Project Robot: A Software Simulation for Systems Engineering Education

Ross D. Arnold and Jon P. Wade
School of Systems and Enterprises, Stevens Institute of Technology, NJ
ross.arnold1@gmail.com
jon.wade@stevens.edu

Abstract: The U.S. defense industry spends billions of dollars each year developing defense systems to keep the nation and allies secure. However, the failure rate of system development is notoriously high. Even when development efforts do succeed, they often do so with cost overruns and compromises in system performance. As a result, large amounts of money are wasted in defense acquisition, leaving the nation both poorer and less secure than it could be.

Though this problem is certainly multi-faceted, one way to approach the problem is to provide better systems engineering education to engineers. Systems engineering skills, generally considered to be key to the successful development of large scale systems, often require many years to acquire. However, recent research investigates the theory that these years can be reduced through the use of simulation software.

This paper describes Project Robot, a defense systems engineering simulator designed to facilitate the acquisition of systems engineering skills at an increased rate. Project Robot was the co-winner of the 2010 Experience Accelerator international systems engineering simulator competition held at Stevens Institute of Technology, NJ. The development of this simulator is a first step towards the design and development of experience accelerating simulations and software games that push the boundaries of engineering education to the next level using modern computer software techniques. The paper introduces the concepts of systems engineering and systems thinking, then discusses the Project Robot game concept, design, theory, and implementation, including detailed screen captures. The paper concludes with a discussion of the future of Project Robot and related research efforts to improve systems engineering education through simulation.

Keywords: systems engineering, systems thinking, systems approach, system dynamics, systems engineering education, systems thinking assessment, educational games, experience accelerator, experiential learning, game-based learning, system analysis and design, systems engineering and theory, simulation

1. Introduction

Despite their importance to both national and global security, many challenges lie in the path of large-scale defense system acquisition and development. One of these challenges is to effectively build the skill sets of the engineers responsible for overseeing these highly complex, large-scale systems. These engineers often shoulder a great mantle of responsibility and can significantly affect the course and outcome of engineering projects. However, many of the most experienced among these systems engineers are leaving the workforce in the near future, leaving an experience gap behind (International Council on Systems Engineering, 2014). This predicament is not shouldered by the defense industry alone; the problem affects areas ranging from healthcare to the space program to industry and more. As highly skilled engineers retire and exit the workforce, key sources predict that a 15 year skill gap will emerge (International Council on Systems Engineering, 2014).

This skill gap opens a Pandora's Box of questions. What happens when we no longer have highly skilled engineers to oversee these massive defense projects? Will we, as a nation, still be able to maintain our military edge over our adversaries? Will we expend so much money in our defense engineering projects that other areas of the nation will suffer as a result? The political implications and affected systems are convoluted, complex and impossible to predict at present. Meanwhile, what can we do to mitigate this huge risk?

One way to approach this challenge is to educate engineering students in these areas of complex systems thinking and systems engineering. However, unfortunately the skills required by the systems engineers overseeing these systems are developed over long periods of time, and through vast experience. These are not skills traditionally taught at the university level, though some universities have begun to offer classes in these areas.

Reference this paper as: Arnold R. D. and Wade J. P., "Project Robot: A Software Simulation for Systems Engineering Education" *The Electronic Journal of e-Learning Volume 15 Issue 5 2017*, (pp410-423) available online at www.ejel.org

As one way to improve education in various systems engineering skills, research into the use of systems engineering simulation is underway (Squires, Dominick, Wade, & Gelosh, 2011; Valerdi, 2012; Wade et al., 2015; Zhang, Bodner, Turner, Arnold, & Wade, 2016). Using simulation, realistic scenarios can be offered with accelerated timescales. These scenarios provide a way for students and learners to experience the full life cycle of an engineering project, at least in some small way. This approach is similar to the use of flight simulators, tank simulators, or other types of training simulators in various other fields. Of course, simulations are not (yet) a replacement for real life experience; it will probably be quite some time before they are. However, simulations offer many advantages over other learning methods; they provide a way for learners to actually experience a scenario with many of its embedded intricacies and complexities, rather than read or hear about it through other educational materials.

As a spearhead effort to research systems engineering simulation, Stevens Institute of Technology held the Experience Accelerator competition in July of 2010. The competition challenged students from across the world to build an innovative serious game or simulation that enabled future technical leaders to gain key experience to prepare them for the demands of developing and deploying complex systems. The overall objective of each simulation was to build insights and wisdom, and hone decision-making skills by (Stevens Institute of Technology, 2010):

- Creating a "safe" but realistic environment for system level decision-making where the participants have to make the trade-offs between the need for more analysis and information and the need to "get done" with the limited time and resources available
- Providing rapid feedback by accelerating time and letting the participants experience the downstream impact and consequence of their actions and decisions

The Project Robot simulation, which is the focus of this paper, is a defense systems engineering simulator designed and developed for the competition. Project Robot tied for the $10,000 grand prize, with another simulation titled Healthcare Reborn (Dimitrov, Hess, Perkins, & Valerdi, 2010). The following sections in this paper describe some key principles of both systems engineering and systems thinking. These two skills are core to Project Robot's design. Following the descriptions of these skills, the details of the Project Robot simulator are described.

2. Systems Engineering

Systems engineering is the design and creation of complex technical systems of any sort. When speaking of systems engineering, it is often implied that the technical systems under discussion are highly complicated, such as airplanes, cars, computers, guns, jet engines, or smart phones. Thusly described, systems engineering may sound overly broad, but this is necessarily so; systems engineering often draws upon many engineering disciplines as well as interpersonal skills. On a single large-scale project, many engineers often perform systems engineering in addition to their other technical duties. However, often the title of Systems Engineer is held by only one, or just a few, engineers. A single term to describe what this Systems Engineer "does for a living" is needed. This broad, general concept of systems engineering is widely accepted in the field (BKCASE Editorial Board, 2016; International Council on Systems Engineering, 2014; Zhang et al., 2016). It is important to be able to collectively describe the skill sets of engineers doing this kind of work so that education and training efforts can be targeted towards their needs.

According to the literature, systems engineering is an interdisciplinary approach and means to enable the realization of successful systems (International Council on Systems Engineering, 2014).

Successful systems must satisfy the needs of their customers, users and other stakeholders (BKCASE Editorial Board, 2016). In the defense industry, this usually means producing cost-effective, performant, user-friendly systems that enhance national security and/or Warfighting functions.

