Software Development Risk Analysis (SDRA)

Final Report

Trevor Jaskot

Eric Jones

Leo Pacatan

George Mason University

OR 699/SYST 699 – Masters Project Proposal

May 2019

Project Sponsor

Derek Eichin, Operations Research Analyst

Office of the Deputy Assistant Secretary of the Army
For Cost and Economics

Networks, Information Systems, Software and Electronics Costing Division

EXECUTIVE SUMMARY

Background. Office of the Deputy Assistant to the Secretary of the Army for Cost and Economics (ODASA-CE) provides Army decision-makers with cost, performance, and economic analysis in the form of expertise, models, data, estimates, and analysis. The Networks, Information Systems, Software, and Electronics Costing (NISEC) division is subordinate to ODASA-CE and the lead for developing life cycle cost estimates for communications-electronics systems, major automated information systems (AIS), defense business systems (DBS) and software intensive systems. NISEC is headquartered at Fort Belvoir, VA and sponsored this study through George Mason University (GMU) Volgenau School of Engineering.

Software development programs in the Department of Defense are plagued with cost and schedule overruns that drive delay in capability delivery. NISEC does not have a sufficient analytic method for validating Program Office Estimates (POE) for these programs, presenting a challenge when attempting to mitigate cost and schedule overruns among software development projects.

This project seeks to develop, document, and deliver a baseline model that represents Department of the Army (DA) software development team activities in order to support ODASA-CE operations research analysts in validating POE for cost and schedule.

Approach. The study approach is to, first, document the concept of operations for software development teams. Documentation provides a common understanding across NISEC operations research analysis and the analysts conducting this study. It also clarifies elements of software development included and excluded from the model, which clarifies many of the inputs and assumptions as well as the architecture of the model. The MIT Agile Project Dynamics working model is leveraged as a basis for Agile Software Development processes.

Then, software development teams are modelled as a discrete event process. The discrete event model considers deterministic and stochastic aspects of the problem and simulates the execution of software projects over time. Project backlog, staff assignments, and software defects, among other state variables, are tracked throughout the simulation and output upon completion to support post processing and analysis of the information.

Lastly, a widely accessible platform is built to host the software development model. The platform ensures usability across any platform and is prepared to continual development by NISEC.

Results. This report documents the delivery of software development team concept of operation, baseline analytic model, and accessible platform. In addition to the listed products for this study, a path forward is recommended for continual development of the analytic model.

The best application of the discrete event simulation model, as delivered, is for trade-off studies for various decision in software program management. For example, staffing, prioritization of tasks, and the breakdown of the project into elements are all candidates for improving the execution of software development projects and can help identify bottlenecks or advise management policies.

Finally, the model can be used to analyze the level of risk associated with cost and schedule estimates due to the stochastic nature, which results in a distribution for measure of effectiveness outputs.

TABLE OF CONTENTS

SECTION 1 - PROJECT OVERVIEW

1.1 BACKGROUND

Software development programs in the Department of Defense (DOD) are plagued with cost and schedule overruns that drive delay in capability delivery. The nature of software development makes projecting cost and schedule challenging. Business constraints (such as staffing, leadership, and client involvement) in combination with the high level of specialization required impact project execution with intangible factors that are difficult to account for in planning. These factors cause unpredictability, even when programs controls are in place.

Software products are required in nearly every Major Defense Acquisition Program (MDAP), spotlighting the challenges in predicting requirements for cost and time. In fact, of 98 MDAPs due to be completed in fiscal year 2010, 92 overran the projected schedule. Across the 92 overrun programs, an aggregate of $402 billion was spent beyond initial projections and programs completed an average of 22 months later than the initial program completion date. An additional 12 programs were cancelled due to program overrun in the same year (Hofbauer, 2). Astronomical overruns such as these have become the norm among MDAPs and the supporting software products are a major driver in the deficiency. Delaying capability deployment to operational units can have a direct impact on the ability of the DOD to accomplish national security objectives, including contingency response and deterrence.

The DOD recently took a number of steps toward mitigating challenges with projecting cost and schedule of developing software products. First, DOD is transitioning the majority of its software teams to operating within an Agile Software Development process. Software development in the DOD historically follows a highly structured, sequential process called Waterfall Development. Programs implementing Waterfall Development structure the entire program plan around the end-state of the project as it is known at inception (Northern, 2-3). Agile development is a more robust program management structure centering on frequent and consistent delivery of operational products and continual incorporation of clients into the development and planning process.

The DOD also established the Defense Innovation Board (DIB) in 2016 to address important problems across defense acquisition and planning. The DIB is comprised of industry leaders across the nation and provide input on DOD processes based on experience in the private sector. The first, and on-going, task to the board is to advise on how to improve the software development process, specifically focusing on the ability to project required cost and schedule. The DIB provided a plethora of advice on how to manage programs and identify impending cost and schedule slip over the last two years. However, to date, solutions for developing early and accurate projections of cost and schedule have not been identified.

Analysts across the DOD are attempting to fill the void by building predictive regression models. These models are developed with data characterizing the completion of prior programs and the corresponding cost and total timeline. Predictive models have, thus far, fallen short in improving program projections. The shortcomings are likely due to the difficulty in accounting for intangible aspects of project teams and software development. Teams rotate often, even across programs in the same office, which leads to differing dynamics and variables that impact overall performance.

DOD analysts are taking the initiative to pursue a variety of analytic options to inform decision-makers when making program projections. The ability to improve the current overruns in software development projects is paramount enabling a robust defense acquisition program and supporting operational units in the deployment of new capabilities.

1.2 Stakeholders

Office of the Deputy Assistant to the Secretary of the Army for Cost and Economics (ODASA-CE) plays a major role in addressing these challenges in the Department of the Army (DA) by providing cost, performance, and economic analysis to better inform decision-makers. The Networks, Information Systems, Software, and Electronics Costing (NISEC) division is subordinate to ODASA-CE and focuses their efforts on developing life-cycle cost estimates for software intensive systems, among a variety of other program sectors. NISEC is the sponsor for this study, commissioned through George Mason University (GMU) Volgenau School of Engineering.

