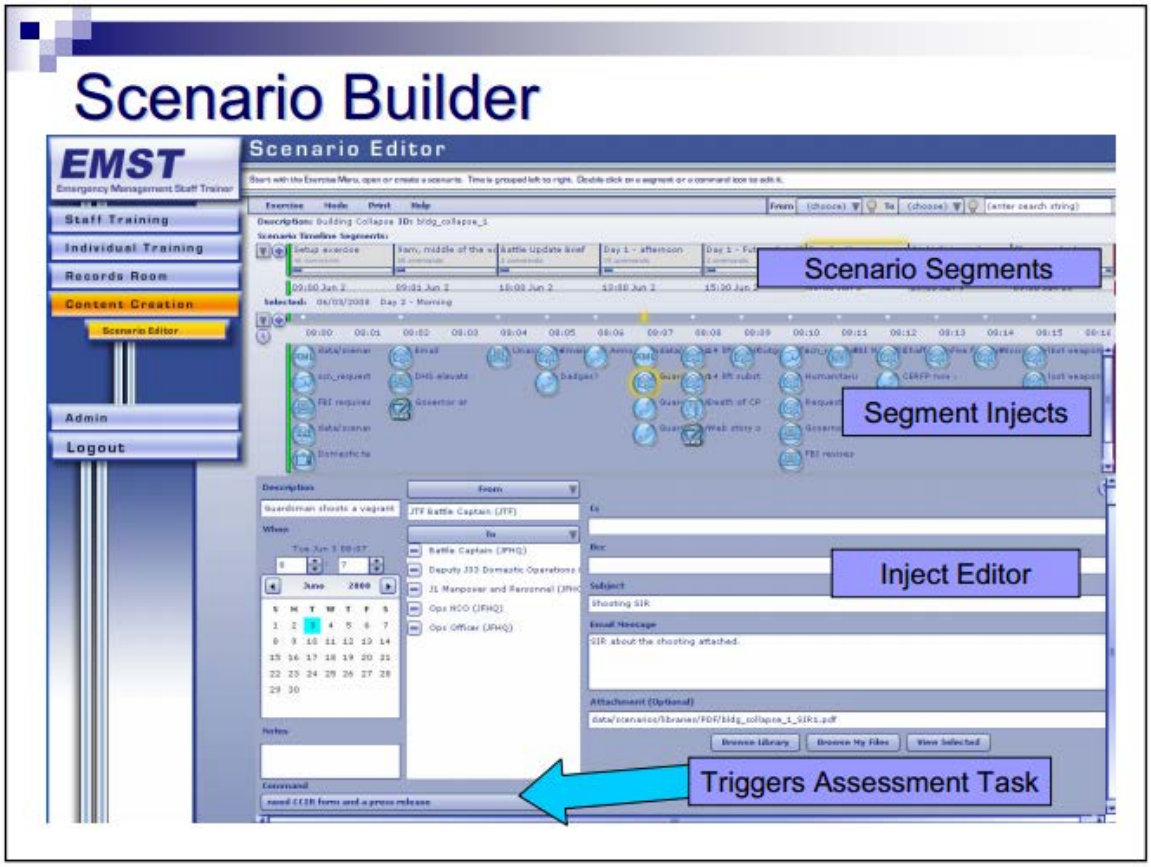

Figure 15: EMST Scenario Builder

VIDEO GAMES

Phases in SEEA are similar to levels in serious games. In the gaming industry, tools such as level editors are powerful means to add new content into an existing game framework. The necessity and use of these tools is similar to those that needed to be designed for the SEEA.

There is a long list of built-in game level editors, varying between different games and different genres of game. Level editors allow for the customization and modification of levels within games. With the help of these tools, players with no programming experience and no knowledge of the game design can easily add new customized gaming contents (e.g. levels and maps) to the game.

War Craft III is a real-time strategy (RTS) video game released by the Blizzard Entertainment. There is a built-in World Editor in this game. The editor is divided into different modules, for example: the Terrain Editor, the Object Editor, the Trigger Editor and the Import Manager. These tools allow for editing similar to the desired functionality of the SEEA Event Editor and Artifact Integrator. The Trigger Editor allows users to create triggers that fire when certain events occur in the game, which has the same idea with SEEA Event Editor. The Import Manager allows the user to import custom models, textures, minimaps, loading-screens and sounds into a map. These

resources can be considered as static artifacts, so the Import Manager is similar to SEEA Artifact Integrator. Figure 16 shows the User Interface of the World Editor of War Craft III.

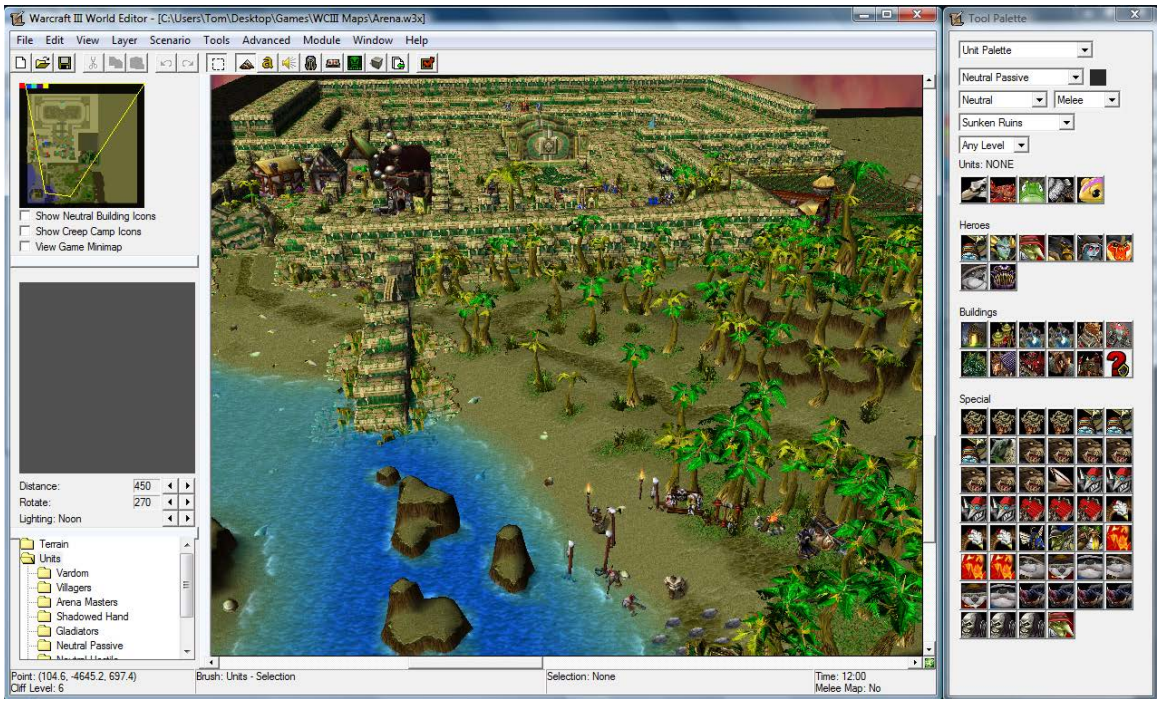

Figure 16: The World Editor of War Craft III

A player called IceFrog built a map using the World Editor in War Craft III. The map is named Defense of the Ancients (DotA) and has attracted millions of players using it for gaming worldwide. Since its original release, DotA has become a feature at several worldwide tournaments, including Blizzard Entertainment's BlizzCon and the Asian World Cyber Games, as well as the Cyberathlete Amateur and CyberEvolution leagues. In a 2008 article of video game industry website Gamasutra, the article's author claimed that "DotA is likely the most popular and most-discussed free, non-supported game mod in the world" (Walbridge, 2008).

DotA is largely attributed to being the most significant inspiration for the multiplayer online battle arena genre. Valve Corporation acquired the intellectual property rights to DotA to develop and release a stand-alone sequel, Dota 2. This is a successful example of the utility of tools for extending content. This also shows the value of design and developing tools for SEEA.

World of Warcraft (WoW) is a massively multiplayer online role-playing game (MMORPG), which is also released by the Blizzard Entertainment. With almost seven million subscribers as of August 2014 (Futter, 2014), World of Warcraft is currently the world's most-subscribed MMORPG, and holds the Guinness World Record for the most popular MMORPG by subscribers. One of the reasons for such a huge number of players is that there are many fascinating quests in the game. A quest is a task given to a player character that yields a reward when completed (http://www.wowwiki.com/Quest). We can consider a quest as a story, which can have a player participating in it. According to an incomplete statistics, there are about 15,590 different quests (http://www.wowhead.com/quests). It is not publicly known if there is a Quest Editor for making

up quests, but it would be a heavy task to create 15,590 quests in the system without any design tools. We can understand that it is possible to hard-code 10 quests for a game; and it is also workable to have some mechanisms setting up in the system for 100 quests; but it would be better to have some tools if there a 15,590 quests to be managed. The phases and artifacts in SEEA are facing the same situation as quests in Wow. This provides motivation for developing useful tools for SEEA.

EXPERIENCE ACCELERATOR REQUIREMENTS

The following describes the target users and their requirements for the Experience Tools.

TARGET USERS

The target users of the experience tools are educators and developers without significant knowledge of the SEEA design. There is no requirement for them to have programming skills. Users are going to use the tools to customize experience contents for the specific needs of the learners.

PHASE EDITOR

MAIN USER INTERFACE

The following are the requirements for the Phase Editor Main User Interface:

- The system should provide user a graphical user interface with the appearance similar to Microsoft Visio as shown in Figure 17.
- The system should provide user a *Menu* with options such as *File*.
- The system should provide user a file option with functionalities of *New File*, *Save File*, *Open File*, *Import Object*, *Export Object* and *Exit*.
- The system should provide user an *About* option with additional information, such as version number.
- The system should provide user a toolbar with common operations of *New File*, *Save File*, *Open File*, *Import Object*, and *Export Object*.