It is often said that systems engineering and technical leadership is as much an art as a science (Zhang et al., 2016). While a traditional model of education can teach the fundamental body of 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.

Systems engineering educators are struggling to meet the growing educational demands for a workforce able to solve problems driven by accelerating technology, rapidly evolving needs, and increasing systems complexity (Bagg, Brumfield, Casey, Granata, & Jamison, 2003; Davidz & Nightingale, 2008; Squires, Wade, Dominick, & Gelosh, 2011). At the same time, there is a widening gap in industry between the need and the availability of systems engineering practitioners with the necessary experience to address these challenges (Charette, 2008).

3. Systems Thinking

A key part of successful systems engineering involves using a skill set called *systems thinking*. Systems thinking is a set of synergistic analytic skills used to improve the capability of identifying and understanding systems, predicting their behaviors and devising modifications to them in order to produce desired effects. These skills work together as a system (Arnold & Wade, 2015).

Systems thinking can be used to better understand the deep roots of complex behaviors in order to better predict them and, ultimately, adjust their outcomes. With the exponential growth of systems in our world comes a growing need for systems thinkers to tackle these complex problems. The problems we now face are stubbornly resistant to our interventions (Richmond, 1993). Now more than ever, it has become necessary to recognize the need for a paradigm shift that can carry our analytical efforts beyond the reductionist approach, and towards a more comprehensive systemic perspective (Dominici, 2012).

Systems thinking provides a holistic approach that results in greater success when describing and analyzing these kinds of complex problems (C. Liew, Foo, Lee, & Goh, 2006). Studies have clearly shown the advantages of using systems thinking when solving ill-defined, non-routine problems and making decisions in a world that exhibits a high degree of interconnectedness along with many uncertainties and complexities (Haines, 2000; Houghton, 1989; Kauffman, 1980; Resnick & Wilensky, 1998; Senge, 1990). Systems thinking has been used to tackle a wide variety of problems in different sectors (Burandt, 2011; C. Y. Liew, Foo, Goh, & Lee, 2014; Luong & Arnold, 2016; Pan, Valerdi, & Kang, 2013).

Systems engineering entails systems thinking and several other skills. These skills are exercised by design during the use of the Project Robot simulation.

4. Project Robot: Game Concept

The Project Robot simulator presents a situation in which the player is responsible, as the lead systems engineer, for the development of a large military robot. This includes a three-year design and development period followed by a ten-year maintenance period. Although such robots do not actually exist in the military today, Project Robot was built by a defense systems and software engineer and is based upon real experience in the defense engineering field. The underlying principles and systems engineering skills required to navigate the scenario are identical to those required in typical defense engineering projects.

Project Robot focuses on several systems engineering challenges. These challenges include balancing stakeholder needs both against each other and against the true needs of a system, the necessity of making tough choices without complete information, and the consequences of early system design decisions. All of these challenges are very real and are present in actual defense engineering, often arising as key barriers to success.

Project Robot presents decision-based game play and was designed with realistic but achievable goals as well as a balance between fun and learning. Rewards are score-based, and the game provides evaluation data in the form of final scores and ratings. Multiple play-throughs are encouraged to produce additional learning.
No training or preparation is required to play the game, provided that the user has a basic knowledge of computers and some basic computer skills. All necessary information and instructions are presented in the game format during gameplay as needed.

5. Project Robot: Gameplay

The Project Robot game is set in the not-too-distant future, with the player taking on the role of a chief systems engineer of a military defense system intended to be used to fight off an alien invasion. During gameplay the player can interact with a set of seven stakeholders that both control project funding and

provide information and feedback to the player. These stakeholders have different viewpoints which are often contrasting. The stakeholders may also change at random; for example, a particular senator with one set of preferences may be randomly replaced by another with a completely different set. The player must take steps to accommodate for these possible changes. The stakeholders include:

- A senator
- The player's direct supervisor
- A military equipment trainer
- A government integrator responsible for putting all the pieces together
- A contracting company responsible for the manufacturing of the robot
- A politician
- A user / Warfighter representative

This set of stakeholders is simplified, but consistent with typical defense projects. The player has the opportunity to ask a limited number of questions (Figure 1) to these stakeholders to determine their needs and preference, and then spend a limited amount of resources to choose and refine different components, sub-systems, and attributes of the giant robot (Figure 2) in order to satisfy these preferences and ultimately field a product that fulfills its purpose.