In addition to NISEC, two major entities are impacted by this study: program management offices (PMO) and DA operational units. The PMO kicks-off the program by using contractual requirements to plan to the development and build of MDAP products. PMOs submit Program Office Estimates (POE) for cost and schedule to NISEC, who reviews the POE. NISEC is responsible for the validation of the program plan and advising POEs, as necessary. After NISEC and PMO work together to establish a validated program plan, the PMO executes the plan through delivery of the product to operational units. Operational units begin fielding, training, and operating products upon completion of the program.

This project seeks to directly impact NISEC's ability to validate and advise POEs. Although the product of this study will distribute within the NISEC's purview, the impact traces all the way to delivering capabilities on time and on budget to enhance the capability of DA units in performing their mission.

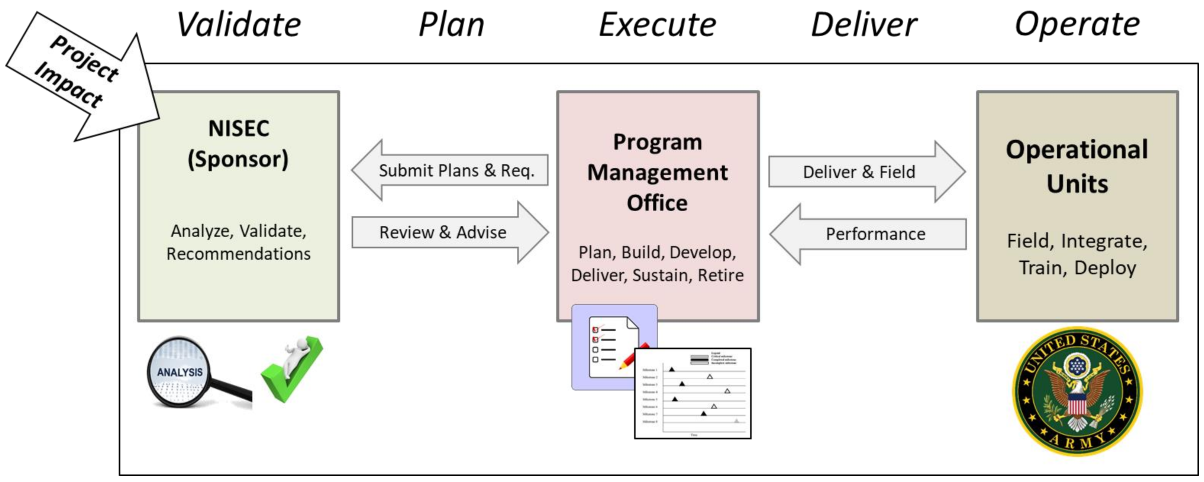


Figure 1. Project Stakeholders

1.3 Objective

The objective of this study is to develop, document, and deliver baseline model of DA agile software development teams for validating cost and schedule estimates. The products completed will be for use by NISEC operations research analysts and deployed on a medium widely accessible to enable use on DOD computers and distribute across many systems.

In support of this objective, the desired end state is the delivery of three products to NISEC analysts. First, the study team will deliver documentation of the concept of operations (CONOPs) for agile software development teams to create common understanding among analysts for use in developing models and tools. Second, a baseline analytic model reflecting the processes of Agile software development teams will be provided. Third, a platform for hosting the model and data regarding study assumptions and architecture will be provided.

In aggregate, upon achieving the end state, NISEC analysts will be prepared to implement a baseline model and further develop analytic tools in support of validating POE for cost and schedule. The study team will, lastly, provide recommendations for path forward to continue to improve the methodology implemented.

1.4 Literature Review

Software development in the DOD has a long history of troubles with cost and schedule overruns. As a result, many teams have focused on addressing various parts of the issues present. The following list are studies relevant to this project. Of the documents available, there are none that attempt to be predictive in nature and strictly look at prescribing appropriate policies. This project is intended to be an initial attempt to close that capability gap.

1.4.1 Agile Project Dynamics

The Agile Project Dynamics (APD) paper by MIT professors Firas Glaiel, Allen Moulton, and Stuart Madnick intends to develop an initial flexible system dynamics model to represent the multitudes of impacts on software development in waterfall and agile frameworks. The systems model then simulates the tradeoffs between different development frameworks. The research conducted by this team will be implemented in the inputs and considerations of the model build for this product.

Excerpt from the document below:

> “We gain insight into the dynamics of how Agile development compares to classic “waterfall” approaches by constructing a System Dynamics model for software projects. The Agile Project Dynamics (APD) model captures each of the Agile genes as a separate component of the model and allows experimentation with combinations of practices and management policies. Experimentation with the APD model is used to explore how different genes work in combination with one another to produce both positive and negative effects” (Glaiel, Moulton, Madnick, 1).

1.4.2 Handbook for Implementing Agile in Department of Defense Information technology Acquisition

This paper, developed by MITRE, will be the foundation of how agile is applied to software development in this project. The team at MITRE looks at how to qualitatively implement agile principles in DOD projects regarding information technology, but does not provide the quantitative approach this project is seeking.

Excerpt from the document below:

> "This report describes how Agile development principles can be applied to an IT systems engineering effort, and explains how an Agile methodology could be used to benefit DOD Government acquisition and development programs" (MITRE, iii).

1.4.3 Review on Traditional and Agile Cost Estimation Success Factor in Software Development Projects

This paper looks at the predictive side of cost estimation and derives factors that will be incorporated in this study. The quantitative solution to cost estimation is not addressed in this paper. We will use this paper as subject matter expertise for a check of our conclusions.

Excerpt from the document below:

> "This paper aimed to discuss success factors that influence in traditional and agile cost estimation process for software development project. Literature survey is carried out from the past researches. Then, this paper presents the success factors that bring to the successful of traditional and agile cost estimation in software development project. Realization these factors will help software development communities contribute positively to the success of traditional or agile cost estimation process in software development project" (Mansor, 965).

1.4.4 Exploring Bottlenecks in Market-Driven Requirements Management Processes with Discrete Event Simulation

This resource focuses on identifying bottlenecks in software development quantitatively. This is a key part of the descriptive analysis in this project to support understanding interdependent processes and bottlenecks. Lessons learned from these studies will be applied as applicable to understanding the driving constraints in NISEC software development teams.

