An examination of the theoretical basis for agile engineering using function-behavior-structure framework and agent-based modeling

Mitchell James Bott

Follow this and additional works at: https://louis.uah.edu/uah-dissertations

Recommended Citation

This Dissertation is brought to you for free and open access by the UAH Electronic Theses and Dissertations at LOUIS. It has been accepted for inclusion in Dissertations by an authorized administrator of LOUIS.
AN EXAMINATION OF THE THEORETICAL BASIS FOR AGILE ENGINEERING USING FUNCTION-BEHAVIOR-STRUCTURE FRAMEWORK AND AGENT-BASED MODELING

by

Mitchell James Bott

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in The Modeling and Simulation Program to The School of Graduate Studies of The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2019
In presenting this dissertation in partial fulfillment of the requirements for a doctoral degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this dissertation.

_________________________________________
(student signature)

10-18-19
(date)
DISSERTATION APPROVAL FORM

Submitted by Mitchell James Bott in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Modeling and Simulation and accepted on behalf of the Faculty of the School of Graduate Studies by the dissertation committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Modeling and Simulation.

Date

Dr. Mikel Petty
Committee Chair

Dr. Bryan Mesmer
Advisor

Dr. Harry Delugach

Dr. Paul Collopy

Dr. Letha Etzkorn

Dr. Harry Delugach

Dept. Chair

College Dean

Graduate Dean
ABSTRACT

The School of Graduate Studies
The University of Alabama in Huntsville

Degree Doctor of Philosophy  Program Modeling and Simulation

Name of Candidate Mitchell James Bott.
Title An Examination of the Theoretical Basis for Agile Engineering using Function-Behavior-Structure Framework and Agent-Based Modeling

Traditional systems engineering methods have been shown to not scale well to very large and complex projects. There is a need for methods that can adapt to changing technology, deliver robust solutions, and provide consistent results. Agile methods, used in the software development domain, offer one possible solution to this need. Often, new methods are developed and deployed with little examination and testing. This is partly due to the impracticality of thorough testing of engineering methods on large and complex projects. This research uses agent-based modeling and simulation to examine the suitability and advantages of Agile methods for large and complex engineering projects. The agents are modeled using Function-Behavior-Structure framework. Novel, agent-based simulations are created where the agents transition through the cognitive states of the Function-Behavior-Structure framework to develop system designs. Models and simulations are verified against data from empirical studies and are then used to examine any benefits that Agile methods have over the more traditional waterfall method. Simulations of a software design show that Scrum likely has benefits over waterfall for loosely coupled systems like software including reduction in defects, less time needed to complete the project, and greater efficiency. Scrum may have negligible benefit over waterfall for highly coupled systems with simulation metrics indicating similar performance.

Abstract Approval: Committee Chair

Department Chair

Graduate Dean
ACKNOWLEDGMENTS

First and foremost, I would like to acknowledge and thank my wife, Lacey, for her support, patience, and encouragement in this work. This endeavor took much longer than I originally planned and I put much of our life on hold while I worked every evening on my degree. I owe her a tremendous debt for her help in getting me to this point. She has been a great partner and support to me in performing this research and I could not have done it without her help.

I would like to thank Dr. Bryan Mesmer for the countless phone conversations, advice, guidance, and encouragement in putting this research together. He was tireless in his positive attitude, rigorous in his technical evaluations and guidance, and a great sounding board for ideas.

I would also like to thank Dr. Mikel Petty for his leadership and direction in getting this work started. He was instrumental in helping me find the path that led me to this research and was great to work with.

I would also like to thank the other members of my committee, Dr. Harry Delugach, Dr. Paul Collopy, and Dr. Letha Etzkorn for their feedback and direction on my work. They greatly enhanced the quality and rigor of this research.

I would like to thank my employer, Northrop Grumman, for supporting me with a generous tuition reimbursement policy, which funded the majority of my doctoral education. I would also like to thank I/ITSEC for the generous RADM Fred Lewis scholarship that provided a portion of the funding for my education.
# TABLE OF CONTENTS

1.0 Introduction ................................................................................................................. 1

1.1 Overview ...................................................................................................................... 1

1.2 Motivation .................................................................................................................. 2

1.3 Research Hypotheses ................................................................................................. 4

1.4 Contributions ............................................................................................................. 6

1.5 Organization of Dissertation ..................................................................................... 7

2.0 Review of the Literature .............................................................................................. 9

2.1 Overview ..................................................................................................................... 9

2.2 Agile Processes .......................................................................................................... 11

2.2.1 Agile Processes: Extreme Programming and Scrum ............................................. 11

2.2.2 Agile Processes for Systems Engineering ........................................................... 12

2.2.3 Believed Benefits of Agile Approaches ................................................................. 13

2.3 Complex System Science .......................................................................................... 13

2.4 Decision Analysis ...................................................................................................... 15

2.5 Agent-Based Modeling .............................................................................................. 16

2.6 Models of Designers ................................................................................................. 17

2.6.1 FBS Model of Design ........................................................................................ 17

2.6.2 The Situated FBS Framework ............................................................................. 19

2.6.3 FBS Validation ...................................................................................................... 19

2.7 Design synthesis process .......................................................................................... 20

2.7.1 Systems Engineering “V” .................................................................................... 20

2.7.2 Waterfall .............................................................................................................. 23