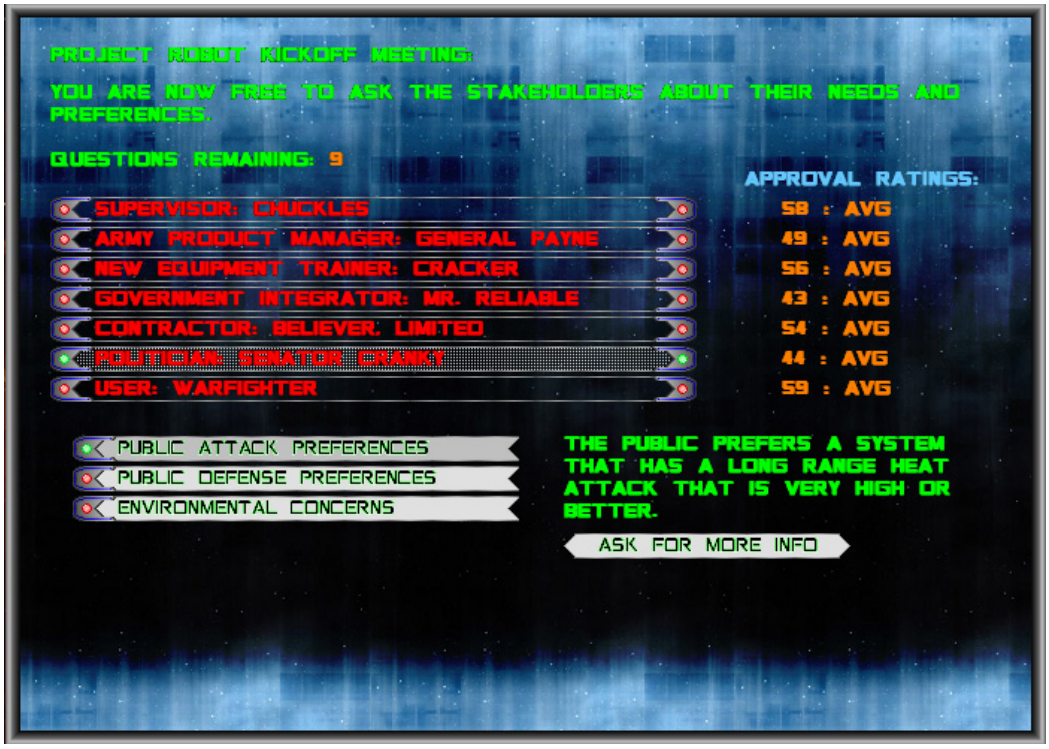

Figure 1: Stakeholder Question Screen

The names of the various stakeholders are deliberately humorous in order to provide an additional layer of enjoyment to the game. Approval ratings for each of the stakeholders are shown on the right side of the screen (Figure 1). These ratings can be accessed at any time from the overview screen (Figure 3).

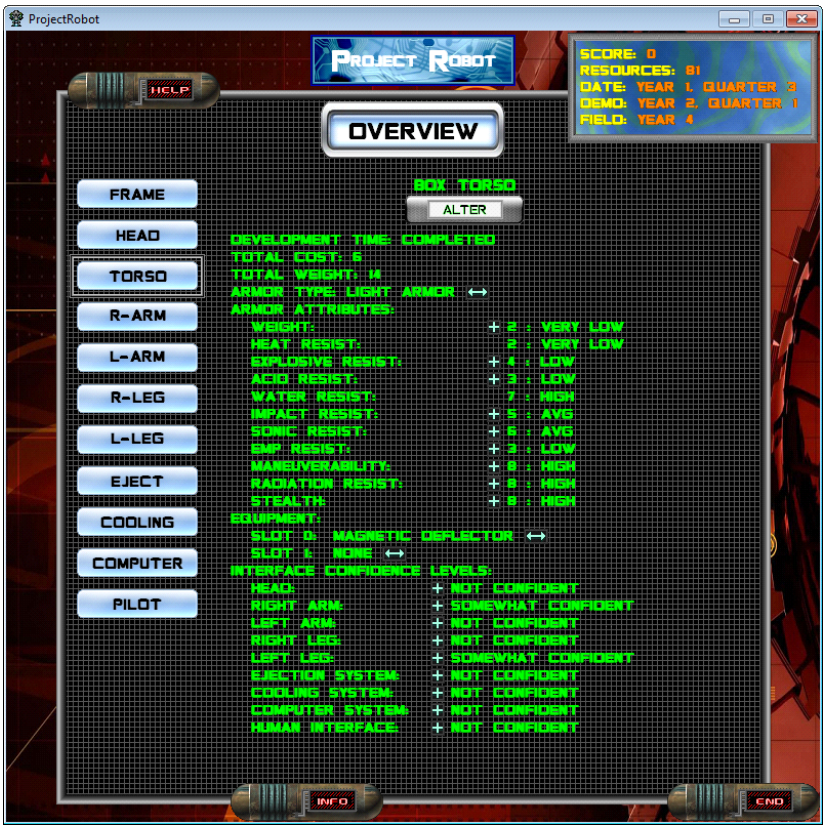

Figure 2: Robot Attributes Screen

The player is able to manipulate the robot's design through interaction with the overview screen (Figure 3) and other related screens. The overview screen shows the overall system specifications and attributes according to its current design, and allows the player to select different parts of the robot to enhance, change, or build.

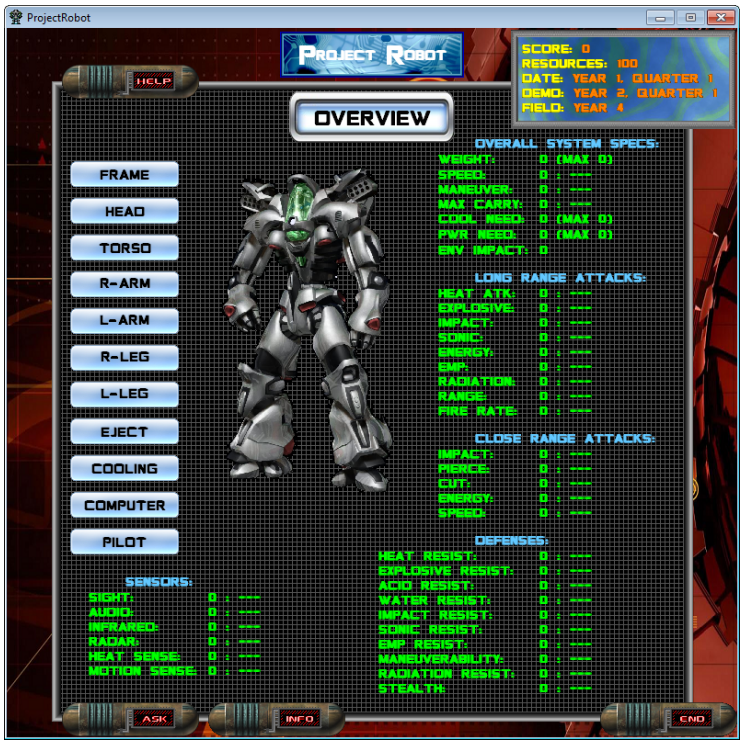

Figure 3: Project Robot Overview Screen

To measure the player's success or failure, the game keeps track of a score based on how well the player's decisions meet the needs of the stakeholders and how well those decisions meet the real needs of the system. The system's real needs are randomly generated at the beginning of a game, and are reflected to some extent in certain stakeholder needs. However, the real needs and the stakeholder needs are not identical. As in real life, engineers and managers can only take best guesses at how a system should be designed. Different stakeholders also have different priorities and perceptions of system needs. Some of these might be accurate, but some are likely very different from true user needs. A software user interface that a designer considers extremely clever might, in fact, not be appropriate for its user-base. Sometimes (uncomfortably often, in fact) systems designed by intelligent people still fail to meet their true needs when used in a real environment.

Score calculation in Project Robot places a much greater emphasis on how well the player's design choices meet the real needs of the system, with less emphasis on how well they meet the needs of the stakeholders; however, if stakeholder needs are not met, stakeholders may cut funding to the program, compromising the player's ability to build the "correct" system. In the real world, money is provided by a person or company. If this person or company is not happy, money will not come, whether or not the system is actually designed correctly. Designing and developing a technical system is a balance of keeping the money-providers happy while developing the system that the end users actually need (and the two often do not align completely). Figure 4 shows the game's stakeholder approval screen.