Excerpt from the document below:

> "This paper presents a study where a market-driven requirements management process is simulated. In market-driven software development, generic software packages are released to a market with many customers. New requirements are continuously issued, and the objective of the requirements management process is to elicit, manage, and prioritize the requirements. In the presented study, a specific requirements management process is modelled using discrete event simulation, and the parameters of the model are estimated based on interviews with people from the specific organization where the process is used" (Host, 323).

1.4.5 Monitoring Bottlenecks in Agile and Lean Software Development Projects – A Method and its Industrial Use

This resource focuses on identifying bottlenecks in software development qualitatively. This is a key part of the descriptive analysis in this project to support understanding interdependent processes and bottlenecks. Lessons learned from these studies will be applied as applicable to understanding the driving constraints in NISEC software development teams. Additionally, we will leverage the qualitative understanding of bottlenecks and check for confirmation in our analysis.

Excerpt from the document below:

> "To achieve the desired high speed of the projects and the optimal capacity, bottlenecks existing in the projects have to be monitored and effectively removed. The objective of this research is to show experiences from a mature software development organization working according to Lean and Agile software development principles. By conducting a formal case study at Ericsson we were able to elicit and automate measures required to monitor bottlenecks in software development workflow, evaluated in one of the projects" (Staron, 1).

1.5 References

Defense Acquisition University (DAU) Press. "Systems Engineering Fundamentals." January 2001. Supplementary Text Publication. PDF File.

Glaiel, Moulton, Madnick (MIT). "Agile Project Dynamics: A System Dynamics Investigation of Agile Software Development Methods." March 2013. CISL# 2013-05. PDF File.

Hofbauer, Sanders, Ellman, Morrow (CSIS). "Cost and Time Overruns for Major Defense Acquisition Programs." April 2011. PDF File.

Modigliani, Chang (MITRE). "Defense Agile Acquisition Guide." March 2014. Release# 14-0391. PDF File.

Northern, Mayfield, Benito, Casagni (MITRE). "Handbook for Implementing Agile in Department of Defense Information Technology Acquisition." December 2010. Release# 11-0401. PDF File.

1.6 Definitions & Abbreviations

Waterfall Software Development	A highly structured, sequential software development process assuming compete knowledge of the product end state at project inception
Agile Software Development	A robust program management structure for software development based on breakdown of software elements into manageable portions and constantly integrating client feedback with frequent, operational deliverables of the product
Theme	The largest element of a software product comprising of major features and functionality in the product

Epic	A subdivision of a theme that drives the assignment and critical path of the project
User Story	A subdivision of epics that are functions desired by the client and can be completed in a single sprint
Story Point	Measure of complexity and level of effort, used to characterize the amount of work required to complete a user story
Velocity	Number of story points a project staff can complete in one sprint
Release	Major capability delivered to the client
Sprint	2-4 week period for iterative completion of user stories to support releases after multiple sprints

CI / CD	Continuous Integration / Continuous Delivery
CONOP	Concept of Operations
DA	Department of the Army
DIB	Defense Innovation Board
DOD	Department of Defense
DODAF	Department of Defense Architecture Framework
GMU	George Mason University
HTML	Hypertext Markup Language
MDAP	Major Defense Acquisition Program
MS	Microsoft
NISEC	Networks, Information Systems, Software, and Electronics Costing
ODASA-CE	Office of the Deputy Assistant to the Secretary of the Army – Cost and Economics
PMO	Program Management Office
POE	Program Office Estimate
SDRA	Software Development Risk Analysis
SME	Subject Matter Expert
SEOR	Systems Engineering and Operations Research
TBD	To Be Determined
V&V	Verification and Validation

SECTION 2 - ANALYTIC PROCESS

The systems engineering process is implemented for this study to ensure high quality products with well documented traceability to sponsor requirements. The process starts with inputs defining the problem, including sponsor needs, project objective, constraints, and measures of effectiveness.

Each aspect feeds into an iterative cycle where the project is ultimately designed and documented. First, the study team develops proposed requirements based on the sponsor needs and objective statement. Requirements are analyzed and submitted for concurrence from the sponsor. Aligned requirements drive definition of the functions the product must perform to meet these requirements. When functions are flushed out, the study team designs a product that performs all specified functions and validates that it satisfies the approved requirements.

Throughout the entire process, the study team critically analyzes decisions to select trade-offs between definitions of requirements, functions, and designs. Additionally, the project is monitored closely to ensure timeliness and team actions are aligned with project objectives.

The output of the process is a database of documented decisions that are traceable to the sponsor's original needs and the project's original objective.

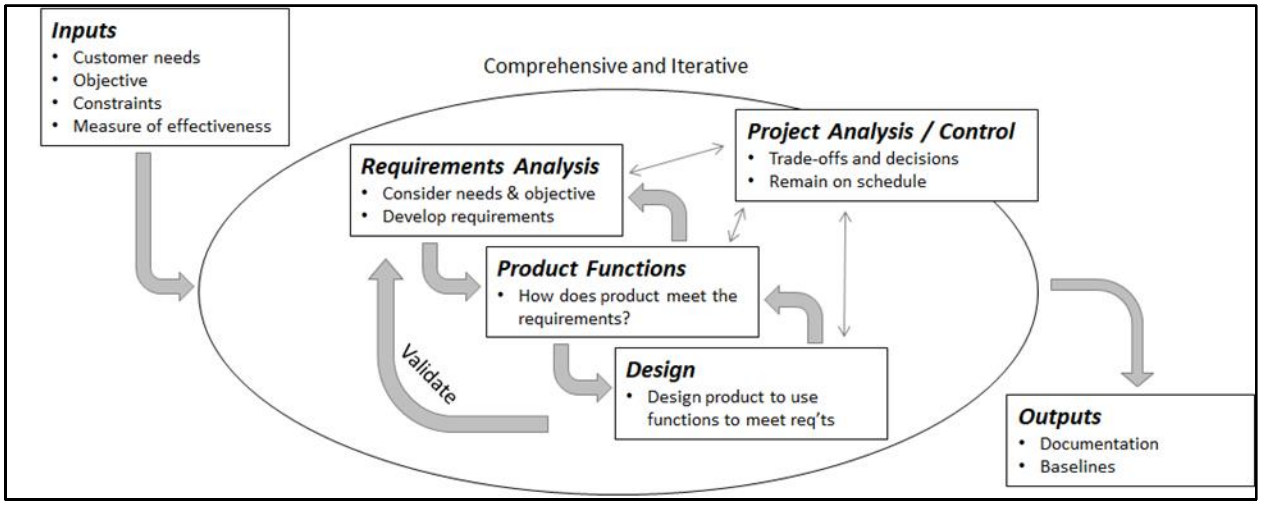

Inputs
• Customer needs
• Objective
• Constraints
• Measure of effectiveness
Comprehensive and Iterative
Project Analysis / Control
• Trade-offs and decisions
• Remain on schedule
Requirements Analysis
• Consider needs & objective
• Develop requirements
Product Functions
• How does product meet the requirements?
Design
• Design product to use functions to meet req'ts
Validate
Outputs
• Documentation
• Baselines