2.7.3 Agile .................................................................................................................... 23
2.8 Model-Based Systems Engineering .............................................................. 24
2.9 Value-Based Engineering ............................................................................. 26
2.10 Summary ...................................................................................................... 28
3.0 Research Methodology ............................................................................... 29
3.1 Overview ...................................................................................................... 29
3.2 Research Question 1: Do theories exist that support the use of the Agile Scrum framework for systems engineering? .......................................................... 31
  3.2.1 Objective 1: Investigate the Theoretical Roots of Agile Scrum .......... 31
  3.2.2 Objective 2: Investigate How Scrum Accomplishes Systems Engineering Tasks ............................................................................................................. 32
3.3 Research Question 2: What benefits does the Agile Scrum framework have over waterfall processes? In what ways does the Agile Scrum framework fall short of waterfall processes? ......................................................................................... 33
  3.3.1 Objective 1: Establish Foundation for Modeling and Simulation Effort.... 33
  3.3.2 Objective 2: Develop a Simulation of a Simply Supported Beam Design . 33
  3.3.3 Objective 3: Develop a Simulation of a Launch Vehicle Development..... 33
  3.3.4 Objective 4: Develop a Simulation of a Large Software Product Development ................................................................................................................. 34
  3.3.5 Objective 5: Analyze Simulation Data ....................................................... 34
  3.3.6 Objective 6: Develop Rationale for Significant Differences in Waterfall and Agile Simulation Results ......................................................................................... 34
  3.3.7 Objective 7: Examine the Results to Determine What Significant Difference Exist between Agile and Waterfall Approaches .................................................. 34
3.4 Summary ...................................................................................................... 35
4.0 Theoretical Roots of Agile Scrum ................................................................ 36
4.1 Overview ...................................................................................................... 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2 Methodology</td>
<td>37</td>
</tr>
<tr>
<td>4.2.1 Literature Review of Scrum</td>
<td>37</td>
</tr>
<tr>
<td>4.2.2 Literature Review of Theories Related to Scrum</td>
<td>38</td>
</tr>
<tr>
<td>4.2.3 Relationships between Scrum and Complex System Science</td>
<td>38</td>
</tr>
<tr>
<td>4.2.4 Relationships between Scrum and Decision Analysis</td>
<td>39</td>
</tr>
<tr>
<td>4.2.5 Scrum in Systems Engineering</td>
<td>39</td>
</tr>
<tr>
<td>4.3 History of Scrum</td>
<td>39</td>
</tr>
<tr>
<td>4.4 Links between the Attributes of Scrum and Established Theories</td>
<td>41</td>
</tr>
<tr>
<td>4.4.1 Previously Identified Relationships between Scrum and Complex System Science</td>
<td>41</td>
</tr>
<tr>
<td>4.4.2 Newly-Identified Relationships between Scrum and Complex System Science</td>
<td>42</td>
</tr>
<tr>
<td>4.4.3 Newly Identified Relationships between Scrum and Decision Analysis</td>
<td>45</td>
</tr>
<tr>
<td>4.4.4 Summary of Relationships between Scrum and Theory</td>
<td>46</td>
</tr>
<tr>
<td>4.5 The Use of Scrum for Systems Engineering</td>
<td>48</td>
</tr>
<tr>
<td>4.5.1 Relationship with INCOSE handbook</td>
<td>48</td>
</tr>
<tr>
<td>4.5.2 Difficulties in Implementing Scrum for Systems Engineering</td>
<td>51</td>
</tr>
<tr>
<td>4.5.3 Agile SE Enabled Through MBSE</td>
<td>52</td>
</tr>
<tr>
<td>4.6 Summary</td>
<td>53</td>
</tr>
<tr>
<td>5.0 Modeling and Simulation Proof of Concept</td>
<td>54</td>
</tr>
<tr>
<td>5.1 Overview</td>
<td>54</td>
</tr>
<tr>
<td>5.2 The Agent Model</td>
<td>55</td>
</tr>
<tr>
<td>5.3 Proof of Concept Simulation Design</td>
<td>57</td>
</tr>
<tr>
<td>5.3.1 Pseudo-Waterfall System Design Process Simulation</td>
<td>59</td>
</tr>
<tr>
<td>5.3.2 Pseudo-Agile System Design Process Simulation</td>
<td>59</td>
</tr>
</tbody>
</table>
10.3.4 FBS Model Calibration .................................................................................. 142

10.4 Summary ........................................................................................................... 142

APPENDIX 1: Face Validation Survey ....................................................................... 150