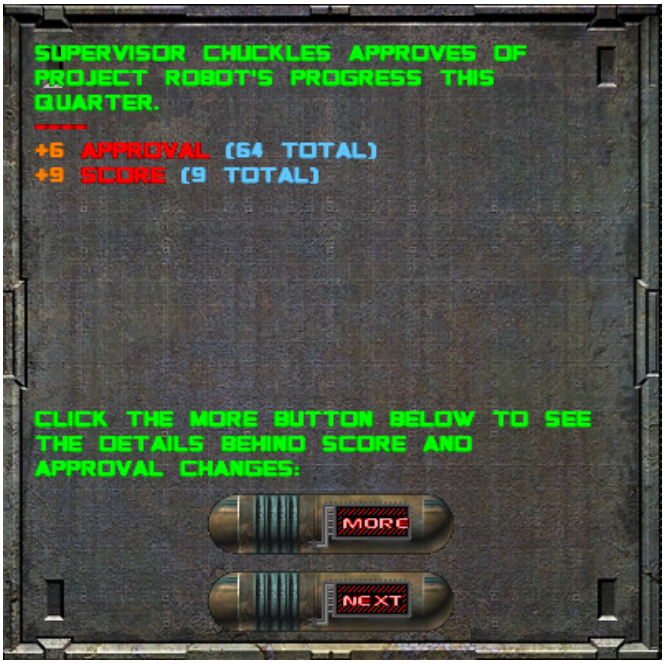

Figure 4: Approval Screen

The game is separated into two phases. Phase One is the design and development phase, and Phase Two is the fielding and maintenance phase. Phase One occurs over a three year period in-game, allowing the user twelve turns (a quarter-year per turn) to develop the robot. The user is awarded more questions to ask the stakeholders after each turn, and, of course, items with associated development times are brought closer to completion. After each year in the development phase (i.e. every four turns) an "operational demonstration" will occur, which essentially is just an in-game way of saying that the stakeholders will alter their approval ratings and settle on funding for the project for the next year. Score and additional funding is given to the player for high approval ratings. The idea behind this gameplay element is to simulate the real world fiscal year cycle that governs most military system development.

After three years of in-game time (twelve turns, as stated previously), the giant robot system is fielded and the game enters into Phase Two. Phase Two covers a period of ten years post-deployment, with one year per turn instead of one quarter (for a total of ten turns). At this point, the score is now determined only by the closeness of the system to the true needs of the field. However, stakeholder approval still governs yearly funding, so it still holds an importance to the player even though it no longer directly affects game score. Also

in Phase Two the player is no longer allowed to make major system changes, such as re-selection of sub-systems. The player may still tweak parameters and alter equipment and accessories, though the resource cost for such tweaks is vastly increased.

The game ends after the full 22 turns (Phase One, development phase, plus Phase Two, maintenance phase), and at that point the player is given a final score. The game ends in failure if the player has not selected all sub-systems by the maintenance phase (Figure 5); without all sub-systems (head, legs, engine, etc.), the robot cannot operate and thus has failed. If the player has selected all sub-systems, then poor systems engineering decisions, such as failing to spend resources to boost confidence in sub-system interfaces, simply result in a lower final score.

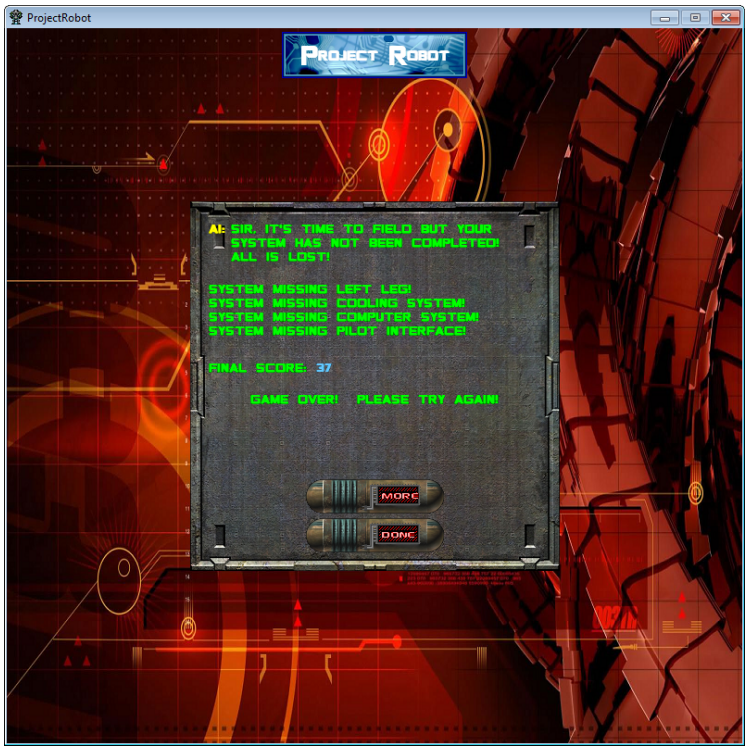

Figure 5: Failure Screen

The game is designed so that the player can replay the game after success or failure, with a different solution and different stakeholders generated at random, using lessons learned from the previous game to try to attain a higher score. This paradigm provides students with a way to directly apply what they have learned from the experience and see how much better they can do; because the solution and stakeholders are randomly generated, the game experience will be different and replay value is high.

6. Project Robot: Lessons and Learning

Project Robot is designed to support a number of learning objectives. Some of its lessons are emphasized as the primary lessons of the game, while others are learned simply through the general experience of playing through the simulation.

The first lesson is based on the fact that many systems have unclear requirements and are the result of the input of multiple people with different opinions and viewpoints; building a successful system depends on how well these viewpoints are balanced and managed. This is reinforced by the way that the game handles multiple different stakeholders with different needs, many of which are not in line with the true needs of the system.

Secondly, the player will develop an appreciation for the importance of early design and architecture decisions, learning the frustration of managing the results from a poor decision made years ago based on incomplete information. This lesson is reinforced by the way that the game handles sub-system selection and

improvement, and how changes to these systems are very resource- and time-consuming, as well as the fact that many components are restricted either cost-wise or technically by architectural decisions.