Figure 2. The Systems Engineering Process

2.1 REQUIREMENTS ANALYSIS

In the initial steps for this study, the team and sponsor worked to concur upon the requirements of a good product. The requirements are implemented in the consideration of operations research approaches to address NISEC challenges. The following include the requirements developed and their definitions in priority order:

Compatibility – The product must be executable on a government system or utilize software that is known to be operating in the federal government. Getting a certificate of net worthiness is considered be possible for a new software medium, but not ideal.

Transparency – Primary users will be ORAs with light database experience. Understanding of how to manipulate input, change interactions or event sequencing, and analyze output must be communicated. Showing how logical operators work or assumptions of the system are also important.

Predictive – Rough order magnitude cost intervals are desired. As close as possible estimates, or point solutions, are prescriptive in nature and don't always have the most use when expressing risk. The trade-off here is that enveloping a prescriptive model with the ability to run a design of experiments off of it can help prescribe policies.

Repeatable – To ensure analysis is replicable between multiple users a system must be present to store and catalog input data and parameters set in the model before it is run. Where stochastic variables are included, a communicated understanding of seed settings, random number generator, and ability to replicate a result is critical as well.

Descriptive – Describing the output should be tractable from modeling at the team level (e.g., which team held up the process, which team is the most utilized, etc.).

Prescriptive – The ability to evaluate a variety of agile processes and program management decisions is important to being able to advise PMOs on how to potentially improve their execution.

Completeness – As long as the documentation is transparent in nature, delivery of baseline products in this study will be adjustable with guidance from the sponsor to version 1.5, 2.0, …, N.0. Therefore, baseline product is the goal for this study.

The top four requirements were considered to be the study's primary requirements. Of the three secondary objectives, completeness is the bottom ranked which drove the study team to focus on providing a strong recommendation for the path forward for the selected approach.

2.2 Technical Approach

Several operations research approaches were considered as part of this study to satisfy the above requirements and objective: stochastic programming, system dynamics diagram, and discrete event simulation. After careful consideration against the requirements, discrete event simulation was selected due to its unique strengths in being implementable on a variety of platforms (compatibility), ease of incorporating stochastic variables in a repeatable process, and being well-suited for analyzing trade-offs to identify best actions.

The architecture and development of the discrete event simulation model will be discussed further in Section 3 – "Products."

SECTION 3 - PRODUCTS

Three products were prepared for delivery during this project. This section discusses the details of each in turn to include documentation of Agile Software Development processes, the analytic model, and the software platform used to deploy all products.

3.1 Agile Software Development Process

As discussed in the introduction, agile development is a robust program management structure centered around frequent communication and delivery of products to the client. The first element of agile is the breakdown of the project into software elements. Software elements partition the project into small, manageable pieces where the timeline can be tightly controlled. The smallest software element is a user story. A user story is a specific function or method that is required for the product. User stories can be completed in two to four weeks and are characterized by story points, which is a unit of measure for the level of effort it takes to complete the user story. Epics consist of multiple user stories and are the software element that drives the critical path in the schedule. For user stories, order of completion does not typically play a large role in tasking, whereas epics have precedent requirements, therefore creating the critical path of the project. Themes are made up of multiple epics and define major software sections that encompass full capability of a client requirement. The breakdown of the project into user stories, epics, and themes defines the entirety of the project backlog, or remaining work to be completed. Throughout project execution, the project backlog is prioritized in accordance with client priorities and program management decisions.

The other element of agile is the method for managing the project timeline. Agile development is founded on regular delivery of a functional product to the client. Agile teams deliver operational products to the client at the end of each sprint. Sprints are the smallest block of time for a delivery and typically span two to four weeks. Releases span multiple sprints and are differing in length, according to the number of sprints it takes to complete the tasks assigned to the release backlog.

Throughout the project, program managers assign prioritized product backlog to the appropriate block in time per the project timeline. This is handled by assigning epics, which drive the project schedule, to releases. This forms the release backlog and defines which user stories must be completed to complete the release. Then, within the release, user stories are sectioned into sprints for completion. Program managers and clients have constant control of the process through regular reviews and the short-term iteration of functional products.