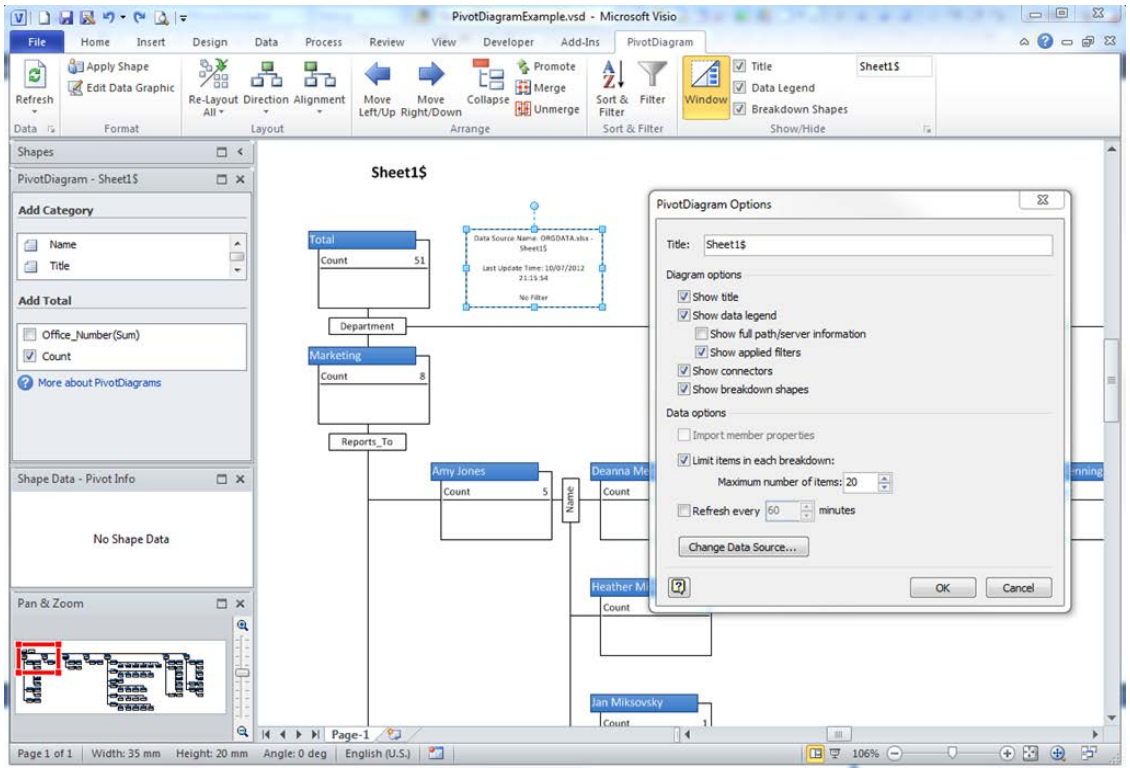

Figure 17: A screenshot of Microsoft Visio 2013

- The system should provide user a panel with a list of named icons (Toolbox), which contain phase related objects such as phase and sub-phase.
- The system should provide user a Canvas on which diagrams or flow charts can be placed.
- The system should provide user a panel with a list of properties (Property Panel) for phase related objects in workspace.
- The system should provide user a button in Toolbox with the name of Import Object, which can be triggered to import existing object, such as existing phase and existing sub-phase.
- The system should provide user an action to drag-drop object from Toolbox to Canvas.
- The system should provide user actions to add, remove, move objects placed on Canvas.
- The system should provide user an action to establish connector between phases on Canvas for phase sequence.
- The system should provide user a hierarchical view for phase and its sub-phases on Canvas.
- The system should provide user operations to edit properties (e.g. name, type, and time) and assets (e.g. contacts) for selected objects (e.g. phase, and sub-phase) on Canvas.

MAJOR FUNCTIONALITY

The following are the major functionalities of the Phase Editor:

- The system should support operations of adding, deleting and modifying phase or sub-phase.
- The system should generate diagram files to save diagrams with information such as connection and position for each item in the diagram.

- The system should parse diagram files to load the diagram to Canvas with properties, connection and position information for each item in the diagram.
- The system should generate a phase sequence file based on the connection relationships of each item on Canvas.
- The system should generate a phase XML file to store phase properties.
- The system should parse an existing phase XML file to load phase properties.

5.4.2.3 OUTPUT FILE

The following are the output files from the Phase Editor:

- A diagram file for storing the diagram on Canvas with connection and position information, as well as property file information of each item in the diagram.
- A phase sequence file for storing phase sequence based on the connection relationship of each item on Canvas.
- A series of phase XML files with the properties.

EVENT EDITOR

MAIN USER INTERFACE

The following are the requirements for the Main User Interface for the Event Editor:

- The system should provide user a *Menu* with options such as *File*.
- The system should provide user a *File* option with functionalities of *New Event*, *Save Events*, *Import Event*, *Export Event* and *Exit*.
- The system should provide user an *About* option with additional information, such as version number.
- The system should provide user a toolbar with common operations of *New Event*, *Save Events*, *Import Event*, *Export Event*, and *Remove Event*.
- The system should provide user a panel with a list of existing events (Event List Panel) of current SEEA.
- The system should provide user buttons for operations such as *New Event*, *Import Event*, and *Remove Event* in Event List Panel.
- The system should provide user a panel (Property Panel) for editing properties of an event.
- The system should provide user a groupbox of components for editing event condition properties in the Property Panel.
- The system should provide user a groupbox of components for editing event action properties in the Property Panel.
- The system should provide user operations to edit properties of event condition and event action in the Property Panel.

MAJOR FUNCTIONALITY

The following are the major functionalities of the Event Editor:

- The system should support operations of adding, deleting and modifying events.
- The system should maintain a file of existing events of current SEEA for adding, deleting and modifying operations.
- The system should generate event XML file to store event properties.
- The system should parse existing event XML file to load event properties.

OUTPUT FILE

The following is the output file from the Event Editor:

- A series of event XML files with event properties.

ARTIFACT INTEGRATOR

MAIN USER INTERFACE

The following are the requirements for the Main User Interface for the Artifact Integrator:

- The system should provide user a *Menu* with options such as *File*.
- The system should provide user a *File* option with functionalities of *New*, *Import Package*, *Export Package* and *Exit*.
- The system should provide user an *About* option with additional information, such as version number.
- The system should provide user a toolbar with common operations of *New*, *Import Package*, *Export Package*.
- The system should provide user a list of buttons for artifact categories such as *Phases*, Emails, *Voicemails* and *PDF Files*.
- The system should provide user a panel to integrate phase/email/voicemail related artifacts and static PDF files.

MAJOR FUNCTIONALITY

The following are the major functionalities of the Artifact Integrator:

- The system should support operation for replacing current phase XML files.
- The system should support operation for creating/modifying available files and folders for each phase/sub-phase/cycle.
- The system should support operation for creating/modifying emails.
- The system should support operation for creating/modifying voicemails.
- The system should support operation for managing static PDF files.

- The system should support operation for converting PDF file to SWF file.
- The system should support operation for compressing the artifact files to a package and exporting the package to file.
- The system should support operation for uncompressing a package of artifact files and importing it to the system.

OUTPUT FILE

The following is the output file from the Artifact Integrator:

- A compressed file with modified phases, events and artifacts.

PHASE EDITOR ARCHITECTURE AND DESIGN

Taking consideration of the system's correlations of each functional component, the system is divided into four modules: Phase Editor, Common, File System and Diagram Designer. Figure 18 shows the modules and their relationships. Such design aims to increase the maintainability of the system as each module has clear readability and portability.

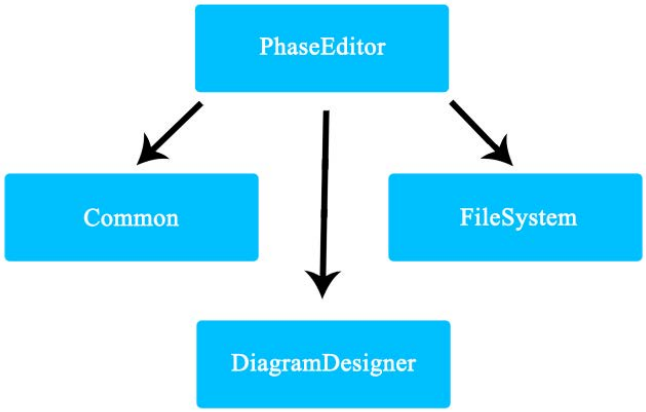

Figure 18: Modules and their Relationships

PHASE EDITOR MODULE

The Phase Editor Module is the entry point into the system. The windows application initializes this. It contains the settings of the application, the user interface, as well as relating view models. The concrete realization of the system is supported in this module. Figure 19 shows the dependency graph of main classes and interfaces of Phase Editor Module.