References ................................................................................................................ 144
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1-1: Summary of large and complex systems engineering challenges [5], [9], [10]</td>
<td>3</td>
</tr>
<tr>
<td>Figure 1-2: Overview of Proposed Research</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2-1: Map of relations between research objectives and literature review topics</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2-2: Illustration of complex behavior</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2-3: The Function-Behavior-Structure Model</td>
<td>18</td>
</tr>
<tr>
<td>Figure 2-4: The traditional systems engineering V</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3-1: Research Methodology Mind Map</td>
<td>31</td>
</tr>
<tr>
<td>Figure 4-1: Overview of Research into Theoretical Roots of Agile Scrum</td>
<td>37</td>
</tr>
<tr>
<td>Figure 4-2: Simple Process with PID controller</td>
<td>44</td>
</tr>
<tr>
<td>Figure 4-3: Process behavior with matched process and control model</td>
<td>44</td>
</tr>
<tr>
<td>Figure 4-4: Process behavior with mismatched process and control model</td>
<td>44</td>
</tr>
<tr>
<td>Figure 4-5: Summary of Relationships between Agile Scrum and Theories</td>
<td>48</td>
</tr>
<tr>
<td>Figure 4-6: Traditional Systems Engineering Process</td>
<td>49</td>
</tr>
<tr>
<td>Figure 4-7: Overview of INCOSE Systems Engineering Processes executed through Scrum with key MBSE enablers</td>
<td>51</td>
</tr>
<tr>
<td>Figure 5-1: Chapter 5 Overview</td>
<td>55</td>
</tr>
<tr>
<td>Figure 5-2: FBS Model Setup for Markov Simulation with Example State Transition Probabilities</td>
<td>56</td>
</tr>
<tr>
<td>Figure 5-3 Revenue vs. FOS function</td>
<td>58</td>
</tr>
<tr>
<td>Figure 5-4 Pseudo-waterfall beam design process</td>
<td>59</td>
</tr>
<tr>
<td>Figure 5-5 Pseudo-Agile beam design process</td>
<td>61</td>
</tr>
<tr>
<td>Figure 5-6 Version Space Learning Example</td>
<td>63</td>
</tr>
<tr>
<td>Figure 5-7 Design space example with 2 design variables</td>
<td>64</td>
</tr>
<tr>
<td>Figure 5-8 Examples of design space and distributions of design variables</td>
<td>65</td>
</tr>
<tr>
<td>Figure 5-9 Materials engineer effort for pseudo-waterfall process. Average is 19.3 four-hour sessions</td>
<td>67</td>
</tr>
<tr>
<td>Figure 5-10 Materials engineer effort for pseudo-Agile process. Average is 14.7 four-hour sessions</td>
<td>68</td>
</tr>
</tbody>
</table>
Figure 5-11 Mechanical engineer effort for pseudo-waterfall process. Average is 13.5 four-hour sessions ................................................................. 69
Figure 5-12 Mechanical engineer effort for pseudo-Agile process. Average is 17.6 four-hour sessions ........................................................................ 69
Figure 5-13 Total number of 4-hour design sessions needed to complete a beam design using a pseudo-waterfall process. Average is 32.9 ............................................... 70
Figure 5-14 Total number of 4-hour design sessions needed to complete a beam design using a pseudo-Agile process. Average is 18.4 ........................................... 71
Figure 6-1: Chapter 6 Overview .......................................................................................................................... 73
Figure 6-2: Example of FBS Calibration Methodology ......................................................................................... 76
Figure 6-3: R->F task histogram and gamma distribution fit ............................................................ 79
Figure 6-4: F->Be task histogram and gamma distribution fit ........................................................... 80
Figure 6-5: Be-S task histogram and gamma distribution fit ............................................................. 81
Figure 6-6: S->D task histogram and gamma distribution fit ............................................................. 82
Figure 6-7: Type I Reformulation task histogram and gamma distribution fit ......................... 83
Figure 6-8: Type II Reformulation task histogram and gamma distribution fit ............................... 84
Figure 6-9: Type III Reformulation task histogram and gamma distribution fit ................................ 85
Figure 6-10: Mean Performing Agent Model .......................................................................................... 86
Figure 6-11: Lowest Possible Performing Agent ................................................................................. 87
Figure 6-12: Highest Possible Performing Agent .................................................................................. 87
Figure 7-1: Chapter 7 Overview .................................................................................................................... 91
Figure 7-2: Representation of team and program structure for software program design simulation ................................................................................................................. 92
Figure 7-3: Launch Vehicle Design Organization ...................................................................................... 93
Figure 7-4: Notional Waterfall Design Process ...................................................................................... 95
Figure 7-5: Agile Process Overview ........................................................................................................... 96
Figure 7-6: Notional Agile Design Process ............................................................................................ 97
Figure 7-7: Illustration of how design variables are tied to FBS states ................................................. 103
Figure 7-8: N^2 diagram of launch vehicle design team simulation ...................................................... 104
Figure 7-9: Notional Depiction of Launch Vehicle ................................................................................... 105
Figure 7-10: Free body diagram of launch vehicle .................................................................................. 106
Figure 7-11: Design Variable Selection for the Coupled Launch Vehicle Design (3 of 9 design variables shown) ........................................................................................................ 109
Figure 8-1: Chapter 8 Overview ............................................................................................ 112
Figure 8-2: Histogram of time to complete a software program design using waterfall processes ........................................................................................................................................ 113
Figure 8-3: Histogram of time to complete a software program design using Agile processes ........................................................................................................................................ 114
Figure 8-4: Histogram of effort hours needed to complete a software program design using waterfall processes ........................................................................................................................................ 115
Figure 8-5: Histogram of effort hours needed to complete a software program design using Agile processes ........................................................................................................................................ 116
Figure 8-6: Histogram of time spent doing rework for a software program design using waterfall processes ........................................................................................................................................ 117
Figure 8-7: Histogram of time spent doing rework for a software program design using Agile processes ........................................................................................................................................ 118
Figure 8-8: Histogram of time to complete a launch vehicle design using a waterfall process ........................................................................................................................................ 120
Figure 8-9: Histogram of time to complete a launch vehicle design using an Agile process ........................................................................................................................................ 121
Figure 8-10: Histogram of effort hours needed to complete a launch vehicle design using a waterfall process ........................................................................................................................................ 122
Figure 8-11: Histogram of effort hours needed to complete a launch vehicle design using an Agile process ........................................................................................................................................ 123
Figure 8-12: Histogram of time spent doing rework for a launch vehicle design using waterfall processes ........................................................................................................................................ 124
Figure 8-13: Histogram of time spent doing rework for a launch vehicle design using an Agile process ........................................................................................................................................ 125
Figure 9-1: Chapter 9 Overview ............................................................................................ 129
Figure 10-1: Identified and potential future relationships between Agile Scrum and theory. * - denotes a relationship that was previously established. ............................................. 141
LIST OF TABLES