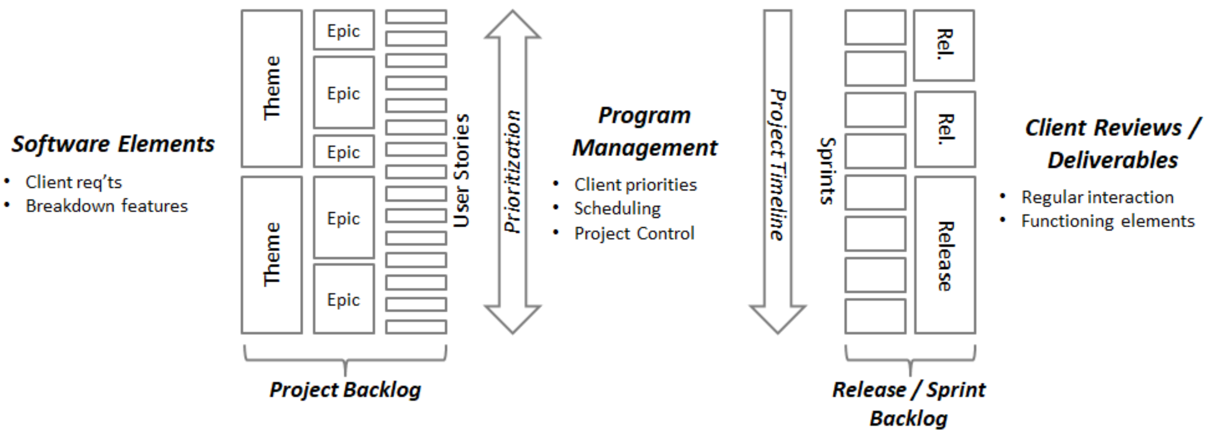


Figure 3. Agile Software Development

Several methodologies exist for implementing agile software development. A team from MIT developed a project called Agile Project Dynamics, which identifies six major functions of agile: feature driven processing, iterative and incremental delivery, refactoring, micro-optimizing, customer involvement, and team dynamics (Glaiel, 4). Agile methodologies implement a different combination of these functions, but every methodology incorporates feature driven processing and iterative, incremental delivery. Therefore, these two features are included in the baseline process used to inform the delivered analytic model.

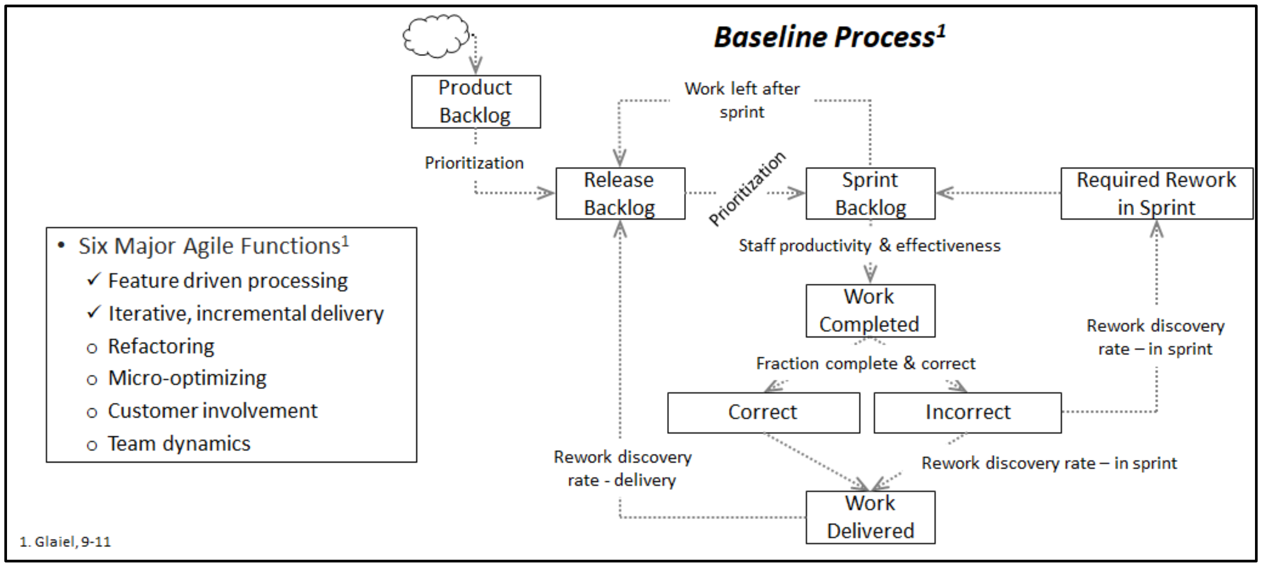

Baseline Process[1]
Product Backlog
Prioritization
Release Backlog
Work left after sprint
Prioritization
Sprint Backlog
Required Rework in Sprint
Six Major Agile Functions[1]
✓ Feature driven processing
✓ Iterative, incremental delivery
o Refactoring
o Micro-optimizing
o Customer involvement
o Team dynamics
Staff productivity & effectiveness
Work Completed
Rework discovery rate – in sprint
Fraction complete & correct
Correct
Incorrect
Rework discovery rate - delivery
Rework discovery rate – in sprint
Work Delivered
1. Glaiel, 9-11

Figure 4. Baseline Model of Agile Development

The baseline process starts with the generation of product backlog, broken down into themes, epics, and user stories. After prioritizing the project backlog, epics are assigned to the release backlog according to their sequencing and priority. Release backlog feeds user stories into the sprint backlog based on the amount of work expected to be able to be completed by the staff. This is where the iterative cycle begins. First, a portion of the sprint backlog will not be completed by the end of the sprint. These user stories will be recycled into the release backlog at the end of the sprint. Of the work that is completed, a percentage of that work may have defects introduced by the developers during production. All work that is correct flows directly into the work delivered to the client. Incorrect work has a probability of being discovered during the sprint, in which case it is re-added to the sprint backlog. If the incorrect work is not detected, it will be delivered to the client with defects. There is a probability that defects previously released to the client may be identified. When identified, the defect generates an additional task into the release backlog to remedy the defect.

This process iterates over time. As epics become available for tasking, they are added to release backlog and so on until the entire product backlog is complete and delivered to the client.

3.2 Baseline Analytic Model

The delivered baseline analytic model is developed in a typical discrete event simulation framework using the Python coding language. The model begins by setting up the simulation. Setup variables for this model are defined using a MS Excel input workbook. The workbook contains multiple sheets where the analyst defined global variables, the breakdown of project backlog, and the staff members allocated to the project. All inputs variables are read-in and used to instantiate objects reflecting staff members, themes, epics, and user stories.

Once the case is set up, the discrete event simulation begins. The event that moves the simulation forward each step, in this model, is time. Every step in the simulation moves the system forward one sprint. At the completion of each step, the model records a series of variables that define the state of the system including task progress, staffing allocation, project defects, and many others.

States of the system are tied together by a series of processes, as reflected in Section 3.1 – "Agile Software Development Process." The task prioritization and assignment are handled using deterministic algorithms in the baseline model. After assignment of tasks is done, the model evaluates the completion and correctness of the work stochastically. Incomplete work is recycled for assignment in the next step and the turn ends.