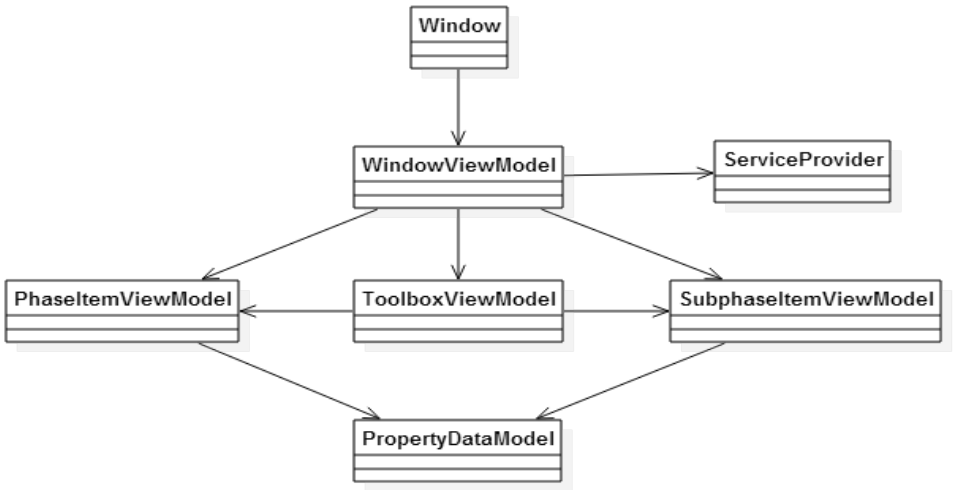

Figure 19: Dependency graph of Phase Editor Module

The Window class has the following major functions:

- Maintaining the application window
- Managing the user interface components
- Communicating with the view model

WindowViewModel class is the view model behind the window. This class has the following major functions:

- Managing diagram items on Canvas
- Communicating with the view model of diagram items
- Contacting with File System Module for file operations

PhaseItemViewModel class is the view model of phase item. SubphaseItemViewModel class is the view model of sub-phase item. ToolboxViewModel class is the view model of Toolbox. The first two view models share the abstract data model, which is the PropertyDataModel class. The ServiceProvider class provides system services such as message box, and dialog box, which will be used by WindowViewModel.

COMMON MODULE

Common Module contains common classes and interfaces for the whole system. It provides interfaces such as PhaseItem, SubphaseItem and Connection for implementing the items that will be put on the Canvas. Also this module includes global static data structures, such as indices for different items.

FILE SYSTEM MODULE

File System Module contains classes for file operations such as input and output. The main functionalities of this module are generating files through the diagrams created in the editor and parsing existed files in order to turn them into diagrams for the editor. It also reads and writes XML files for phase and sub-phase properties.

DIAGRAM DESIGNER MODULE

The Diagram Designer Module is provided by the MVVM Diagram Designer, which contains the core classes that are needed to create a diagram in WPF. It implements most of the functions for Canvas. Figure 20 shows the dependency graph of the main classes and interfaces of DiagramDesigner.

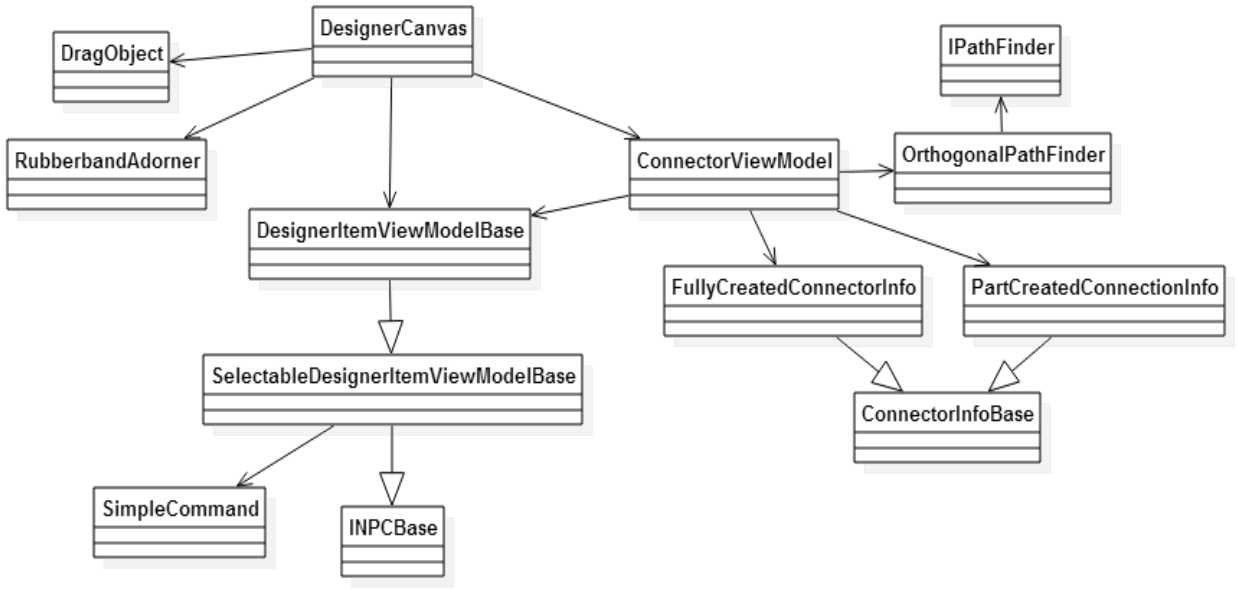

Figure 20: Dependency graph of DiagramDesigner

The DesignerCanvas class is the implementation of Canvas. Actions drag-drop and rubber-band selection are implemented in DragObject and RubberbandAdorner classes. Interfaces such as DesignerItemViewModelBase, SelectableDesignerItemViewModelBase and INPCBase are the abstract bases of most items relating to Canvas. ConnectorViewModel is the view model of connections between items on Canvas. The connection line can find available paths between two connected items automatically with the help of OrthogonalPathFinder class. Each connection has two connected items. An item that the connection starts from is called a Source Item. The class FullyCreatedConnectorInfo keeps information for Source Item. An item that the connection ends with is called a Sink Item. The class PartCreatedConnectionInfo keeps information for Sink Item. Both FullyCreatedConnectorInfo and PartCreatedConnectionInfo classes come from ConnectorInfoBase, which is the connector abstract basis. The major functionality of this module was discussed in the last Chapter.

EVENT EDITOR ARCHITECTURE AND DESIGN

The architectural pattern of Event Editor also follows the MVVM architecture pattern. The architecture is similar to Phase Editor Module of Phase Editor. Figure 21 shows the dependency graph of the main classes for Event Editor.

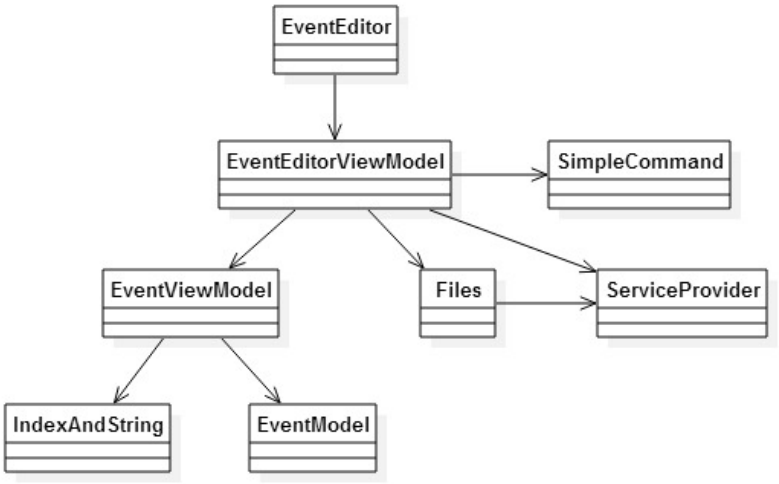

Figure 21: Dependency graph of Event Editor

EventEditor class is the entry point into the system. The windows application initializes this. It contains the settings of the application, the user interface, as well as relating view models. The concrete realization of the system is supported in this module. It has the following major functions:

- Maintaining the application window
- Managing the user interface components
- Communicating with the view model

EventEditorViewModel class is the view model behind the window. This class has the following major functions:

- Managing event list items
- Communicating with the view model of event
- Executing commands
- Contacting with File System Module for file operations

SimpleCommand class encapsulates the general command methods. Command is a feature of WPF. The user interface components such as Button can bind with the command. The implementation of the command execution happens in EventEditorViewModel class. EventViewModel is the view model of event. The abstract data model for it is EventModel class. With the help of IndexAndString class, EventModel can be converted to user interface component compatible data model in EventViewModel class for representing. Files class contains file operations such as read and write event XML files. It also has functions for loading default settings from files for the system.

ARTIFACT INTEGRATOR ARCHITECTURE AND DESIGN

The major use of Artifact Integrator is to create and modify artifacts, which will be integrated into SEEA system. The data model for Artifact Integrator is simple. Most of the functionalities are

relating file operations. So there is no need to continue using the MVVM architecture pattern in this design. Figure 22 shows the dependency graph of the main classes for Artifact Integrator.

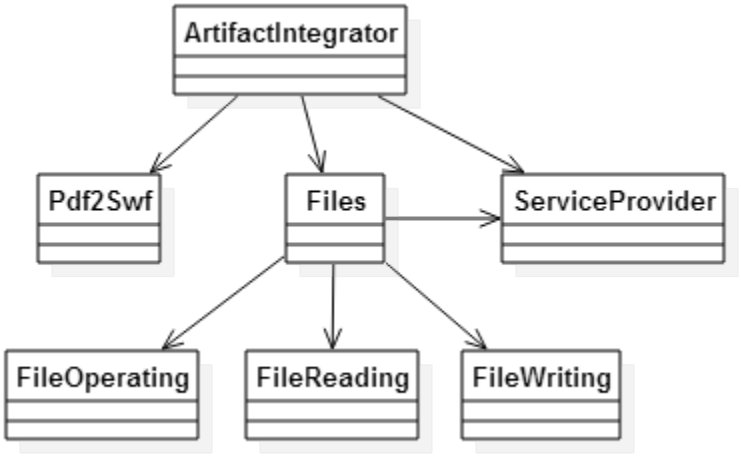

Figure 22: Dependency graph of Artifact Integrator

Similar to EventEditor class in Event Editor, ArtifactIntegrator class is the entrance to the Artifact Integrator. It has the following major functions:

- Maintaining the application window
- Managing the user interface components
- Contacting with File System Module for file operations

Pdf2Swf class is used to convert PDF file into SWF file, which is compatible with Flash. This class uses a process to execute an external application (pdf2swf.exe) for the conversion. And similar to previous design, the ServiceProvider class provides system services such as message box, and dialog box. Files class takes control of file operations. It contains the following sub-modules and functions:

- FileOperate:
    - Getting file/folder information
    - Copying file/folder
    - Renaming file
    - Deleting file
- FileRead:
    - Loading and parsing XML file
    - Loading TXT file
- FileWrite:
    - Generating and exporting XML file
    - Exporting TXT file

The following describes the design and development of the experience tools.