Table | Page
--- | ---
Table 4-1: Summary of relationships between scrum, complex system science, and decision analysis | 47
Table 4-2: Comparison between the INCOSE and Scrum processes | 50
Table 5-1: Summary of Agent-Based Model Properties | 57
Table 6-1: Number of Samples Derived from Source Data | 77
Table 6-2: Mean and 90% Confidence Interval (CI) Transition Probabilities for an 8 Hour Work Period | 86
Table 8-1: Summary of software design simulation metrics | 119
Table 8-2: Summary of launch vehicle design simulation metrics | 126
Table 9-1: Tally of Face Validation Survey Results | 131
Table 9-2: Validation of Simulations against Published Metrics | 135
CHAPTER 1

INTRODUCTION

1.1 Overview

This dissertation contributes to the body of knowledge in modeling and simulation by developing a simulation of engineering design teams. The dissertation contributes to the systems engineering body of knowledge by enabling the efficiencies of systems engineering processes to be compared through simulation. This dissertation defines efficiency of systems engineering processes as the number of effort hours required to complete the design of a given system. The simulation is built around the Function-Behavior-Structure (FBS) model of designers [1]–[3]. Systems engineering processes historically have not be able to undergo rigorous testing due to the limitations of time and resources needed to test them in a controlled manner. The simulation framework, outlined in this research, provides a mean for simulating systems engineering approaches, providing an economical and practical way to evaluate their efficiency.

Chapter 1 presents motivation for this research in section 1.2, outlining why this research has merit. Section 1.3 outlines the research hypotheses and the steps used to determine if the hypotheses are valid. Section 1.5 provides details on the organization of the dissertation and Section 1.4 outlines contributions that this research provides to the modeling and simulation, and systems engineering communities.
1.2 Motivation

The systems engineering “V” process has been utilized for decades in the systems engineering field. It consists of a logically ordered process to follow when developing a system. The “V” process pairs well with a waterfall approach to project execution since the “V” process is defined as a sequential development process. The “V” process has been shown to provide consistent results when the product being developed is well understood [4].

Many large and complex engineering projects have unprecedented or at least, significantly new designs, have loosely defined stakeholder requirements, and may require new technology [5]. These types of developments can be subject to significant changes over the course of development [6]. The scalability and suitability of the “V” model to large and complex engineered systems is dubious [5]. Some specific issues with the “V” model are addressed in this section.

Development time for certain engineered systems is currently on an unsustainable trend. One example is the trend of cost growth for military aircraft. The cost of these aircraft are increasing at a rate much faster than the growth of the defense budget for the United States [7]. It has been predicted that, given the current course of engineered system development, in the future the entire defense budget of the United States will be required to procure a single aircraft, and, further into the future, the entire gross domestic product of the United States will be required to procure a single aircraft [7]. It is clear that this is an unsustainable course.

The types of engineered systems that are of most interest for this research are known as large-scale complex engineered systems. These systems have significant cost - in the tens of millions to billions of U.S. dollars for a single system. Failures of these systems have significant impact on national defense and economic output. They also have protracted design cycles. Large-scale complex engineered systems can be "one-of-a-kind." “One-of-a-kind” systems often have highly specialized purposes and may not be easily based on other products. Reuse of past components could be limited. Examples of these systems include aircraft, large maritime vessels, and spacecraft. The critical and complex nature of this type of engineered system often drives a need for extensive testing. Given that these systems often operate under a variety of environments and have
numerous requirements, test campaigns often require tremendous time. Unplanned behavior of these systems tends to be discovered late in the development potentially requiring design refactoring.

Large-scale complex engineered systems often operate for decades and thus need to be able to adapt to changing environments and mission needs. These systems also interface with other engineered systems and do so over an extensive operating lifetime. The complexity in large-scale engineered systems makes it difficult to update to changing needs and drives significant operating costs to keep these systems modern [5].

Another characteristic of large-scale complex engineered systems is large and dispersed design teams. These systems require multiple engineering disciplines working on numerous subsystems necessitating the involvement of many individuals and organizations. These design teams may have over 1,000 individuals. The Boeing Company alone employs 153,027 people as of January 1st, 2019 [8]. Geographically separated groups with different cultures, preferences, and processes develop each part of the system [5]. The divergent preferences amongst these individuals can negatively influence the development effort when they become manifested in the design [5]. Figure 1-1 provides a summary of challenges in developing complex systems.

<table>
<thead>
<tr>
<th>Large and complex systems engineering challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Loosely defined requirements</td>
</tr>
<tr>
<td>• Long development times</td>
</tr>
<tr>
<td>• High cost</td>
</tr>
<tr>
<td>• High consequence of failure</td>
</tr>
<tr>
<td>• Unique, cannot be based on other products</td>
</tr>
<tr>
<td>• Wide variety of operating environments</td>
</tr>
<tr>
<td>• Long operating life</td>
</tr>
<tr>
<td>• Interfaces with other systems</td>
</tr>
<tr>
<td>• Large and dispersed design teams</td>
</tr>
</tbody>
</table>

Figure 1-1: Summary of large and complex systems engineering challenges [5], [9], [10]
Developing large-scale complex engineered systems is often filled with unplanned changes, such as changes in requirements, in technology, and in markets. Traditionally, systems engineering is performed on these projects using a sequential approach where each engineering discipline involved in the project each take an input from other disciplines that have completed their work, perform work, and then produce outputs that are used by the next engineering discipline scheduled for work [11]–[14]. Products mature as engineering disciplines perform requirements analysis, design, implementation, verification, and validation of the system following the “V” process. This is often called waterfall planning. This approach is not setup to deal with changes, and when major changes occur, they can cause much of the process to be reset, resulting in major schedule delays and increases in cost to perform development.

Agile methods for product development have become popular in software development [15]. They offer ways for engineers to adapt to changes during development that are not part of waterfall methods [16]. Agile methods tend to focus on involving all major disciplines simultaneously, working together on a single aspect of the overall system, and developing each part of the system in small increments. When changes occur, they are addressed by revisiting affected portions of the design in small increments. The Agile approach is meant to allow changes to be incorporated without upsetting previously completed work. Agile methods may provide benefits for systems engineering of large-scale complex engineered systems as a way to better deal with change that can occur during development.