Thirdly, the player will develop an understanding that, at times, it may be necessary to make decisions and continue to move forward with system development even when full information is not available. Very often these types of decisions must be flexible to accommodate unseen circumstances. This concept is reinforced in game by the way that information is garnered from stakeholders at a staggered pace over successive turns. The concept also manifests itself in the way that the true system needs are not revealed until the system is actually fielded. When the true system needs become apparent, changes to the system may be needed even after the three year development cycle is complete. In these cases, a design that emphasizes flexibility is highly desirable. If the system was designed to be inflexible, changes may not be possible after deployment and the system may perform poorly with no reasonable avenue for improvement.

Another of Project Robot's lessons is the idea that interfaces between sub-systems must be touched and explored. Interfaces and interactions between components should not be left to chance. Spending resources on interface improvement may not actually show visible attribute improvement, so it may not seem intuitively beneficial. However, the quality of interfaces between components tend to have a large effect on overall system robustness. The game reinforces this take-away by the nature of the score assigned in the post-development phase based on interface confidence levels.

Scores in Project Robot are calculated based on external needs, including numbers (attacks, defense, sensors) and preferences, internal needs (limits such as weight, cooling power), and true needs vs. stakeholder desires. Of particular note, the needs of the Program Manager are generally the most critical during system development in order to procure sufficient funding, but these needs do not always align with the true needs of the Warfighter. The needs of the Program Manager and Warfighter generally align more closely than the needs of other stakeholders, but are still not identical and thus balancing is needed between procuring funding, and developing the correct product for the Warfighter. If the Program Manager is not happy, funding will be withheld and the program will fail. However, if the Program Manager is completely appeased, the product will actually not be the correct product for the Warfighter. The player (the Chief Systems Engineer) must mindfully balance the appeasement of the Program Manager with the needs of the Warfighter, discovering both through strategic information-gathering (using the provided questions). Additionally, the warfighter himself does not know the precise true needs of the project, and thus some randomness is introduced.

Future unimplemented plans were to include a research screen which would allow a player to allocate resources to research the true operational needs of the system. This would add an additional facet – forcing the players to choose between operational solution and stakeholder needs, instead of simply between the needs of different stakeholders (as in the submitted implementation). In the implementation submitted to the Experience Accelerator competition, this capability was not included. In future iterations of the simulation, such a capability would be desired.

Overall, the player will take away a general understanding and appreciation for many critical aspects of the system engineering process that are typically only gained through experience, as the game provides a limited but important subset of systems engineering experiences that are difficult to teach in a classroom setting.

An area for additional consideration is the use of post-game reflection as a learning tool. Although Project Robot did not implement this method, research indicates that it can be effective (Wood, 2011). A simple way to introduce this concept would be to provide players with an additional "reflection" screen upon completion (or failure) of the Project Robot scenario. This screen could be used simultaneously as a reflection for the player, and as a way to collect data about the player's usage patterns. On this screen, players could be asked a variety of appropriate questions, such as:

- What was the main reason for your success (or failure?)
- What could you have done better to improve your success next time?
- What area did you feel you spent too much time on? Not enough time on?
- Which system attribute did you feel was most important? Least important?

- Which stakeholder's opinion did you feel was the most critical? Least critical?
- What is the most significant lesson you have taken away from this play through?

These questions are just a sampling of the types of questions that could be asked. Using this method, players would have the opportunity to think and reflect upon their actions, as well as provide additional information that could be used to assess the player's learning. This information could also be leveraged to evaluate the effectiveness of Project Robot as a learning tool. For example, the significant lessons learned, and the stakeholder's opinions judged as most critical, should match up with the simulation's learning objectives. In this way, a post-game reflection would address three significant areas: learning, learning evaluation, and simulation improvement.

7. Project Robot: Learning Evaluation and Assessment

The evaluation and assessment of systems thinking and systems engineering is an area of significant ongoing research (Arnold & Wade, 2015). Project Robot provides a final score to its users, and this score can actually be used to evaluate learning over successive plays. The final score indicates the closeness of the robot's final specifications to the true needs of the system, and thus describes with some accuracy, the player's success in the simulation. If this score is recorded periodically over the course of several plays, variations in the score could be used to demonstrate that learning has occurred.

Problems arise in that it might be difficult to separate out systems engineering skill from simply skill in the game interface itself; surely over multiple plays, the player will acquire a better understanding of the Project Robot game and its interface, and therefore play more successfully. However, the variations in scores of experienced systems engineer vs. an inexperienced or non-systems engineer could be used to determine the validity of the simulation score as a learning assessment measure.

Pattern matching techniques could also be used to record the playing patterns and strategies of expert systems engineers. These patterns could then be used to analyze and evaluate the playing style of less experienced engineers. For example, more experienced engineers might focus more questions on particular stakeholders, while inexperienced engineers might distribute questions equally across all stakeholders. Research into the use of this technique for systems thinking assessment using simulation is ongoing (Arnold & Wade, 2015) and could be applied to Project Robot as well.

Usage pattern matching is likely to be effective for simulation performance assessment, but requires different validation than the use of a game score. Usage pattern matching assumes that the simulation itself accurately represents a system that experts would excel in, while non-experts would not. Although the score in the actual simulation becomes less important than the player's usage patterns, the fidelity of the simulation remains crucial. One way to validate the fidelity of the simulation could be through post-game evaluations provided to expert systems engineers; let the experts verify the system. Usage pattern matching techniques for simulation performance assessment do offer several benefits, such as a de-emphasis on the game's scoring mechanism (which could be flawed or incomplete), and a way to explore the complete picture of the way a player interacts with the system (potentially revealing more insight).

8. Project Robot: Architecture and Implementation

Project Robot was built by a single programmer in the C++ programming language. It was designed for the Microsoft Windows family of operating systems. All of the artwork is original, with the exception of the picture of the Robot itself and the background of the first screen, which is a NASA photo from the Hubble Space telescope.

By design, the game does not require a Windows install. This capability affords it the flexibility to be played by a user lacking Administrator rights. However, the game does utilize the Windows Application Programming Interface (API) and thus can only be used on Windows operating systems. Although this design allowed for a complete, robust program development at the time, modern web languages allow for the implementation of a high level of capability while facilitating cross-platform deployment. Also, the C# and Java programming languages both provide methods for cross-platform deployment. One of these technologies would be ideal for a port of the Project Robot code, or for future projects within this subject area.