DEVELOPMENT ENVIRONMENT

The following are the components of the experience tool development environment:

- Target Platform: Microsoft Windows platform of version 6.0 and up
- Framework: Microsoft .NET Framework 4.0
- Programming Language: C#
- Integrated Development Environment: Microsoft Visual Studio 2013 Professional
- Development Model: Incremental Build Model
- Dependent Project: MVVM Diagram Designer

PHASE EDITOR DEVELOPMENT

In this development, the Phase Editor is broken down into a number of components. One of the components is implemented in each increment. Each increment contains the following steps: determining task, developing, testing and publishing. In an increment, we set up a task list first, which contains the features that are going to be implemented and needful changes that were described by the users in the last iteration. Then, we developed the features and make the requested changes. The features were then tested after they have been developed. When the tests are passed, we build a new version of the Phase Editor, deliver it to the users and get feedback. The feedback is used in the next iteration.

After the six increments described above, we implemented all of the requirements of the Phase Editor. Figure 23 shows a screenshot of the initial release of the Phase Editor. We shall continue to refine and improve it with the feedback from the users in the future iterations.

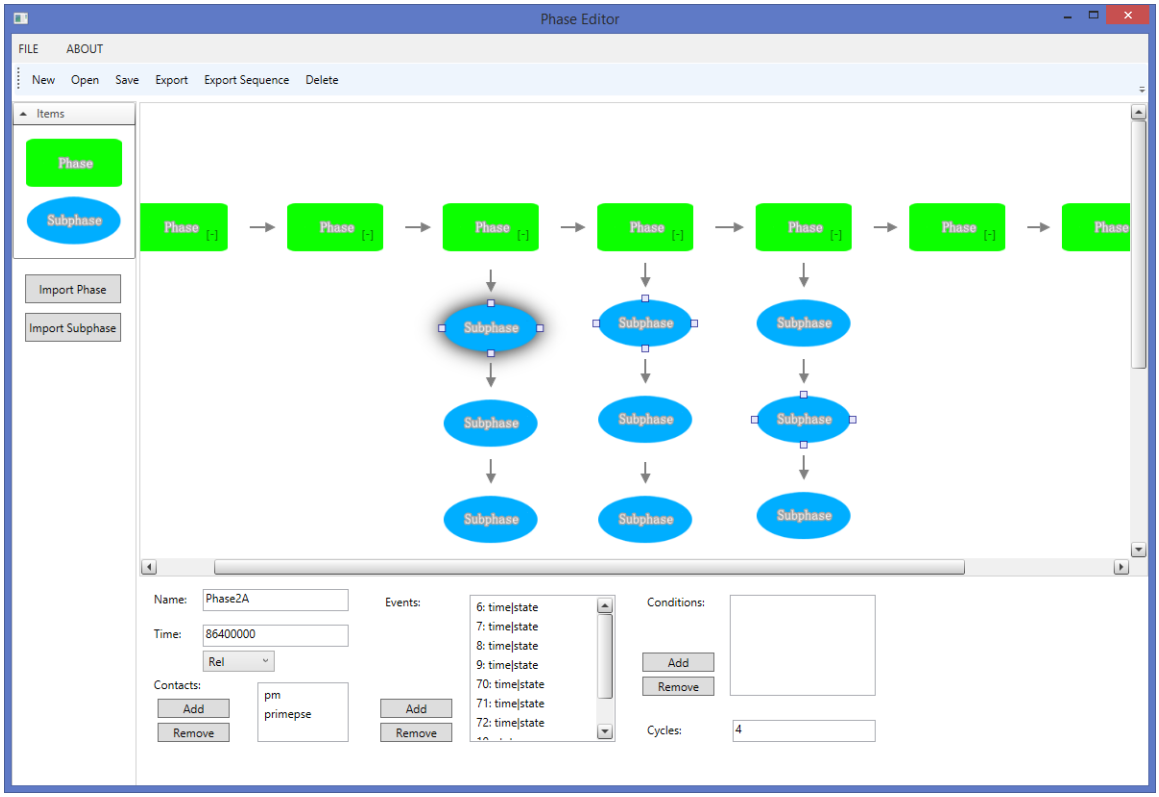

Figure 23: A screenshot of the initial release of the Phase Editor

EVENT EDITOR DEVELOPMENT

Similar to the development procedure of Phase Editor, the Event Editor is also decomposed to components. A few increments were used to implement the Event Editor. Each increment contains the same steps as the increments of Phase Editor, which are determining task, developing, testing and publishing.

After completing the three increments, we implemented all the requirements of the Event Editor. Figure 24 shows a screenshot of the initial release of Event Editor. We shall keep improving it based on the feedback from the users in the future iterations.

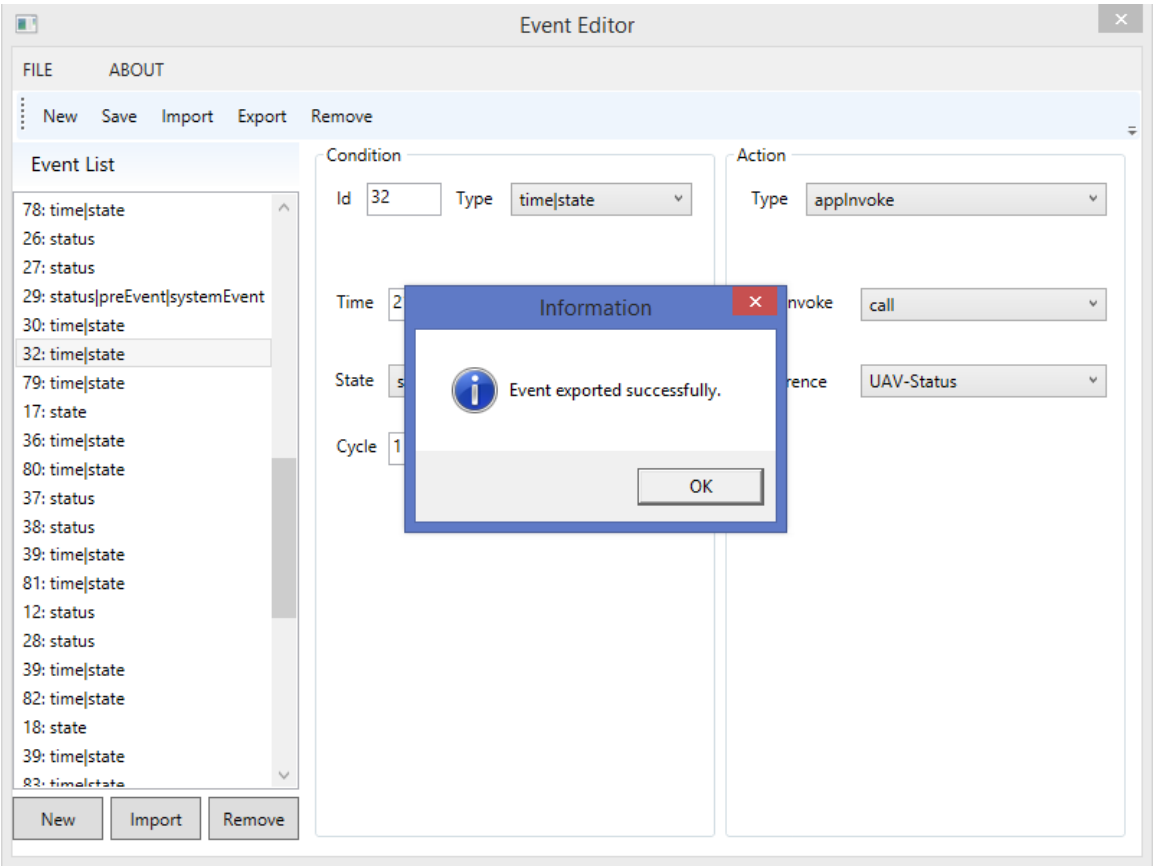

Figure 24: A screenshot of the initial release of Event Editor

ARTIFACT INTEGRATOR DEVELOPMENT

Similar to the procedure of the previous developments the Artifact Integrator is also decomposed to components. Five increments were used to implement the Artifact Integrator. Each increment contains the same steps as in the increments of Phase/Event Editor, which are determining task, developing, testing and publishing.

After completing five increments, we implemented all the requirements of the Artifact Integrator. Figure 25 shows a screenshot of the initial release of Artifact Integrator. We shall keep improving it based on the feedback from the users in the future iterations.