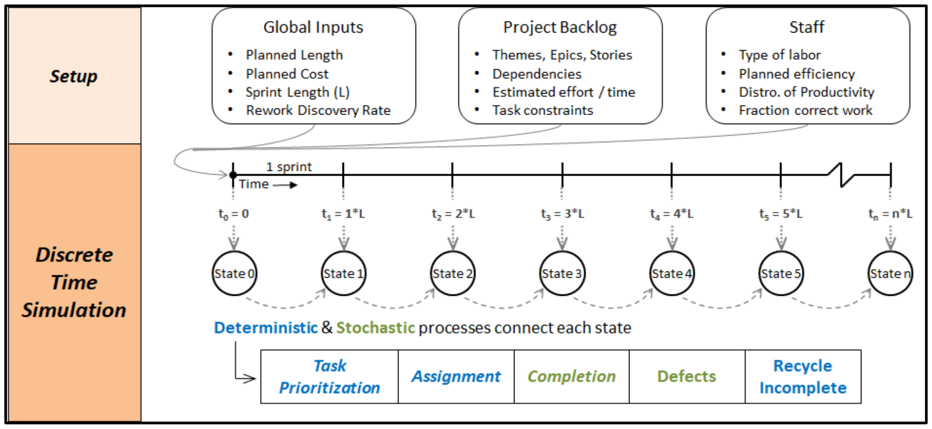


Figure 5. Discrete Event Simulation Architecture

The end of each turn triggers the state of the system to be recorded in a series of output variables. When the simulated project is complete, the state of the system at every step is output to a MS Excel file for post processing by the analyst. The output file includes information such as project progress over time, tasking over time, and defects in the project. These outputs can be post processed into a number of key measures of effectiveness including the project drawdown, staff utilization by sprint, and distribution of cost and schedule requirements for the project.

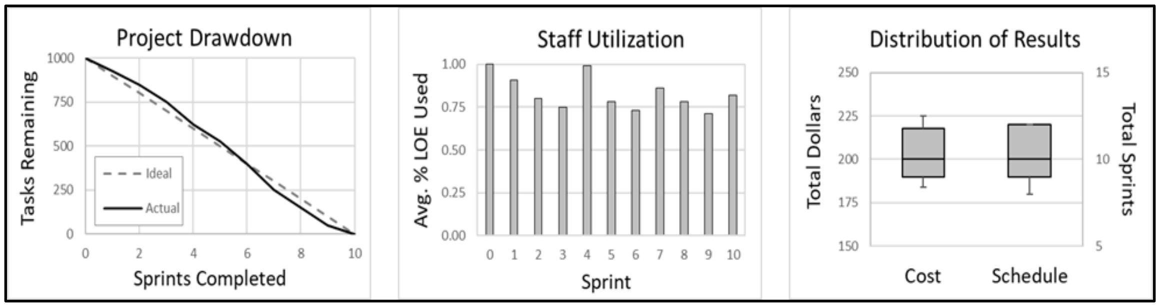


Figure 6. Output Post Processing (notional data)

The output is available to be post processed in a number of ways in order to support flexibility for the analyst to inform a variety of questions. This discrete event simulation model is effective for evaluating the baseline processes of agile software development and conducting trade-off studies for staffing, task organization, and others. A path forward for further model development is included in Section 4 – "Conclusion."

3.3 Software Platform

The team developed web access-controlled application through username and password. The project information was co-located with the analytic model, so it served two purposes on project. The first was creating a baseline analytic tool and the last was linking this site in the Systems Engineer and Operations Research (SEOR) Department website. The generate tab was designed to host user interface for the analytic model. The About tab documents the path forward and includes developer notes for continued use.

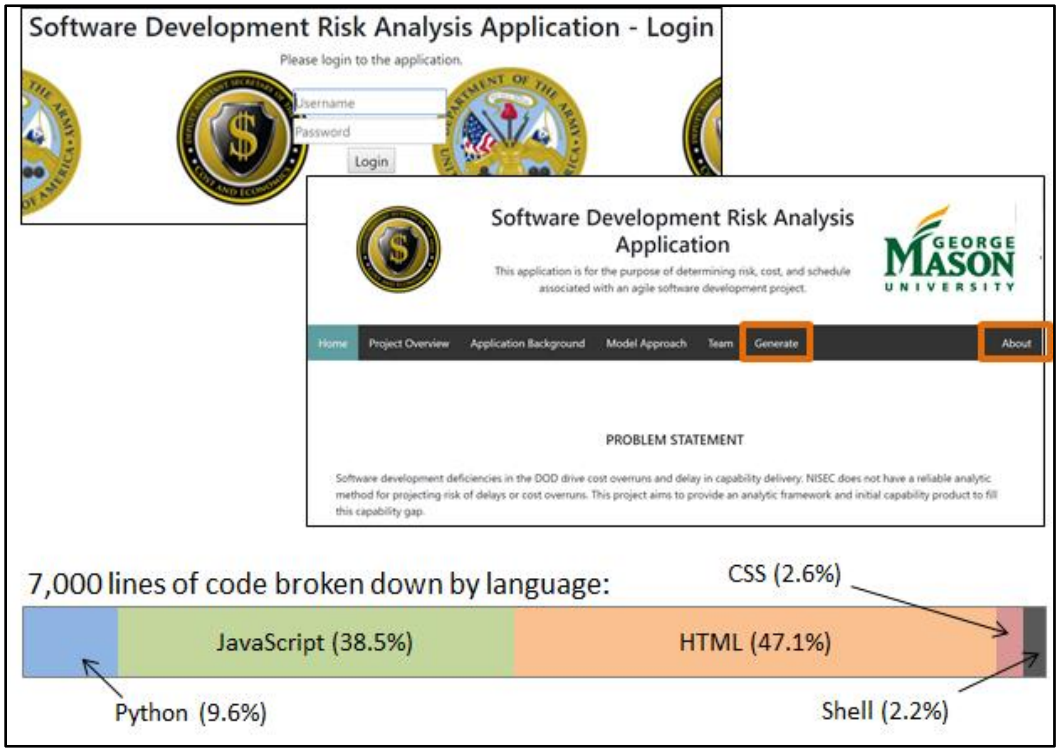
Using the processes for which the team had researched, this software application model build leveraged the agile principles of software development. The team had many influences into the selection of which programming language to select, what requirements to specify, and what compliance code from which to author. Many of the precursors and requirements the team ultimately chose to move forward with were government regulation requirements determined by DOD unclassified systems.