9. Project Robot: The Competition

During the Experience Accelerator competition in 2010, Project Robot was exercised by several evaluators as well as demonstrated to an audience by its creator. Universally, the game was rated as a high-quality software program with a compelling gameplay style and high potential to achieve important systems engineering learning outcomes. Subjective opinions provided by graduate student evaluators revealed that the simulation actually did improve systems engineering knowledge and skills to what was considered a moderate degree. However, as noted in the above sections, a more quantifiable method of learning evaluation would be highly desirable. The evaluations resulted in the emergence of several key feedback areas.

Firstly, evaluations revealed that the game accurately depicted an appropriate defense engineering system, including multiple common pitfalls associated with building such a system. A number of strengths were identified, many of which were incorporated into future research projects (Zhang et al., 2016). Notable strengths included:

- Conflicting stakeholder needs with limited ability to discover all stakeholder desires
- Limited resources to be distributed over many different system attributes
- Changing stakeholders over time
- An accurate engineering life cycle for large defense projects
- Long-term consequences for short-term shortcuts and decisions
- A need to balance building the correct system vs. facilitating continued inflow of funds
- The need to design for uncertainty
- A visually engaging, easily-approachable user interface
- Depth of the number of available choices and options

Along with these strengths, evaluators also noted areas in which the simulation suffered or could be improved upon. Some evaluators felt that the simulation was too difficult to learn, and took too long to complete. The first play-through of the simulation typically took between 2 and 3 hours to complete. Some evaluators expressed that a shorter simulation might be more appropriate, while others indicated that the time was actually appropriate. Notably, those who felt that the simulation took too long tended to be experienced professors, while those who felt the simulation was appropriate were younger graduate students who may have been more accustomed to learning and playing new computer game (as well as the time it takes to do so).

Several evaluators expressed their opinions that the game did not feel “sufficiently game-like.” Rather, it was more of a simulation, and less of a game. This may or may not be considered a problem, depending on the intended audience and use of the simulation. For example, the simulation was likely more appropriate for a classroom session than as a publicly-available game. However, in either case, a highly engaging simulation is desirable.

Ultimately, due to the game’s accurate depiction of systems engineering activities and their associated pitfalls, along with its close adherence to the objectives of the competition and fluid gameplay style, Project Robot tied for first place in the competition.

10. Conclusion and Future Work

Many of the ideas and concepts both included in the Project Robot simulation and discussed in this paper have been incorporated into the Systems Engineering Experience Accelerator (SEEA) (Zhang et al., 2016). The SEEA is a web-based software platform with a Java-based back-end that facilitates learning through simulation. It is the spiritual successor to the Experience Accelerator competition, and has built upon the lessons learned during the competition as well as the ideas presented in the competition’s game entries. The concepts introduced in Project Robot are being furthered through the SEEA simulation and other systems engineering research efforts.

One important area of future work is to actually apply the Project Robot simulation to learners and evaluate the results. As part of the competition, Project Robot was only played by a 12 players in total. All 12 players subjectively stated that they felt they had gained Systems Engineering insight through the game. However,

applying the simulation on a larger group of subjects as a more formalized research effort would be desirable. An evaluation method that formally assesses learning is also highly desirable.

Despite Project Robot's various successes, improvement to several areas of the simulation would be desired based on feedback acquired during the competition. Also, new enhancements could significantly improve the game as well.

During the Experience Accelerator competition, the introduction and orientation of the game turned out to be relatively time-consuming, taking 30-45 minutes. A simpler, more streamlined version of the game would be desired. For example, the sub-system selections offered to the player could be much more limited than those included in the current game. Also, the different attributes of the robot could be reduced from over twenty to less than ten. Minor changes such as these would likely improve the initial gameplay experience considerably while not impacting the learning in a significant way.

Project Robot forms the basis for a game that could evolve based upon the feedback of students and instructors using the game in a classroom setting. Its architecture and design is intended to be expandable and flexible; the lessons of Systems Engineering are not lost upon the design of Project Robot itself. One exceptional feature would be a multiplayer mode in which a team of systems engineers could all develop different parts of the system in tandem, with a much more in-depth view into the selections of system sub-components, right down to specification diagrams, informational drawings and data sheets of each selectable sub-component that would appear just like real life.

Another improvement to the game could allow spin-off programs from the main program. For instance, if the giant robot performs well in the field after two years, perhaps one or more spin-off programs would be requested by the Product Manager, and the player would then be required to manage several projects all at once. This could be an advanced version of the game. It would be played by the player after successful completion of the basic version, perhaps in a successive systems engineering class or lesson. For example, in a theoretical class called "SYS 999 – Practical Applications of Systems Engineering" perhaps one class period would be based upon the students playing a session of the basic version of Project Robot, with a second game session afterwards, for a total of two sessions of the basic version. Once the students had an idea of how to play and had learned some of those systems engineering lessons, they could move onto the advanced version of Project Robot in the next class period, which could then include those spin-off systems and possibly the multiplayer mode as well as other more complex game play aspects that would be difficult for a new player to understand. Players could even import their data from the basic version and continue working with the system they previously built, perhaps with new technological developments and more options that would give them added appreciation for system flexibility.

Improvements to the Graphical User Interface are also desirable. For example, instead of a screen that lists stakeholders and questions as text, the game could include a graphical depiction of a meeting table along with pictures of the faces of the various stakeholders. The game could also include a meeting room to speak with stakeholders, an office to analyze system requirements, and a lab or set of labs where each sub-system component could be selected and refined. These improvements would create a more modern game-like environment that would likely feel more interactive and easier to learn. Additional UI improvements could include dynamic buttons that animate and move around depending on where the mouse is on screen, floating graphical tooltips for help on different game features and things of that nature that could greatly enhance the playing experience.