1.3 Research Hypotheses

This dissertation examines theoretical and simulation evidence that supports the use of Agile methods for complex system development. In this research, two questions are addressed:

1. Do theories exist that support the use of the Agile Scrum framework for systems engineering?
2. What benefits and disadvantages does the Agile Scrum framework have compared to waterfall processes?

The questions limit the scope of the research to the “scrum” Agile method. There are many different Agile methods of varying process and approach [17]. Since scrum is
used about five times more often than other Agile methods, it was chosen as the basis for the research as a representative Agile process [18]. The first question is meant to be answered to determine if Agile Scrum provides a good basis for performing systems engineering. The second question is meant to determine if Agile Scrum has benefits in its use over waterfall methods in systems engineering. To address these questions, the research is broken into three main tasks: examination of theories supporting Agile, development of a systems engineering simulation, and analysis of Agile and waterfall process efficiency in the simulation.

The history of Agile development methods will be reviewed with a focus on ties between Agile development methods and established theories to answer question 1. Systems engineering methods should be based on theory. Without a basis in established theory, one has no certainty that the methods used to develop one type of system will work for another, or that they will work at all.

A novel simulation approach to examine systems engineering methods will be developed. The novel simulation approach uses the FBS model of designers to model the progression of engineers through the design process. FBS transition probabilities are based on data gathered from industry. This agent-based model treats members of an engineering team as agents.

The simulation results will be examined to determine metrics to analyze the performance of the engineers under waterfall and Agile methods. The key metrics to be examined include schedule performance, total effort (effort-hours), and impact of changes. Example system development efforts that will be simulated include a space-launch vehicle and a large software project. The simulation will be validated by confirming that the performance differences between the two simulated methodologies are in agreement with what has been reported by adopters of Agile methodologies in literature. Additional validation will be performed by consulting systems engineering experts to help judge the subjective aspects of the simulation through a survey and basing the underlying model on established theory. The results of the simulation will be used to answer research question 2.

Figure 1-2 provides an overview of the tasks that will be performed to answer the two research questions. Past, relevant work is used in the literature review to answer
research question 1. Existing models are used to develop a simulation of design teams. Validation data from past studies are used to validate the data produced by the simulation and answer research question 2.

1.4 Contributions

The primary contribution of this research is in establishing a Modeling and Simulation (M&S) methodology and framework for the evaluation of systems engineering processes. The work done in this research creates a first-order evaluation method that provides a mean to compare systems engineering process efficiency, not necessarily predict the efficiency. This simulation framework builds on the work of Gero and his FBS model of designers [1]–[3]. This work contributes to the M&S community by establishing an agent-based simulation methodology for the evaluation of systems engineering approaches. Specifically this research:

- Creates an agent-based simulation of designers working on large and complex engineered systems using the FBS model of design.
• Creates a methodology for calibrating FBS models of design with data from real-world large-scale complex engineered system development projects.

This research also contributes to the systems engineering community by researching links between Agile systems engineering and systems engineering theory. It also contributes the systems engineering community by using the agent-based M&S methodology described above to measure the difference in efficiency between waterfall and Agile systems engineering approaches. This provides the community with a means to quantitatively compare systems engineering processes in a cost-effective manner. Specifically, this research:

• Establishes links between Agile Scrum and complex system science. Links are also established between Agile Scrum and decision analysis.

• Provides a way to measure the efficiency of waterfall and Agile Scrum development processes using M&S.

Past research into this topic has focused on process models [19]–[22], where this dissertation is focused on models of designers (agents) and putting these agents through the process as a way to simulate the systems engineering process. This research also expands on models of designers, by mechanizing the FBS model into a simulation, where existing research is focused on establishing the conceptual model [3], [16], [23]. Further detail on the differences between this dissertation and existing research is contained in Chapter 2.

1.5 Organization of Dissertation

Chapter 1 of this dissertation is an introduction, providing an overview of the work, the motivations behind the research, the research hypothesis, and contributions of the research. Chapter 2 is a literature review, presenting information on topics that support the research. Chapter 3 presents the methodology used to answer the research hypotheses. Chapter 4 presents research related to the first research question on the tie that Agile processes have to established theories. Chapter 5 presents the work used to develop an initial proof-of-concept of an agent-based simulation of design teams. Chapter 6 contains the work performed to calibrate the agent-based model with data from a real-world engineering project. Chapter 7 presents the work used to develop the agent-based
simulations of design teams. Chapter 8 contains the results of the agent-based simulations and how the simulations help to answer research question 2. Chapter 9 contains the work performed to verify and validate the simulations used to answer research question 2. Chapter 10 concludes the dissertation with conclusions drawn from the research and future work.

Each chapter begins with an overview to explain to the reader the content within the chapter. The chapters are organized around major topic areas within the research. They are also organized to meet formatting requirements for the dissertation report.
CHAPTER 2

REVIEW OF THE LITERATURE

2.1 Overview

This chapter provides a review of topics related to the subject research. Each section contains a brief description of the topic area and then provides a summary of the current state of research in the topic area. Topics were chosen based on their relation to the research agenda. Figure 2-1 shows what topics relate to what research objectives introduced in section 1.3. Also, in this chapter key terms are identified and defined that are used throughout the dissertation.
Figure 2-1: Map of relations between research objectives and literature review topics
2.2 Agile Processes

Agile processes are being studied so that they can be simulated in an agent-based simulation. Another reason to research Agile processes is to determine the theories that they are based on, if any. This research helps to determine if there are theories supporting the use of Agile processes. Agile processes are also used in this research both to examine if they have benefits or drawbacks when compared to traditional systems engineering. The research into Agile processes supports answering both of the research questions.