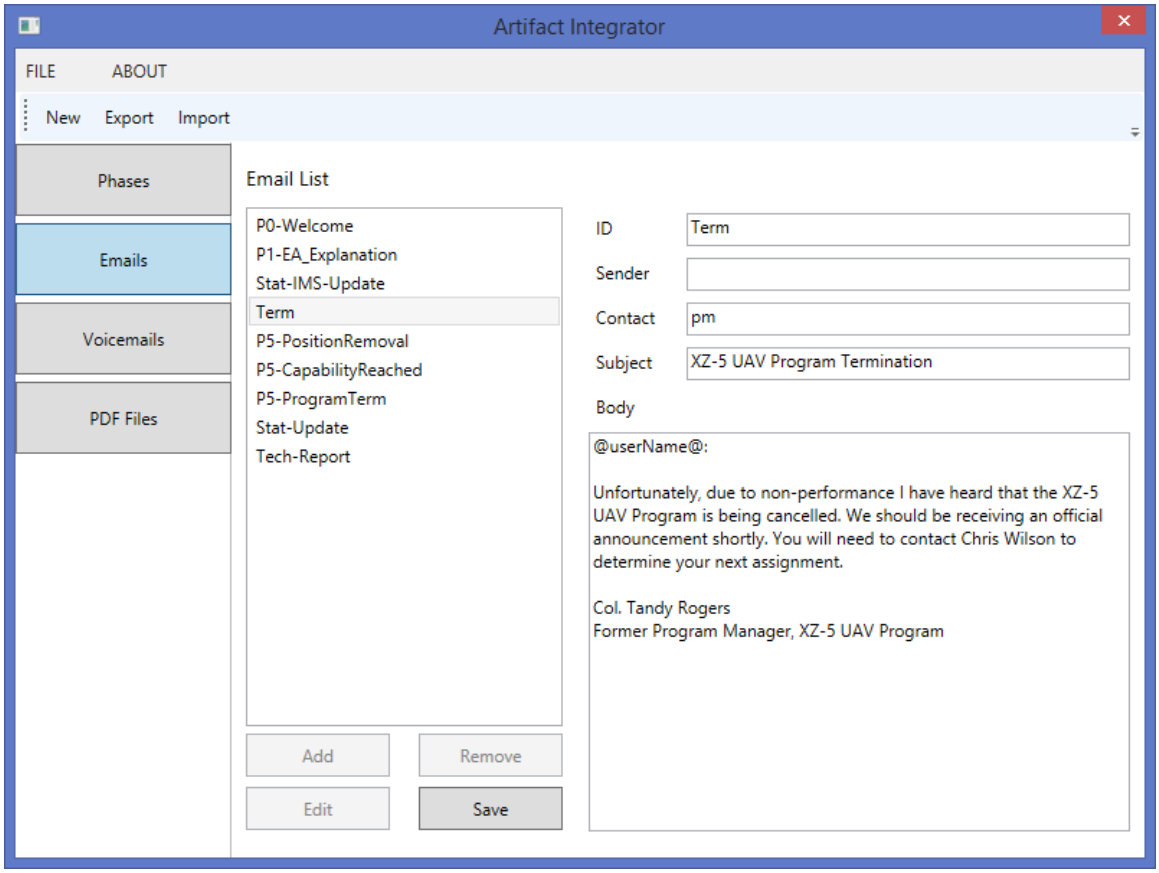

Figure 25: A screenshot of the initial release of Artifact Integrator

TESTING AND EVALUATION

Testing was carried out in every increment. We consider those tests to be unit testing. After all the iterations were completed, integration testing was done for all the functionalities, which are the ones implemented in each increment. Black-box testing, a method of software testing that examines the functionality of an application without peering into its internal structures or workings, was used for this.

Before the initial release version is published, system testing was completed. In system testing, the following tests have been executed:

- UI testing
- Compatibility testing
- Integration testing

Test such as extreme values testing were made during the UI testing. In compatibility testing, the system was run on different platform such as Vista, Win 7 and Win 8. The integration testing in system testing is different from the previous integration testing. In this test, new phase files created with the Phase Editor, together with new events created with the Event Editor, were tested on the SEEA.

LEARNING ASSESSMENT TOOL DESIGN

The accurate assessment of learning is a critical element of the Experience Accelerator's capabilities. To date, much of the focus has been in developing an experience that has the potential to facilitate effective learning. The evolution of this experience has been directed by feedback from subject domain experts, informal and formal evaluations as described below. However, the Experience Accelerator (EA) has now reached the point of maturity at which more precise tools must be developed to measure learning. This section describes the work completed to date, and the current concept for learning assessment tools development.

PREVIOUS EXPERIENCE ACCELERATOR ASSESSMENTS

The following sections describe the informal and formal learning evaluation work performed on the Experience Accelerator.

INFORMAL ASSESSMENTS

An initial formative assessment of the EA was a survey designed for distribution at the NDIA Annual Systems Engineering Conference 2012 conference in which the EA would be demonstrated. While audience members would not be utilizing the EA themselves, they would observe as an EA team member walked through the EA scenario, working through the experience while explaining the goals, background of the EA. This evaluation focused on three areas: audience members' perceptions of the usefulness of the EA, their perceptions of the importance of different components, and their perceptions of how well the EA supported various experience features. The survey utilized a seven point Likert scale, ranging from "strongly agree" to "strongly disagree", to gather audience perceptions of the EA.

The results of the NDIA conference demonstration provided feedback from eleven audience members. The first section asked if the EA appeared as though it would be useful to their peers, students, or employees, and also asked if they liked the look and feel of the interface. Responses to the Likert questions demonstrated a strong belief that the EA would be useful and that the look and feel was appealing. Written comments further supported that the EA was viewed as an important project for the field. Given that the EA was originally envisioned as a three dimensional environment but was ultimately developed as a two dimensional simulated desktop environment (where the learner views emails, documents, and computer-to-computer online voice conversations), it was encouraging that the surveyed responses did not indicate that the simple visual look was a concern.

The second section of questions asked audience members to indicate the importance of components of the EA system to the project's success based on the description of the project's goals. Components listed included supported competencies, the EA's architecture, the design of the experience, tools for designing experiences, supported systems dynamics, and assessment and evaluation. Audience members rated the importance of each component on a seven point Likert scale. All components of the project were rated as very important, so the audience members indicated the scope of the project was on target.

The third section asked audience members to rate their perceptions of how well the EA currently supported the targeted features, which included competencies, architecture, experience design, design tools, systems dynamics, and assessment and evaluation. Ultimately, feedback showed that, audience members, without having hands-on experience with the EA, perceived that the targeted features were largely supported. We also were able to identify future participants interested in evaluating or contributing to the EA project.

FORMAL ASSESSMENTS

In order to utilize learners who could provide an accurate evaluation of the EA as implemented in its intended context, Defense Acquisition University students enrolled in SYS 302 were identified as ideal candidates for the formal evaluation. The EA will be utilized in SYS 302, replacing one case-study exercise, so the use of these students to evaluate the EA provides a perfect match between the test subjects and the target context for the EA.

Test subjects utilized in the EA were split into five teams, and completed only two full phases of the experience with the opportunity to quickly go through the remaining phases. Each team was comprised of students playing assigned team roles. A team of six was comprised of a Lead Systems Engineer (LSE), a learner for each of subsystem of the system being developed in the experience (four of these) and an EVM expert. The original plan specified that the LSE would submit the official recommendations to the EA for the team, and each team would deliver a presentation to the rest of the class describing the team's choices. Subjects were then to replay the full experience individually on their own time and complete a questionnaire designed to capture their thinking patterns. However, in order to fit within the course schedule, a number of elements of the evaluation plan had to be changed or eliminated.

The primary changes were due to time constraints; the students did not complete their learning questionnaires and did not repeat the experiences to measure their changed behaviors. However, valuable data was still obtained in the evaluation with regard to capturing the students' thinking patterns while completing the first two phases of the EA as well as the student and instructor's perspective on the EA and its efficacy. The students were observed during the two days of class that the EA was implemented, and their team presentations were also gathered. These presentations included their reflections on their lessons learned. A number of the targeted lessons were clearly represented in the team presentations, indicating the effectiveness of the EA in promoting its targeted learning outcomes.

For example, for problem solving and recovery learning objective, the importance of *small* schedule delays and the use of additional staff to remediate the schedule problem was touched on by several teams. Teams 1, 3, and 5 mentioned the need to hire staff early in their lessons learned, while Team 2 called for more dramatic staff shifts. Team 4 highlights the results of slipping the schedule too much, lamenting that they were fired for slipping CDR.

Another example can be seen with "Cutting corners to make short term goals while ignoring long term outcomes" which stresses the need to make decisions early. This was touched on repeatedly by the teams in their lessons learned. Team 1 mentions hiring more staff early, Team 3 calls for shifting staff earlier, Team 4 notes how their early emphasis on software worked, and Team 5 reflects on the need to ramp up staff more quickly.

These examples demonstrate that for these two targeted learning outcomes in particular, nearly all of the teams learned the outcomes; the teams very clearly highlighted the outcomes as lessons learned in their presentations. Other learning objectives were also highlighted by the different teams. These lessons were learned despite the fact that the learners only fully completed the first two phases of the experience before speeding through the remaining phases in order to see their results. Furthermore, the EA was designed to be played multiple times by learners, so these results are indicative of impressive learning gains given the limited implementation of the experience.

Students also were asked to provide their perceptions of the EA – what it did well and what could be improved. The class discussed these and the comments were captured. Comments highlighted a number of key features of the EA as positives. These included the case-based format of the EA, noting its representation of real-life issues and modeling of real work interactions. Students also noted the importance of immediate feedback on the decisions and the interactive nature of the simulation – accelerating learning by simulating a project lifecycle in a short amount of time. The challenging aspect of the simulation was also highlighted as it was noted that being fired kept the learning challenging a more interesting, a possible reference to the EA's targeted "scar tissue" – emotional connection in order to promote learner transfer of learned objectives. One key positive from the feedback was that the user interface received a great deal of praise.

Recommendations provided additional insights on how we can further improve the EA. For example, a greater focus on performance and technical aspects as opposed to cost and schedule was recommended. A few small bugs were also identified which will be corrected, and information which will be included in future iterations of the instructors' manual will help to further improve the efficacy of the EA.

Ultimately, the formal evaluation of the EA was hindered by the inability to compare novice learner actions and thinking to those of experts. However, while this comparison could not be made in this implementation, it certainly could be done in a future implementation. Furthermore, clear evidence of learning was gathered from the data that was gathered and learner perspectives on the efficacy of the EA were largely positive while still providing some helpful suggestions for improvement. The formal evaluation can therefore be seen as a success and indicative of the EA's efficacy in meeting its targeted learning objectives.

CURRENT WORK

In order to evaluate the efficacy of the EA, literature on assessing systems thinking competencies was reviewed from a variety of fields. Example assessments were sought that would be well suited to a pre – post evaluation, measuring learners' systems thinking competency prior to introducing the EA and following their uses of the EA. Unfortunately, nothing was found in the literature that was well suited to a quality evaluation of the EA. Therefore, a new experimental design grounded in the literature will need to be devised, along with a set of tools to facilitate its application.