Another interesting improvement could be to remove the "content" from the "game engine" to allow any content to be used in the game. Such a capability would allow an instructor to take an existing Systems Engineering project, one that had failed, for instance, and plug its contents into the game. Students could then play through and experience the failure themselves, or try to make the project succeed. This type of capability is being explored in other systems thinking research such as the Systems Engineering Experience Accelerator (SEEA) (Arnold & Wade, 2017). Project Robot is currently undergoing a software port to the SEEA platform. The SEEA is the spiritual successor to the Experience Accelerator competition, and has been put to use by the United States Defense Acquisition University as well as the United Kingdom Ministry of Defence. The SEEA allows the introduction of a variety of different systems engineering scenarios, as well as the use of different simulators to produce different kinds of results. The SEEA expands the limits of systems engineering education

as it continues to grow and improve, allowing both beginner and veteran systems engineers to experience engineering projects in accelerated time according to specific learning objectives. The future plans for the SEEA involve additions of multiplayer capabilities, additions of new simulators and systems, and additions of evaluation capabilities for both systems engineering and systems thinking. This is a future in which systems engineers must no longer spend decades working complex engineering projects to learn critical "life lessons" about systems engineering, which can be costly and detrimental on a massive scale. In this future, we see a significant reduction in time, money, and material waste as engineering education improves, moving us another step closer to global sustainability.

References

Arnold, R. D., & Wade, J. P., 2015. A Definition of Systems Thinking: A Systems Approach. *Procedia Computer Science*, *44*, 669–678. http://doi.org/10.1016/j.procs.2015.03.050

Arnold, R. D., & Wade, J. P., 2017. A Complete Set of Systems Thinking Skills. In *INCOSE International Symposium 2017*. Adelaide, South Australia.

Bagg, T. C., Brumfield, M. D., Casey, C. A., Granata, R. L., & Jamison, D. E., 2003. Systems Engineering Education Development (SEED) Case Study. In *Thirteenth Annual International Symposium of the International Council On Systems Engineering (INCOSE)*. Retrieved from http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20030025438.pdf

BKCASE Editorial Board., 2016. *The Guide to the Systems Engineering Body of Knowledge (SeBOK), v1.7*. Hoboken, NJ: R.D. Adcock (EIC).

Burandt, S., 2011. Effects of an Educational Scenario Exercise on Participants' Competencies of Systemic Thinking. *Journal of Social Sciences*, *7*(1), pp.54–65.

Charette, R. N., 2008. What's wrong with weapons acquisitions? *IEEE Spectrum*, *45*(11), pp.33–39. http://doi.org/10.1109/MSPEC.2008.4659382

Davidz, H. L., & Nightingale, D. J., 2008. Enabling systems thinking to accelerate the development of senior systems engineers. *Systems Engineering*, *11*(1), pp.1–14.

Dimitrov, I., Hess, J., Perkins, L. N., & Valerdi, R., 2010. *Healthcare Reborn: a systems engineering simulator*.

Dominici, G., 2012. Why Does Systems Thinking Matter? *Business Systems Review*, *1*(1), pp.1–2. http://doi.org/10.7350/bsr.a02.2012

Haines, S., 2000. *The Systems Thinking Approach to Strategic Planning and Management*. CRC Press.

Houghton, R. S., 1989. *A Chaotic Paradigm: An Alternative World View of the Foundations for Educational Inquiry*. University of Wisconsin, Madison.

International Council on Systems Engineering., 2014. A World in Motion: Systems Engineering Vision 2025. In *24th Annual INCOSE International Symposium*. Las Vegas: INCOSE.

Kauffman, D. L., 1980. *Systems One: An Introduction to Systems Thinking*. Pegasus Communications.

Liew, C., Foo, K., Lee, E. A., & Goh, K. T., 2006. *Systems-Thinking Skills Exhibited by UiTM Sarawak Diploma Students in Solving Non-Routine Problems*. Selangor, Malaysia.

Liew, C. Y., Foo, K. K., Goh, K. T. H., & Lee, E. A. L., 2014. Is Problem Solving and Systems Thinking Related ? A Case Study in a Malaysian University. *Pertanika Journal of Social Sciences and Humanities*, *22*(3), pp.345–363.

Luong, J., & Arnold, R. D., 2016. Enhancing the Effects of Theatre of the Oppressed Techniques Using through Thinking: Reflections on an Applied Workshop. *Pedagogy and Theatre of the Oppressed Journal*, *1*(1).

Pan, X., Valerdi, R., & Kang, R., 2013. Systems Thinking: A Comparison between Chinese and Western Approaches. *Procedia Computer Science*, *16*, pp.1027–1035. http://doi.org/10.1016/j.procs.2013.01.108

Resnick, M., & Wilensky, U., 1998. Diving Into Complexity: Developing Probabilistic Decentralized Thinking Through Role-Playing Activities. *Journal of the Learning Sciences*, *7*(2), pp.153–172. http://doi.org/10.1207/s15327809jls0702_1

Richmond, B., 1993. Systems thinking: Critical thinking skills for the 1990s and beyond. *System Dynamics Review*, *9*(2), 113–133. http://doi.org/10.1002/sdr.4260090203

Senge, P., 1990. *The Fifth Discipline, the Art and Practice of the Learning Organization*. New York, NY: Doubleday/Currency.

Squires, A., Dominick, P., Wade, J., & Gelosh, D., 2011. *Building a Competency Taxonomy to Guide Experience Acceleration of Lead Program Systems Engineers*. In *9th Annual Conference on Systems Engineering Research (CSER), Redondo Beach CA.*

Stevens Institute of Technology., 2010. Experience Accelerator: An Interdisciplinary Student Competition. Hoboken, NJ.

Valerdi, R., 2012. Developing Systems Thinking Competencies through Facilitated Simulation Experiences. In *USC CSSE Annual Research Review*.

Wade, J., Watson, W., Bodner, D., Kamberov, G., Turner, R., Cox, B., McKeown, J., 2015. *Developing the Systems Engineering Experience Accelerator (SEEA) Prototype and Roadmap*. Hoboken, NJ.

Wood, K., 2011. Simulation Video Games as Learning Tools: An Examination of Instructor Guided Reflection on Cognitive Outcomes. Dissertation, Georgia State University, GA. Retrieved from http://scholarworks.gsu.edu/msit_diss/77

Zhang, P., Bodner, D. A., Turner, R. G., Arnold, R. D., & Wade, J. P., 2016. The Experience Accelerator: Tools for Development and Learning Assessment. In *2016 ASEE Annual Conference & Exposition*. New Orleans, LA: American Society for Engineering Education.

Appendix A: Experience Accelerator Objectives and Guidance

The following objectives were offered as part of the Experience Accelerator Competition of 2010 (Stevens Institute of Technology, 2010). This appendix describes how Project Robot fulfilled each objective.