Agile development began after the Agile Manifesto, a document outlining the principles behind Agile software development, was published in 2001 [11], [24], [25]. Agile was originally envisioned as a methodology for better software development. The main principle of agility outlined in the Agile manifesto is realized by empowering software developers and relying on their technical excellence to create simple designs to meet customer needs [15]. This agility allows designers to respond to changes in requirements and business needs easily.

Agile evolved in such a way that Agile processes were created based on the knowledge and experience of leaders in the software community, but no theoretical underpinnings were created for many of these early approaches [15]. The most popular research topics on the theoretical foundation of Agile have been complexity science, knowledge management, personality, and organizational learning [15], [24]. It is noted that many Agile studies are not concerned about theoretical underpinning, which leads to a perception that Agile is a-theoretical [15].

2.2.1 Agile Processes: Extreme Programming and Scrum

The most common Agile methods are extreme programming (XP) and Scrum [15]. Extreme programming stresses customer satisfaction and incremental delivery of software features [26]. Scrum also emphasizes incremental feature delivery and is designed to be flexible so that customers can change their minds mid-development without upsetting the development effort. Scrum uses “sprints” which are short, focused efforts to develop a particular feature [14]. These sprints are driven by a work backlog that is ordered by priority, so the most important items are worked on first. Another Agile approach, known as Kanban, focuses on moving work through a defined process as quickly as possible [27]. Kanban divides the process into small tasks that can be focused
on individually [27]. Since scrum is used about 5 times more often than other Agile methods, it was chosen as the basis for the research as a representative Agile process [18].

Scrum is based in complexity science [24], [28]. Major tenets of complexity science such as self-organization are implemented in Scrum. Scrum teams are allowed to form themselves. Daily meetings are designed to accelerate this process and force self-organization [24]. Emergence, a trait of complex system science where systems begin to behave as more than just the sum of their parts, has been claimed to occur in teams practicing Scrum [24]. It manifests itself as high productivity of the design team, where they are performing better than each individual could on their own [28].

Many traditional approaches to development assume that the development process is fully defined and thus can be planned with detail and certainty. It is often the case that these processes are, in fact, not fully defined, but the methods we use to execute them treat them like they are[11]. Unpredictable results are expected from this type of process execution. A waterfall process such as the “V” process does not define how to react to unexpected output from any of the intermediate processes[24]. Scrum treats the development process as a "black-box" or empirical process. It makes no assumptions about how the process will play out and remains flexible to accomplish what may be needed in the future. This matches the Agile manifesto’s principle of valuing responsiveness to change over following a detailed plan [25].

2.2.2 Agile Processes for Systems Engineering

Agile approaches have also been used for systems engineering on projects that also include hardware [29]. The Scaled Agile Framework (SAFe™) is one example of an Agile approach to systems engineering [27]. This framework uses aspects of Scrum, XP, and Kanban to create an Agile approach for an entire enterprise [27]. There are a total of 3 or 4 levels in the SAFe approach, and the lowest two: team and program, are of most interest for performing Agile systems engineering, as these levels are where products are developed. Higher levels are concerned with organizational management. At the team level, individual product development teams are allowed to choose different Agile approaches depending on what best fits their work. At the program level, teams are synchronized using a scrum of scrum approach [27]. Scrum of scrum uses representatives from lower level teams that hold meetings to synchronize and manage work between
teams as the work merges into a single system [24]. In the SAFe approach, scrums at the program level end in demonstrations of the completed system [27]. SAFe has documented benefits such as a 20-50% increase in productivity, a 30-70% faster time to market, and a 50%+ reduction in defects [27].

2.2.3 Believed Benefits of Agile Approaches

Agile approaches have been used on complex system development and are viewed as a possible way to increase productivity [30]. This is typically measured as reduction in development time. Studies of Agile process adoption for software development have shown productivity gains around 36% to 42% in most cases with some teams reporting project failure and others reporting a six-fold increase in productivity [24], [31].

A major focus of this dissertation is examining how Agile approaches can be used for systems engineering. The theoretical basis for Agile as a systems engineering approach is examined in the dissertation research. Agile processes have not been rigorously related to theory. A major focus of the work in this dissertation is to start relating Agile processes to theory.

2.3 Complex System Science

Complex system science is researched in this dissertation to examine its basis in theory and how those theories tie to Agile processes. This is being done to examine the theoretical basis for an Agile process to assist in answering research question 1.

Complex system science developed from the need to understand unexpected behaviors in certain systems. These systems were of a type that consisted of many components [32]. The unexpected behaviors have been found to relate to the way components of a system interact with one another [32], [33]. These interactions create behaviors of the system which cannot be explained from the behavior of individual parts of the system. This is known as emergent behavior.

Emergence is a key trait of a complex system [32], [34]. Complexity science attempts to find simple causes for these complex effects [35]. Rules, based on these simple causes, are used to determine how parts of a system will interact together, then phenomena such as feedback and learning can be used to demonstrate the overall behavior for a system [34]. These rules, when applied to a population of system components, through an analysis, can simulate the emergent behavior.
An example of emergent behavior is initial conditions. Emergence can result when small changes in initial conditions produce unexpected results. Equation 2-1 gives an example of this from chaos theory [35].

\[ y_{t+1} = r y_t (1 - y_t) \]

**Equation 2-1**

In Equation 2-1, \( y \) is the response variable and \( r \) is an initial condition. Figure 2-2 below illustrates how the response can vary wildly with a small change in the initial condition.