While the Experience Accelerator (EA) has a broader goal of accelerating the learning of critical SE competencies through an experience-based system, systems thinking skills are a key component of the targeted learning outcomes. Systems thinking is at the core of the targeted EA

SE competencies and therefore one of the primary competencies to be assessed in order to evaluate the effectiveness of the EA.

Systems thinking seeks to improve decision making and complex problem solving through deep systemic understanding. Typically, in order to assess learning gains in these areas, three approaches are utilized: measuring performance resulting from decisions (such as a game or simulation score), reviewing decisions and actions that were taken, or measuring learner understanding (the rules and mental operations that lead to decision making) (Rouwette, Größler, & Vennex, 2004; Karakul & Qudrat-Ullah, 2008). Measuring understanding seeks to verify that improved decision making arises from understanding the system and not simply from trial-and-error (Kopainsky, Pirnay-Dummer, & Alessi, 2012).

All of these approaches are valid and can result in worthwhile evaluation. As systems thinking skills are applied in order to understand and solve complex problems, educational research on the assessment of problem-solving skills can be helpful in designing an effective evaluation.

In order to solve an ill structured problem, students must be able to deconstruct the problem into its constituent parts (e.g., stakeholders, relationships among them, impacts of the problem on them), define the problem in their own words, determine resources to help them understand the problem, determine and pursue learning issues, and develop and test a solution. Research on the evaluation of problem-solving skills tells us that in order to evaluate problem-solving ability, we must assess students' ability to do each of these steps. The EA seeks to accelerate the learning of novice SE's and advance them more quickly to expert SE performance. Experts use heuristics to skip steps; novices typically are not capable of doing this.

A meta-analysis of problem-solving assessment literature found that 18 of 23 studies deemed of high quality used cases or simulations as assessment methods (Belland, French, & Ertmer, 2009). With the EA Simulation, we have the means to measure learner's performance within the experience. Learners make decisions within the EA, the simulation determines the results of those decisions, and we are provided with outcomes that we can utilize in order to assess the effectiveness of learners' decisions.

In order to assess learners' levels of understanding and to determine if the EA improves learning, a more thorough picture of the thinking behind learners' choices is needed. Therefore, to assess learners' understanding, we should also elicit their views of the system, the problems they faced, and the thinking behind their decisions made to solve these problems. Emerging literature in systems dynamics increasingly has instead been seeking to assess learners' understanding or mental models.

Therefore, in order to assess learner performance, we can capture it (through EA results), and we can analyze the actions and decisions taken by the learner. In order to assess learner understanding, we can capture learner approaches to decision-making (through verbal protocols) and use expert choices and protocols as a baseline for "good" decision making.

Our evaluation plan therefore focused on:

1. Benchmarking with an objective "score" which is also useful in motivating students
2. Comparing SME EA actions and results to novice SE actions and results.

3. Comparing SME written (or transcribed verbal) descriptions of their decision-making process during the EA to novice SE written (or transcribed verbal) descriptions of their decision-making process during the EA in experience 1 and experience 2.
4. Tracking learning with changes in 1-3 above through a learner's multiple iterations through the experience.

To support this plan, the EA will be instrumented to record information as a learning laboratory. Research will be done to determine the requisite data that needs to be recorded and the EA will be updated accordingly. Prior to completing this research, the following data has been selected and will be collected from the EA:

- **Participant Identification:**
  - Learner's Name & demographic information
  - Team Name & other members
  - Instruction Name & Roles played in Experience
- **Experience Session information:**
  - Experience Name and Version
  - Date of Experience Start and End
  - login dates and duration of each session
  - Phases/cycles covered in each login session
  - Elapsed time & number of session per Phase/Cycle
  - Links to past experience information
- **Learner Experience Inputs & Actions:**
  - Self-Assessment
  - Initial Recommendation Input
  - All subsequent Recommendation Inputs
  - Workflow sequence with each action recorded with a timestamp
  - Who is called and which questions are asked, in which order
- **Instructor Input**
  - Feedback provided to Learners (dialog, email, etc.)
  - Recommendations accepted/rejected
  - Instructor's observations
- **Simulation Output:**
  - Last phase/cycle completed
  - Results of schedule, cost, range and quality
  - Final Status Charts
  - Final score
- **Reflection**
  - Reflection feedback provided to the Learner
  - Learner's reflection input

Next, a set of analysis tools will be developed to make sense of this information. Test cases will be created to provide benchmarks to baseline this analysis. Finally, a demonstrable set of learning experiences will be recorded and analyzed to provide feedback on the capabilities of the system.

POTENTIAL DIRECTIONS

The goal of the EA is to accelerate the growth of expertise in novice users through a case-based simulation. When learning is viewed as acquiring expertise, it is appropriate to evaluate learning by comparing how learners are thinking and performing compared with experts. When dealing with the complexities of system dynamics, causal loop diagramming represent a strong choice as a concept mapping methodology for use in comparing the understanding of novices and experts (Spector, Christensen, Sioutine, & McCormack, 2001). Concept maps are intended to explore meaningful learning and "tap into a learner's cognitive structure and to externalize, for both the learner and the teacher to see, what the learner already knows" (Novak & Gowin, p. 40). The use of computer technology to standardize concept map approaches allows for automatized and objective assessment of learner concept maps (Weinerth, Koenig, Brunner, & Martin, 2014).

Causal loops visually represent the interrelationships of a dynamic system; the hypothesis follows: given the same problem case, systems engineering experts will construct similar diagrams; novices, however, will construct different diagrams. Therefore, the more similar a learner's diagram is to that of experts, the higher the learner's expertise. Annotation of expert diagrams can further illuminate the underlying thinking and support better comparison between novice and expert (Spector, 2006).

Spector (2006), proposes his Dynamic Evaluation of Enhanced Problem-solving (DEEP) methodology, which applies these approaches in ten steps:

1. Identifying characteristic complex problems.

2. Eliciting expert conceptualizations of the problem space.

3. Representing expert patterns in both textual and graphical formats.

4. Determining salient features of these representations.

5. Establishing measures of similarity in salient features.

6. Eliciting novice problem conceptualizations.

7. Representing novice patterns in the same formats used for expert responses.

8. Identifying the presence or absence of salient features as a first rough measure of higher order problem-solving ability in learners.

9. Establishing measures of distance from expert patterns for each salient feature as a more

refined measure of higher-order problem-solving ability.

10. Analyzing changes in learner responses over time and through experience; initial measures can provide a baseline; subsequent measures can provide evidence of learning; post-instruction measures can provide an indication of overall progress (p. 112).

In the case of the SEEA, the developed cases represent the complex problems, and the use of experts to develop annotated causal loops will support the automatized comparison by the system to novice created causal loops. Therefore, the assessment tool would need to support both the creation of causal loop diagrams as well as incorporate data from expert-created diagrams in order to automate the comparison and evaluation of learner performance with the simulation. Given the complexities of this approach, the tool could minimally support the storing and manual evaluation/comparison of the concept maps, as well as other comparative performance measures between experts and novices, such as simulation choices and scoring. As described in the DEEP methodology, changes in learner performance within the SEEA over time can provide evidence of learning and continued growth.

While few automated assessment tools have been successfully developed, one example is the Highly Integrated Model Assessment Technology and Tools (HIMATT) toolset, which has a record of research validating its use (Pirnay-Dummer, Ifenthaler, & Spector, 2010). Furthermore, there is some evidence that the act of creating concept maps themselves can support the learning of complex systems (Stewart, 2012), so the very act of providing data to assess learning through the assessment tool could prove beneficial.

Learner construction of various forms of knowledge representation will be explored in future research.

CONCLUSIONS AND FUTURE RESEARCH

This section summarizes the results of Part 1 and describes the future work in Part 1 of the EA Tools Development program.

CONCLUSIONS

This report has presented results from the first part of a three-part project to create tools to support design, development and measurement of learning of experiences in the Experience Accelerator as shown in Table 5. Each Part builds on the successful completion of its previous part. The tools development efforts fall into four major categories – simulation tools for building and testing simulation models that mimic the behavior and results of acquisition programs that focus on system design and development, experience building tools that provide the structure for such system engineering experiences and the events that occur in them, learning assessment tools to measure the efficacy of the experience and EA infrastructure changes to support this work.

Table 5. Tool development in three parts

Time	Simulation Tools	Experience Builder Tools	Learning Analysis Tools	EA Infrastructure
Part 1	Develop prototype Sim Builder and Tuner tools	Develop prototype Phase and Event Editor, and/or Artifact Integrator tools	Research and create high level design of Learning Assessor tool	Update EA infrastructure to support tools
Part 2	Extended functionality of Sim Builder and Tuner tools, make available to broader community. Determine new functionality as needed	Refine Phase and Event Editor, and/or Artifact Integrator tools, make available to broader community. Open source. Determine new functionality as needed	Develop prototype Learning Assessor prototype	Update EA infrastructure to support tools
Part 3	Extend functionality of Sim Builder and Tuner tools, make available open source. Determine new functionality as needed. Support existing tools, Specify new tools	Extend functionality of tools, make available open source. Support existing tools, Specify new related tools	Develop Learning Assessor tool, make available to for external evaluation. Specify future functionality	Update EA infrastructure to support tools

An iterative, agile approach was used for the Part 1 work as described earlier in this report. Each of the tools required different amounts of time and effort for each of these phases depending upon their maturity. For example, the Experience Building Tools required the shortest amount of time before they can be specified, as there have already been some preliminary investigations for their use. The Learning Analysis tools required substantially more research in their specification such that their implementation was not expected to take place during this first phase of the EA tools program. The overall results, as anticipated, are prototypes that require

various levels of additional development and engineering for deployment. The status of each of these developments is described below.