Creating a "safe" but realistic environment for decision-making where the participants have to make the trade-offs between the need for more analysis and information and the need to "get done" with the limited time and resources available:
This objective is one of the primary challenges presented by Project Robot; with a limited number of questions to ask stakeholders each turn and a time limit in terms of when the system must be complete, the player must make trade-offs between learning more information about stakeholder needs and system needs and simply making the best choice possible at any given time despite lack of information.

Providing rapid feedback by accelerating time and letting the participants experience the downstream impact and consequences of their actions and decisions:
Project Robot is designed to directly meet this objective by its format of a 13 year system life cycle that awards players based on actions and decisions made during system development. Most of the game score is awarded during the 10 year post-development phase but is based primarily on decisions made during the 3 year development phase, which shows players how their decisions have impacted the program success many years after the original design.

Developing the insight that even the "best" system is worthless if it does not provide the expected value in the environment in which it is being used. It is therefore important to understand the fundamental needs a system shall address, and the multidimensional context that it is developed and deployed in:
This concept is directly addressed by the score awarded in the post-deployment phase; even the “best” system, either as dictated by one or more of the stakeholders or as determined by the player (for instance a system with the highest possible long range attack and defense values) is worthless if it doesn’t meet the true needs of the system (i.e. provide the expected value), and this is reflected in the score awarded in Phase Two of the game. This shows the importance of understanding the system’s fundamental needs, which can be most closely ascertained by spending questions on the User stakeholder, who has the most intimate knowledge of what the system must do.

Developing the insight that a system's defining properties are largely a consequence of the interfaces and interactions between its constituent elements and less a result of the properties of the elements themselves. In addition, identifying the impact of early conceptual and architectural decisions on system capabilities and flexibility:
A major factor of score in the post-deployment phase of Project Robot is the confidence level of the interfaces between various sub-systems and system components, which is a part of the system that the player may spend resources on during both the development and maintenance phases of the life cycle. This shows that even a system with excellent parameters, such as high defense values and great maneuverability, is likely to fail without proper time and resources spend refining and exploring the system’s interfaces. As to the second piece of this objective, early architectural decisions have a major impact on the system in Project Robot due to both funding and development time; for instance, changing the Frame of the robot results in every other system requiring re-development, so if the Frame of the robot were changed halfway through development cycle, the player would probably not even be able to deploy and would receive a game-over at the entrance to Phase Two. The same is true for other sub-components – each takes time and funding to develop, and this time and funding is lost if major changes are made later in the development cycle, so it is very important to make flexible decisions about sub-systems that allow for different accessories, armor types, or other equipment that may be necessary to achieve system success, especially when information available is slim, such as at the beginning of the game.

The real needs are often: hidden behind perceived needs or expressions of needs in the form of (inadequate) solutions, or a smokescreen of conflicting and inconsistent "requirements,” and might evolve while the solution is being developed:
The real needs in Project Robot are hidden behind the desires and wishes of the stakeholders – no stakeholder’s preferences are perfectly in line with the “true” solution, and many are even inconsistent with each other. Additionally, some stakeholders can and will change completely partway through the game,

bringing entirely new sets of needs and requirements to the table that may not be consistent with the current stakeholder needs.

Many stakeholders have to be satisfied before the solution can even be made available to the stakeholder(s) with the "real" needs:
In order to achieve score and funding, approval ratings for each stakeholder must be taken into consideration, even when their needs may not reflect the "real" needs of the system. If these approval ratings are ignored, the project may not even get enough funding for completion, or at the very least, will not have enough funding to develop a solution that will adequately satisfy the User stakeholder after being fielded, resulting in serious score penalties.

Most system level problems are either due to issues at the interfaces, interactions between system elements, or a poor understanding of the real needs:
This is reflected in the way that Project Robot calculates and adds score. Interface confidence levels are a major factor to score, and if the system's real needs are not understood and met, score will suffer heavily as score is primarily focused on the proximity of the player's solution to the real system solution after the fielding phase.

(Early) conceptual and architectural choices for a solution will impact the ability/flexibility to address the real needs as they, as well as the understanding of them, evolve while the solution is being developed, the ability to support the solution over time, to evolve the solution to address emerging future needs, and to effectively and efficiently create system variants to address related needs:
This was covered earlier in this document to some extent, but for some additional info: As many systems in Project Robot have restrictions and requirements, and many items have an associated development time, early architecture and design decisions are very important and have major impacts down the line, as critical components could be restricted or parameters unfulfilled, resulting in compensation in some other less cost-efficient area.

Trade-offs can have many dimensions - such as technical, economic, organizational, cultural, or socio-political:
Trade-offs in Project Robot are primarily limited to technical and economic factors, with some stakeholders desiring specific cost parameters and awarding approval and score based on keeping costs within certain limits. Additional trade-offs could be included in future versions of the game, such as losing funding or score based on public approval (although this is already touched upon with the Senator stakeholder) or international approval, or organizational conflicts of interest (for instance, certain sub-systems may be developed by the system engineer's own organization and would be preferred components).

All solutions will most likely have unintended consequences on its environment once deployed:
This is not addressed in the current version. One way to include this capability in a future version is to include environmental impacts for sub-system has environmental which are reflected in score and reported in the field reports.

Integrating the different constituent components/elements of a solution will in most cases lead to surprising emergent behavior. This can be reduced by up-front considerations to understand the intended and unintended interactions between the components:
Project Robot gives the player an ability to allocate resources to research, development, and understanding of the interfaces and interactions between sub-systems, which increases their confidence levels. Those levels have a major impact on score during Phase Two of the game, acting as a multiplier in many cases, even though on the surface the player doesn't seem to be "getting anything out of" putting resources into the interfaces, as no parameters are increased. Just like real life, focusing resources on understanding interfaces and interactions is a matter of learning that it's something that simply has to be done.

The deployment of the solution and exposure to the actual end-users can often lead to the realization that it "missed the mark", or at least missed significant aspects of the real need and what the critical attributes of an acceptable solution would look like:
This is one of the primary objectives of Project Robot. As the final solution is randomly generated and not exposed to the player, the player has to try to figure out what the solution is based on conflicting stakeholder needs. Once the system is deployed, the player may find that his or her solution is vastly different from the "real" solution. If this were the case, the player's score would suffer significantly.