**Figure 2-2: Illustration of complex behavior**

Complex system science has its roots in game theory, nonlinear dynamics, and systems theory [34]. Complex behavior, originally thought to be the result of incomplete information, has been shown to be based on the laws of physics, which provides a firm basis for modeling and understanding complex behavior [32].

Complex system science is used in this research to show how the characteristics of Scrum defined by Takeuchi and Nonaka [12] are analogous to the aspects of complex
system science. This is part of relating aspects of Agile processes to theory, an existing gap in Agile research. This helps provide information to assist with answering research question 1.

2.4 Decision Analysis

Decision analysis is being examined for its possible ties to Agile processes, addressing, in part, research question 1.

Since much of engineering design requires making decisions, it is critical to understand the elements of decision-making. We are specifically interested in normative decision theory, or the theory of how people should make decisions [36].

Utility theory is the mathematical basis for normative decision theory. Utility is defined as "the desirability of preference that individuals or societies have for a given outcome [37]." A utility function, the mathematical representation of utility, captures the value and risk preferences of an individual.

Several others have added to Utility theory, most notably, John von Neumann and Oskar Morgenstern. They expanded the theory to include several axioms of rational behavior and examined ways to maximize expected utility based on rational behavior. The axioms of rational behavior include ordering of alternatives, reduction of compound lotteries, continuity, substitutability, transitivity, and monotonicity [36].

Utility theory has been used in value-based engineering to help determine how stakeholders will value certain attributes of a design [38]. This ultimately forecasts how much utility a certain design will bring, whether it is in revenue to a company or in mission success to a military customer [39]. Expected utility, the average utility when uncertainty is accounted for, can be a much more direct measure of the success of a system than a specification of requirements and thus can help designers pick a design that will more readily meet the needs of the stakeholders [40].

Decision analysis is being used in this research to determine if characteristics of Agile processes [12] have a relationship to decision analysis. These types of relationships are useful in determining if Agile processes are based on theory. The relationship between Agile processes and theory is an existing research gap that this dissertation is focusing on. This will aid in answering research question 1.
2.5 Agent-Based Modeling

Agent-based models and simulations are considered a key part of this research since they are used in simulations of waterfall and Agile design teams to answer research question 2. Due to the complex nature of research question 2, simulation is a favored approach as real-life testing of the question is not considered feasible. Simulation is commonly used to examine complex concepts like the ones in this research [41]–[43].

Agent-based modeling utilizes central elements within the model, known as agents, to represent different entities of interest [42]. The model may examine how agents interact with one another and/or their environment. Agents typically have the following attributes [43]:

- A single discrete entity with decision making ability
- Contained in an environment with other agents
- Directed to meet a goal, but not necessarily to maximize an objective
- Able to function autonomously in its environment
- Has the ability to change and adapt based on its experience

Since attributes above align well with designers and design teams, it was determined that an agent-based approach to design team modeling was the preferred method in this research. Agent-based models also are useful in modeling complex phenomena [43], which makes them well suited to modeling large design teams as the work of these teams can create complex behavior [36].

Agent-based models require a reasonable level of abstraction to credibly model most problems [43]. The FBS model of design, described in 2.5, is expected to provide a reasonable level of abstraction for designers that allows an agent-based modeling approach.

Agent-based models are used in this research to represent engineers as they perform work as part of design teams using either Agile or waterfall methods. The use of agent-based models to simulate design teams using different processes is an existing research gap that this dissertation is addressing. The agent-based models will be simulated to examine the differences between the two approaches.
2.6 Models of Designers

Models of designers are being researched to find a model that can be used as an agent in an agent-based simulation of a design team. In the simulation methodology, the designers are the agents and thus, a model of them is critical to the simulation effort as it is the basis for the simulation. This research will support research question 2.

Many different models of how to design have been proposed. Examples include models of “negotiating” between members of a design team [44], coordination of design variables through an optimizing algorithm [45], models of organizations and how they process data [19], [20], and a model of state transitions [23]. For this dissertation, the ability to simulate a complex engineering team in a reasonably simple way was required. The ability to capture design as a probabilistic process was also desired so variations in the process could be understood. For these reasons, the Function-Behavior-Structure (FBS) model of design was chosen.

2.6.1 FBS Model of Design

Early research into the process of design started in the 1960s and 1970s. One shortcoming of this research is that it lacked a common ontology, thus it is difficult to understand precisely what each of the researchers found as the term "designing" was used to describe many different activities. To solve this issue, the FBS ontology of design was created [46]. It consists of several function, behavior, and structure states that designers move between depending on what part of designing they are performing [23]. This ontology is independent of what is being designed, making it easily adaptable to many different domains.

The FBS structure began evolving in the 1980's, but initial efforts conflated aspects of behavior and function [47]. This was later corrected and the FBS structure was first presented in 1987 [1]. The FBS structure consists of 3 main constructs, function - determining what the design is for, behavior - what the design does, and structure - what the design is [47]. There are connections between these items because of causality. Function leads to behavior and behavior leads to structure. There is no direct link between function and structure. Behavior is divided between expected behavior (Be) and the behavior of the structure (Bs). Be indicates what the expected behavior is, which comes from functions. Bs indicates the actual behavior of the design, which is derived
from structure. Other items have also been added to the ontology. Requirements (R) have been added as a subset of Function, and Description (D) is the final activity in design where the structure is documented. There are also 3 types of reformulations from the structure state. These either modify the structure, the expected behavior, or the function [23], [46], [48].