- Having an intimate familiarity with the unclassified systems where this application and web server code needs to work, the team realized the following items:
- Using open source where possible will produce the most diverse codebase for the customer(s)
- Using a Web Server framework was necessary for code practice and standardization so that future maintenance and upgrades are nearly seamless
- Customer requirements determined that the project scope focused primarily upon data and data structure

The software platform is a self-healing and self-circulating system that uses its own codebase to build the source from which the website is generated. To do this the application uses the Web Server framework known as Flask. Flask is a standardized set of files and packages that government unclassified systems already practice.

The framework is simply an application model design that developers can follow in order to create a stable, secure, and infrastructure agnostic program. For both customer need and data security, using a framework was the clear choice for this development.

This program roughly totals 7,000 lines of code broken and produced across different languages. They are Python, JavaScript, HMTL, CSS, and Shell. The link to application site containing the tool is: http://syst699riskapp.herokuapp.com/generate, the home-page is shown in Figure 7.


Software Development Risk Analysis Application - Login

Please login to the application.

Software Development Risk Analysis Application

This application is for the purpose of determining risk, cost, and schedule associated with an agile software development project.

PROBLEM STATEMENT

Software development deficiencies in the DOD drive cost overruns and delay in capability delivery. NISEC does not have a reliable analytic method for projecting risk of delays or cost overruns. This project aims to provide an analytic framework and initial capability product to fill this capability gap.

Figure 7. Baseline Model of Agile Development

In addition to the application is a needed suite of tools to create a fully platform agnostic program. Although many open source tools exist to create a local or production server, government systems have a strict set of data integrity requirements. Since the case for the team's development stemmed from the strict data stipulation, the team took it upon themselves to add extra scope to the project.

This scope of software development contains local scripts, integrated python security code, and a modular architecture design. Because the system was designed around data and modularity, this application can be pieced and parsed where needed. The reason why a developer would need to isolate or integrate each modular section comes from the design requirements needed for a customer set. As an example, high scalability and extreme security are needed when dealing with proprietary information. This application does not use external security for authentication, it has an introverted authorization system self-contained in the code to prevent data leakage or browser interference. By doing this the team has accomplished a few items: create the secure platform where future customers will need to house information, have a maintainable system where minimal oversight and overhead is needed, and create a simple design where advanced knowledge of coding practices are not necessary to successfully produce a working application.

In tandem to the modular design, the team integrated a number of established tools that are commonly used by open source and government personnel. These tools work in unison to create, modify, and update the application so it can provide content to the end-user. The framework is stable and consistent. Special access permissions can be granted to allow developers to continually directly update the web application. Each tool used is proven to be safe and secure and are widely implemented across the Information Technology (IT) community.

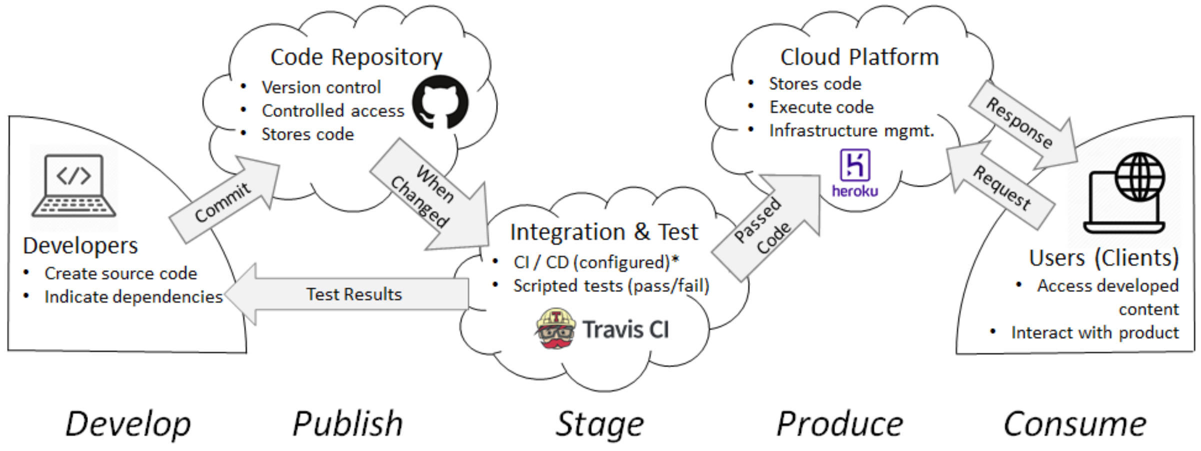


Figure 8. Software Platform Architecture

The following is a discussion on the various tools used in the software platform architecture:

Git is a command line interface with nearly infinite plugin capability and online tooling documentation for developers. It is also used in Government unclassified systems, so customer can integrate products directly.

GitHub is completely open source code repository and resource available across all operating systems. It is already in use on Government unclassified systems. Also, it can be directly processed to work with Government class systems. Here is the access link https://github.com/tjaskot/riskapp.

Travis-CI is open source continuous integration / continuous delivery (CI/CD) tool which be integrated with git/GitHub/ and Heroku. Ci/cd configuration framework can be leveraged pipeline internal to Government unclassified and classified systems. The CI/CD pipeline and integration is the heartbeat of the platform. This system contains all the success and failure logic for both the code and the cloud platform to which the code is being deployed and released. By design the CI/CD pipeline can be massively scaled to accommodate for numerous simultaneous builds and deployments. This tool determines whether code is secure, stable, and has the authorization to continue from a development to a production system. It will only allow the next step in deployment progression to commence when all code has been built, packaged, and tested on the foundations which the developer deems necessary. After deployment to the cloud platform, this tool sets quite an intelligent integration with code and platform into play. When there is a code change on any branch of the program repository, the CI/CD tool extracts the change and builds the entire cloud platform system locally to ensure that deployment and release will be seamless for the user. This whole validation step is done while the developer is coding. The benefit to this is that when the developer is ready to move to the next step in deployment the tool is already completed all success and failure tags so the developer simply needs to approve the production deployment phase without touching a single item. This tool creates an automated, seamless, and natural flow gate process whereby both the government systems and agile software development can extremely benefit.