The simulation tool development project achieved its Part 1 objectives with the completion of the following:

- Requirements and use cases for simulation tools
- Review of existing tools to leverage
- Prototype Sim Builder with GUI for model building and capability to manipulate sub-models for purposes of modularity and model archiving/curation
- Prototype Sim Tuner that allows testing of model behavior drilled down to variables of interest

The Experience building tools development achieved success in its Part 1 objectives, including the development of the Artifact Integrator, which was an optional item, with the following items:

- Requirements and use cases for Phase Editor, Event Editor and Artifact Integrator
- Review of existing tools to leverage
- Prototype Phase Editor with GUI for phase property and sequence editing
- Prototype Event Editor with GUI for event property editing
- Prototype Artifact Integrator for integrating files created with Phase Editor and Event Editor into the Experience Accelerator

Progress was achieved in the learning tools area as noted below:

- Literature review
- Initial concept exploration
- Captured data identification

However, the infrastructure development work necessary to support the Experience building tools diverted resources from this effort.

The following work was completed in the infrastructure development:

- Review of current infrastructure design
- Server-side implementation updated to support Phase Editor
- Review and evaluate client-side implementation for Artifact Integrator support

The risk identification and mitigation strategy noted below was largely effective in addressing the program risks.

- **Risk**: Ensuring that there is graduate student staff with the knowledge and capabilities required for effective, efficient tool development despite gaps in funding.
- **Mitigation strategy**: Keep current graduate students engaged as much as possible with stopgap funding from other sources. Determine other sources of students or software research developers.

- **Risk**: Rapidly determining the success metrics and requirements for Experience Design and Development Tools, given the diverse set of users and user needs.
- **Mitigation strategy**: Engage a representative set of users early and work to ensure proper alignment.

- **Risk**: Providing design and development tools that meet the needs of a set of educators and developers with potentially diverse application areas of interest.
- **Mitigation strategy**: Examine application areas and needs of user base early and identify generic tool features to support multiple applications/needs. Eliminate "outlier" application areas of necessary.

- **Risk**: The design and development tools in some case span different modules of the EA technology breakdown structure, thus necessitating multi-perspective development.
- **Mitigation strategy**: Identify dependencies and engage team members to integrate tools across modules as appropriate.

- **Risk**: The likelihood of feature creep that prevents the completion of an adequate set of tools for to provide a complete environment for Experience creation.
- **Mitigation strategy**: Utilize agile develop practices to ensure that the highest value features are being developed at all times.

A successful demonstration of the tools has been given to the sponsor. The tools are now at the stage where they are ready to be evaluated by external users for their use in Experience and Simulation development. The iterative approach was quite successful at providing incremental functionality that was reviewed with its potential users throughout the development effort, prioritizing the most important features and delivering working prototypes throughout the effort. This approach will be continued in Part 2. It is also noted that updating the EA infrastructure to remove its hard coded limitations required more work than was initially anticipated.

FUTURE RESEARCH

Part 2 of the EA Tools Development program will build upon the success of Part 1. This phase will provide the ability for non-EA researchers to begin to use the tools and provide feedback that will be used to prioritize the development. The following is a brief description of the future work in each of the four major tools areas.

Future research for the simulation tools includes the following items:

- Sim Builder
    - Specify initial set of sub-models for model archive and populate;
    - Review upstream systems engineering functions not currently modeled in the SEEA for inclusion in model archive;
- Sim Tuner
    - Provide support for sub-model tuning;
    - Provide support for understanding variable dependencies;
- Chart Designer module
    - Specify use cases and requirements;
    - Develop module;
- General
    - Identify potential dependencies between simulation tools and other tools; and
    - Engage user community for evaluation and feedback, refine tools as needed.

Future work for the Experience development tools includes the following items:

- Enhance Phase Editor based on user's feedback
- Enhance Event Editor based on user's feedback
- Enhance the Artifact Integrator based on user's feedback
- Research and develop integrated simulation, experience and learning assessment tools for an entire EA experience

Future research for the Learning Assessment tools includes the following items:

- Complete research in learning assessment techniques and tools
- Create high level design of tool specification
- Develop preliminary prototype

Additional development work is necessary in the EA infrastructure. The current EA technology is hard-coded to a large extent; some required changes for both the server-side and the client-side were discovered for future work. Investigations were made on how to build tools for defining experience aspects other than phase and events. Server side code contains many well-defined and hard coded state variables and simulation input/outputs. These variables need to be extracted to allow further changes on the experience. The pages designed using Adobe Flash are important to the whole experience. Since the current flash client was designed for UAV experience from bottom up, there are numerous pages that need to be altered in order to enable other types of experiences.

Table 6 shows the Part 2 work plan for each of the three tool sets and infrastructure. This has been updated from the original RT-123 proposal based on results and findings of Part 1. As in Part 1, an iterative, agile approach will be used in the development, so that the results will be time-boxed to complete with the Part 2 timeframe. However, the specific features and functions have not been determined for these deliverables, as they will be determined in the "specify" phase outlined below. The completion of Part 2 will expand the research conducted in Part 1, while continuing to address the research objectives described in Section 3.2.

Table 6. Tool Development: Part 2

Phase	Simulation Tools	Experience Builder Tools	Learning Analysis Tools	EA Infrastructure
Specify	25%: Specify additional functionality for Sim Builder and Sim Tuner based on user feedback; Specify initial set of sub-models for model archive; Review upstream systems engineering functions not currently modeled in the SEEA for inclusion in model archive; Specify use cases and requirements for chart specification module; Review existing literature/tools for characterizing upstream inputs and downstream effects for simulation variables; Identify potential dependencies between simulation tools and other tools.	25%: Specify additional functionality for Phase and Event Editor, and Artifact Integrator based on user feedback; Specify use cases and requirements for remote upload module for Artifact Integrator; Review existing literature/tools for remote upload; Identify potential dependencies between experience tools and other tools.	25%: Complete research into learning assessment techniques and tools	25% Review needs of infrastructure to support tools. Create specification for longer term infrastructure
Develop	65%: Extend Sim Builder and Sim Tuner with additional functions from "specify" phase; Populate initial model archive with sub-models; Include upstream systems engineering functions in model archive; Extend Sim Builder with chart specification module and GUI; Extend Sim Tuner with decision support tools for users (e.g., upstream inputs and downstream effects for critical variables); Address dependencies with other tools. Make available as open source.	65%: Refine Phase and Event Editor, and Artifact Integrator tools, make available to broader community for review; Extend additional functionality for Phase and Event Editor, and Artifact Integrator based on user feedback; Extend remote upload module for Artifact Integrator; Address dependencies with other tools. Make available as open source.	65%: Create high level design of Learning Assessor tool specification of learning analysis tools and preliminary prototype, as allowed by available time and resources	65% Update infrastructure as necessary for tools.
Evaluate	10%: The developed tool(s) will be evaluated iteratively through its development culminating in this final Part 2 evaluation.	10%: The developed tool(s) will be evaluated iteratively through its development culminating in this final Part 2 evaluation.	10%: Evaluate design and preliminary prototype	10%: Evaluate results through tool evaluations

APPENDIX A: IDENTIFICATION AND PRIORITIZATION OF TOOLS

In developing a new experiential learning module using the current EA, a developer would undertake a variety of tasks with limited tool support. This appendix provides a list of these tasks, which depend on detailed knowledge of the SEEA implementation. It also presents a prioritization based on these tasks for tools that would support the developer in executing these tasks.

Experience Conceptual design:

- Determine the specific competencies, 'aha's' and targeted difficulty levels to be supported.
- Select/design a particular scenario (e.g., a particular DoD program) for the experience that supports the desired learning and develop a storyboard for the desired experience elements.
- Define the program's high-level performance measures, their targets that constitute success and deviations from target that cause failures.
- Define phases and cycles within phases.
- Define challenges and landmines consistent with competencies, 'aha's' and difficulty levels.
- Specify important non-player characters/roles, plus any live player roles.
- Specify desired simulated program behavior, status outputs, inter-relations between elements, and learner control points.
- Specify desired artifacts (e.g., program background material, learner decision/recommendation forms, etc.).
- Specify the types of feedback to be provided to the learner based on decisions made during the experience.

Experience Development, testing and enhancement:

- Specify program status variables and targets that operate at a more detailed level than high-level performance measures.
- Implement phase and cycle behavior manually.
- Develop simulation models and datasets to ensure desired simulated program behavior and inter-relations of program elements (via testing/tuning)
- Develop artifacts to be populated with simulation output to provide learner insight into current and previous program status
- Develop non-player characters and state-based dialog whereby the learner can query the NPCs to discover additional information (or be distracted by inconsequential minutia).
- Embed challenges and landmines into simulation models and NPC dialog.
- Develop/write desired artifacts (e.g., program background material, learner decision/recommendation forms, etc.).
- Write scripted feedback to learner based on alternate learner decisions, linked to program outcomes
- Integrate artifacts, simulation models, NPC dialogs, and learner feedback into Experience Accelerator via a manual process involving re-linking or recompilation.
- Test consistency of experience.