In order to allow the team and future developers to easily integrate with this project, the cloud platform of choice was Heroku. This is an open source cloud platform that hosts any application software for public consumption. Government unclassified systems use Cloud Foundry (CF) as the cloud platform of choice. CF is built upon the open source version of Heroku. Hence why the direct correlation between the team's selection of Heroku as the application's deployment platform.

Flask is an open source package readily available on unclassified Government systems. It makes of Python coding language that both the sponsor and the study team are familiar with to develop. It automatically creates virtual servers and web hosts. Also, is very lightweight and completely packable with the GitHub codebase.

SECTION 4 - CONCLUSION

Software development projects across the DOD are plagued with cost and schedule overruns. These overruns have met astronomical numbers in recent years, which directly impacts the DOD's ability to reliably meet its national security missions. Even though a number of steps have been taken to address the problem, including transitioning to the agile strategy, establishing the Defense Innovation Board, and building predictive regression models, none have been sufficient in indicating appropriate cost and schedule projections early in the program timeline.

This study sought to address the challenges of planning for software development projects by developing products in support of NISEC analysts, who develop life-cycle cost estimates for software intensive programs to validate program office estimates for those programs. Three products were delivered to NISEC: documentation of the agile software development process, a baseline discrete event simulation model, and a software platform for hosting the model and relevant information.

Attempting to predict employee performance, in general, is risky and there is no ideal operations research method for approaching this problem. Predictive regression models fall short because of their inability to adapt the model to new intangible variables as projects teams and clients shift between projects. Discrete event simulation is, likewise, risky because of the number of input variables that must be tuned to precise values (which are not easily measured or known) for the model to become predictive.

Best applications of this discrete event simulation model include trade-off studies staffing, prioritization of tasks, and project breakdown. In this capacity, the model is effective in identifying bottlenecks in the project, whether caused by task sequencing or staff allocation. Additionally, the model is particularly efficient at proving prescriptive information in support of making trade-off decisions. The stochastic nature of the model results in a distribution of results, which allows the analyst to identify the risk involved with various decisions.

The products developed in this study are baseline products and prepared for further development by NISEC analysts. The study team recommends a series of items for analysts to improve the baseline model and expand beyond the baseline.

Two improvements are recommended for the baseline model. First, it is recommended to improve values and distributions coded into the model to reflect empirical data. Additionally, the team recommends integrating a metaheuristic method for assignment of project tasks to ensure the schedule most closely represents how a program manager would allocate tasks.

The model can also be expanded beyond the baseline to include the remaining four agile functions: refactoring, micro-optimization, client involvement, and team dynamics. Including these factors would allow for greater flexibility when evaluating various agile methodologies.

The last recommendation by the team is to continually integrate model updates into the provided software platform for distribution and documentation across analysts using the tool.

Even with potential upgrades, the implementation of this, and any, operations research tool is risky when used to predict the performance of people. However, if applied in the appropriate fashion, the products delivered in support of this study provide a powerful tool for identifying challenges in software development and allowing NISEC analysts to inform decision-makers on program estimates for cost and schedule.

APPENDIX A – DELIVERED FILES AND FILE STRUCTURE

Gitignore prevents all Python local cache from being uploaded to either GitHub or the ci/cd pipeline or Heroku.

Travis.yml connects the GitHub repository to the Travis CI platform and integrates all versions of choice (Python 2.7 and 3.5). It verifies that all code is functional and working prior to it being deployed to cloud. Declarative ci/cd syntax can take files and auto generate a pipeline for taking the GitHub code and then builds, tests, and deploys the code base to Heroku.

Procfile is the Heroku file that tells the platform which type of application to host (web application) and the name of the file to run.

Manage.py is a blank file for Heroku's knowledge to take the code and create a virtual container management and orchestrator.

Requirements.txt lists all of the packages which the application depends upon. This file tells GitHub, Travis CI, and Heroku which versions of tooling they should download form the internet. Also, it can be modified to pull from any resource and the government can choose which internal endpoints to download packages from. All packages are currently available in unclassified systems.

Riskapp.py is a Python file used by flask to create the web application. It contains all code which interfaces between the end-user and the repository itself. Also, it validates that all endpoints are available, accessible, and secures endpoints of application. The additional secure app asserts that all URL endpoints have not been intercepted nor modified to non-specified endpoint.

Riskapp.pyc is the compiled file of the riskapp.py and it is used by developers to run the codebase on a local host, and validates no current errors exists in the program or the codebase.

Runtime.txt is the selected version of Python which the user or customer can define. It tells Heroku which version of Python should be executed during the runtime within the container post-deployment of application.

Setflaskvars.sh is interactable cli tool made to export all necessary variables for a developer when local testing needs to be used. This turns developer's codebase into plain English user interactable cli via any operating windows explorer.

Testpython.sh is Python script to be run internal to the Heroku container prior to runtime execution of code.

Env is autogenerated Python virtual environment used by the developer to test all changes locally prior to pushing code change. It can be used with any version of Python to test cross-platform agnostic capability of application. It hosts a local server on the developer's machine for all Python testing purposes.

Static houses all static variable and user files. All files under the css folder contain browser rendered visuals. All files except for common.css are used only for the corresponding URL route in riskapp.py and a file under the /templates/directory (example: about.css corresponds to the about.html file). The common.css is sourced into each HTML file because everything within common.css has values used more than one-time across multiple files.

Files/ is a directory for all data and inputs. Also, it is available for all outputs and exported data.

Images/ is a directory for all static image files used by the application. All images in this folder sourced in an HTML file and hosted by flask to create reactive design and interactable website visuals.

Js/ is similar to that of the css directory, the js folder is used for all static JavaScript files which the user's browser will be rendering. Each file corresponds to an HTML file in the /templates/folder (example: about.js corresponds to the about.html file). Common.js is sourced into each HTML file because values within common.js are used more than one-time across multiple files.

Templates are all files in within this directory have a corresponding application URL route (example: login.html is associated with the /login URL endpoint). Index.html is associated to the index route loaded on URL request, which means this file is loaded when users goes to the webpage link from their browser session. It houses each HTML file loaded into the user's browser. Each file has the actual document object model that is loaded into the webpage, which the end-user sees on their computer.