These efforts can be categorized in the following areas with the areas with the greatest need and potential for tool development highlighted:

Objectives & Experience Concept Development

- Learner Profile creation
- Competencies and Aha's identification
- Experience story boarding and conceptualization

Context

- Project specification
- Project state and thresholds
- Roles, motivations personality factors and character types
- Review types and result options

Experience Events and Flow

- ***3 Experience Phases & Time specification***
- ***4 Event specification***
- Challenge Control -> simulation parameter setting, event triggering
- Evidence of Challenges and Landmines
- Mitigating Actions & Effects
- Relationships between Competencies/Aha's, Challenges/Landmines, Mitigating Actions & Effects

Reflection & Evaluation

- Feedback format
- Scoring
- ***6 Learning evaluation***

Artifacts

- Background information
- ***7 Project reviews***
- ***8 Learner recommendation forms***
- ***9 Simulation status reports***
- emails
- ***10 Dialog***
- Mentoring
- Evaluation feedback

Simulation

- ***1 Models construction***
- ***2 Parameter setting***

Overall

- ***5 Artifact entry***
- ***11 Process and tools documentation***

Since the Experience Accelerator is an open source application, few existing tools were found suitable to facilitate the automation of developing a new experience. During the course of designing and developing the Systems Engineering Experience Accelerator, several existing tools were discovered in the open-source (or near open-source) world, or were developed for purposes of the project. These include:

- ChatMapper – a dialog specification application that have a sophisticated graphical user interface for writing state-based NPC dialog, with emotive/personality characteristics (e.g., an angry response vs. a calm response). ChatMapper strictly speaking is not open-source, but it does produce open-source outputs in XML. An XML translation tool was developed that has the capability of translating ChatMapper XML files into files that can be executed by the Experience Accelerator dialogue engine (EADE). Prior to this, a tool was developed that converted text dialog into EADE XML.
- SystemDynamics[6] – a graphical user interface application that allows a system dynamics model to be developed visually and written to an XML file. This is an open-source product, but with limited functionality. It has been enhanced substantially during the project.
- JFreeChart [7] – an open-source graphics application that generates charts and other visualization of datasets. This is used to generate program status visualizations from the simulated program. It has been customized to accept an XML specification for the various charts.
- MVVM Diagram Designer[8] – an open-source project that provides capability of drawing diagrams under C# .Net Framework. It has been used to help to implement the canvas for Phase Editor.
- Pdf2Swf[9] – an open-source PDF to SWF Converter. It enables the user to have fully formatted text, including tables, formulas, graphics etc. inside Flash applications. It has been used in Artifact Integrator to convert PDF files into SWF files for static artifacts.
- Templates for project reviews, email status updates and learner decision/recommendation forms – these have been created in text/tabular format and allow population of status data and other data into the template by the Experience Accelerator application.

[6] http://system-dynamics.sourceforge.net/
[7] http://www.jfree.org/jfreechart/
[8] http://www.codeproject.com/Articles/484616/MVVM-Diagram-Designer/
[9] http://www.swftools.org/

These tools and template have proved valuable, but are of limited use to an outside developer community.

Over the course of the Experience Accelerator development, a number of tools and templates were developed to expedite development. Of these high leverage areas, the following are largely complete, but in need of refinement and documentation:

- ***7 Project reviews [templates created]***
- ***8 Learner recommendation forms [templates created]***
- ***9 Simulation status reports [XML syntax for automated generation]***
- ***10 Dialog [External tool, output used by current system]***
- ***11 Process and tools documentation [in process]***

APPENDIX B: REFERENCES

EXPERIENCE ACCELERATOR RESEARCH PUBLICATIONS

1. Bodner, D., Wade, J., Squires, A., Reilly, R., Dominick, P., Kamberov, G., Watson, W. (2012). Simulation-based decision support for systems engineering experience acceleration. In Proc. 2012 IEEE Systems Conference, Vancouver, BC, Canada.
2. Bodner, D., Wade, J., Watson, B., Kamberov, G. (in press). Designing an experiential learning environment for systems engineering and logistics. In Proc. 2013 Conference on Systems Engineering Research (CSER), Atlanta, GA.
3. Squires, A., Wade, J., Dominick, P., Gelosh, D. (2011a). Building a competency taxonomy to guide experience acceleration of lead program systems engineers. In Proc. 2011 Conference on Systems Engineering Research (CSER), Redondo Beach, CA.
4. Squires, A., Wade, J., Watson, B., Bodner, D., Okutsu, M., Ingold, D., Reilly, R., Dominick, P., Gelosh, D. (2011b). Investigating an innovative approach for developing systems engineering curriculum: The Systems Engineering Experience Accelerator. In Proc. 2011 American Society for Engineering Education (ASEE) Annual Conference and Exposition, Vancouver, BC.
5. Squires, A., Wade, J., Watson, W., Bodner, D., Reilly, R., Dominick, P. (2012). Year one of the Systems Engineering Experience Accelerator. In Proc. 2012 Conference on Systems Engineering Research (CSER), Rolla, MO.
6. Wade, J., Kamberov, G., Bodner, D., Squires, A. (2012). The architecture of the Systems Engineering Experience Accelerator. In Proc. International Council on Systems Engineering (INCOSE) 2012 International Symposium/European Conference on Systems Engineering (EUSEC), Rome, Italy.
7. Wade, J. P., "Design and Development Tools for the Systems Engineering Experience Accelerator," Systems Engineering Research Center, June 18, 2014.

JOURNAL ARTICLES AND REPORTS

1. Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. Educational Researcher, 18(1), 32-42.
2. Alex Baker, E. O. N., Andr, et al. (2005). An experimental card game for teaching software engineering processes. 75: 3.
3. Chapman, G. M. and J. F. Martin (1995). Computerized business games in engineering education. 25: 67.
4. Chen, W. F., W. H. Wu, et al. (2008). Work in progress - A game-based learning system for software engineering education: T2A12.
5. Ebner, M. and A. Holzinger (2007). Successful implementation of user-centered game based learning in higher education: An example from civil engineering. 49: 873-890.
6. Jain, A., and Boehm, B. (2006). "SimVBSE: Developing a Game for Value-Based Software Engineering," CSEET 2006.
7. Morsi, R., Jackson, E. (2007). "Playing and Learning? Educational Gaming for Engineering Education". Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE '07. 37th Annual, Milwaukee, WI.
8. Okutsu, M. (2009) Teaching Engineering Design Principles via a Serious Game Format, West Lafayette, Indiana, April 21-22, 2009.
9. Steif, P. S. H. M. (2006). Part II - Contributions in: Engineering Education Research, Engineering Design, Simulation and Games, Pre-college Education, Logistics, Engineering Systems, Sports Engineering -

Comparisons between Performances in a Statics Concept Inventory and Course Examinations. 22: 1070.
10. Taran, G. (2007). Using Games in Software Engineering Education to Teach Risk Management: 211.
11. Walbridge, M. (June 12, 2008). "Analysis: Defense of the Ancients – An Underground Revolution". Gamasutra. Retrieved June 23, 2008
12. Futter, M. (August 5, 2014). "[Correction] World of Warcraft Subscriptions Slide by 800,000". Game Informer. GameStop. Retrieved August 5, 2014.
13. Belland, B. R., French, B. F., & Ertmer, P. A. (2009). Validity and Problem-Based Learning Research: A Review of Instruments Used to Assess Intended Learning Outcomes. *Interdisciplinary Journal of Problem-Based Learning, 3*(1). Available at: http://dx.doi.org/10.7771/1541-5015.1059
14. Karakul M, Qudrat-Ullah H. 2008. How to improve dynamic decision making? Practice and promise.
15. In *Complex Decision Making: Theory and Practice*, Qudrat-Ullah H, Spector JM, Davidsen PI (eds). Springer/NECSI: Berlin; 3–24.
16. Kopainsky, B., Pirnay-Dummer, P., & Alessi, S.M. (2012). Automated assessment of learners' understanding in complex dynamic systems. *System Dynamics Review, 28*(2), 131-156.
17. Novak, J.D., & Gowin, D.B. (1984) *Learning how to learn*. Cambridge University Press, Cambridge.
18. Pirnay-Dummer, P., Ifenthaler, D., & Spector, J. M. (2010). Highly integrated model assessment technology and tools. *Educational Technology Research and Development*, *58*(1), 3-18.
19. Rouwette EAJA, Größler A, Vennix JAM. 2004. Exploring influencing factors on rationality: a literature review of dynamic decision-making studies in system dynamics. Systems Research and Behavioral Science 21(4): 351–370.
20. Spector, J. M. (2006). A methodology for assessing learning in complex and ill-structured task domains. *Innovations in Education and Teaching International*, *43*(2), 109-120.
21. Spector, J. M., Christensen, D. L., Sioutine, A. V., & McCormack, D. (2001). Models and simulations for learning in complex domains: Using causal loop diagrams for assessment and evaluation. *Computers in Human Behavior*, *17*(5), 517-545.
22. Stewart, M. (2012). Joined up thinking? Evaluating the use of concept-mapping to develop complex system learning. *Assessment & Evaluation in Higher Education*, *37*(3), 349-368.
23. Weinerth, K., Koenig, V., Brunner, M., & Martin, R. (2014). Concept maps: A useful and usable tool for computer-based knowledge assessment? A literature review with a focus on usability. *Computers & Education*, *78*, 201-209.

BOOKS AND CHAPTERS

1. Cohen, S. (2006). Virtual decisions: digital simulations for teaching reasoning in the social sciences and humanities. Mahwah, N.J., L. Erlbaum Associates.
2. Driscoll, M. P. (2005). Psychology of learning for instruction (3rd Ed.). Boston: Pearson Education, Inc.
3. Gibson, D., C. Aldrich, et al. (2007). Games and simulations in online learning: research and development frameworks. Hershey, PA, Information Science Pub.
4. Herrington, J., & Oliver, R. (1995). Critical characteristics of situated learning: Implications for the instructional design of multimedia. In J. Pearce & A. Ellis (Eds.), Learning with technology (pp. 235-262). Parkville, Vic: University of Melbourne.
5. Herz, J. C., M. R. Macedonia, et al. (2002). Computer games and the military two views. Defense horizons no. 11. [Fort McNair, Washington, D.C.], Center for Technology and National Security Policy, National Defense University.
6. Jones, K. (1997). Games and simulations made easy: practical tips to improve learning through gaming. London; Stirling, VA, Kogan Page.

7. Kolb, David. (1984). Experiential Learning. Prentice Hall, Inc.
8. O'Neil, H. F. and R. S. Perez (2008). Computer games and team and individual learning. Amsterdam; London, Elsevier.
9. Prensky, M. (2001). Digital Game-based Learning. New York: McGraw-Hill.
10. Shaffer, D. W. (2006). How computer games help children learn. New York: Palgrave Macmillan.
11. Van Ments, M. (1999). The effective use of role-play: practical techniques for improving learning. London, Kogan Page.