There are eight state transitions involved in the FBS framework as seen in Figure 2-3 below. The formulation transition translates requirements to functions and functions to expected behavior. The synthesis transition changes expected behavior to structure. The analysis transition changes structure to the state “behavior from structure”. Evaluation checks behavior from structure against expected behavior. The documentation transition goes from structure to documentation. Type 1 reformulation goes from structure to a modified structure. Type 2 reformulation goes from structure to expected behavior. Finally, type 3 reformulation goes from structure to function [2].

![Figure 2-3: The Function-Behavior-Structure Model](image)
2.6.2 The Situated FBS Framework

The situated FBS framework was developed in 2000 to extend the FBS framework to account for interactions between 3 worlds that the designer and the design exist in [49]. These are the external world - things outside of the designer, the interpreted world - the designer's interpretation of the external world, and the expected world - what the designer expects his or her actions to do. These worlds are related to one another through interpretation - transform from the expected world to the interpreted world, focusing - taking information from the interpreted world to the expected world, and action - using the expected world to bring about change to the external world [46], [50].

In the situated framework, each of the major areas of the FBS model are further elaborated to describe how the model operates across the 3 worlds. The original FBS model is expanded to account for a cognitive processes across the 3 worlds [50].

2.6.3 FBS Validation

Verbal protocol analysis has been used to empirically study and validate the FBS ontology [3], [23], [51]–[53]. Verbal messages between designers and different engineering disciplines are used to analyze if the FBS structure appropriately captures their thinking. These studies showed that the FBS ontology is a robust method for capturing the work of designers [3], [23], [51]–[53], demonstrating the usefulness of the FBS ontology, and its validity as a model of designers. The empirical studies were performed by using engineers and designers in short design sessions rather than over the course of a true development effort. Due to this, the studies indicated transitions between FBS states that do not exist in the FBS model. This is viewed as an artifact of the validation approach. Actual validation on a design effort is difficult due to the timespans of these efforts. The validation studies did prove that designer actions can be grouped into the FBS states, but in the short-term studies, the state transitions can break from the model.

FBS is noted as being limited since it is abstracted to such a high level. Other criticisms include changing definitions of the FBS states, an overall unclear purpose, and an implied level of detail in state definitions that isn’t supported by the high level nature of the model [46].
Overall, FBS offers a useful technique for understanding design at a high level, which was determined appropriate for agent-based modeling of design teams for the purpose of examining research question 2. FBS was selected as the model of the design process for this research due to its top-level nature, its basis in empirical data, and the ability to tailor it to individual design efforts without reformulating the model. The situated framework was not chosen since it does not have as much validation supporting it. The situated framework has significantly more transitions which would present a more complex approach to calibrating transition probabilities from real-world data. The situated FBS framework may be of use after it is further matured as discussed in 10.3.2.

2.7 Design synthesis process

The processes that create a design are being studied in this dissertation in support of research question 2. These design synthesis processes involve many engineers representing many disciplines each performing their tasks to break down the system into manageable pieces that then can be designed and analyzed [54], [55]. Design synthesis processes are mechanized into the computer simulation used to answer research question 2, thus an understanding of them is critical to properly representing the work of a design team.

The process by which a design is created can be difficult to fully define. Guidance exists in standards such as ISO/IEC/IEEE15288 and MIL-STD-499, which provide generic processes to be tailored to the needs of each individual design effort [56], [57]. Attempts to rigorously define the design process have been made, but have been found to lack the flexibility to adequately address all possible design situations [58]. Hazelrigg defined a rational design synthesis process, but admits that it is not well suited to large and complex systems due to the need to involve large numbers of designers [36].

2.7.1 Systems Engineering “V”

Systems engineering has traditionally been defined using the “V” model [59]. This model defines a hierarchical process where a concept of operations is decomposed into requirements. Requirements are then further decomposed in a hierarchical manner until small, manageable pieces are created, often along functional or disciplinary boundaries [5]. The design is then implemented into hardware and software products. The decomposed system is then re-composed through integration and test. The process ends
with a validation that the engineered system meets its intended purpose. For certain engineered systems, there can be a large number of the manageable pieces of the system.

An illustration of the “V” model is given in Figure 2-4. There are some inherent assumptions to using the “V” model. One is that the sum of the parts equals the whole [5]. For certain engineered systems development, the number of parts may be immense, and functionality may be spread amongst many different parts. Changes to one part of the system can affect many others and can do so in ways that are not obvious. This leads to a risk of the system not performing as intended when parts of it are developed in isolation from one another. These issues tend to manifest during the latter part of “V” process when the system is integrated together and tested. This tends to lengthen the integration phase. Evidence of this issue on software programs was found by a U.S. Defense Science Board study, which concluded that 90% of software programs did not perform to budgets and schedule constraints mainly due to unplanned time in integration [60], [61].

Another major assumption in the “V” model is that stakeholders can accurately describe their preferences in requirements documents. Stakeholders are people who are affected by the system design and include users, maintainers, purchasers, support
organizations, as well as other affected organizations. Requirements state what is not wanted in the design and do not state the preference of the stakeholder [40], [62]. This leads to limitations on the design space that can rule out superior designs. When requirements fail to communicate the preference of the stakeholder, designs that do not fulfill the stakeholder’s needs result since these preferences were not communicated to the designers. This can lead to the need for expensive rework of a design, the design being unsuccessful in the market, and dissatisfaction of customers with the developer [9], [36].

A further assumption of the “V” model is that the products from each step in the process contain no defects [11]. Typical “V” models contain no inherent mechanisms to deal with defects and issues.

Engineered systems development can also lead to the proliferation of requirements. As the system is decomposed, more and more requirements are generated at each level of the system hierarchy. The more parts to a system, the more resulting requirements. A study of software projects found that the effort (in effort-hours) required to complete an engineering project is a function of the number of requirements cubed [63]. The uncertainty associated with the required effort also increased as a cube of the number of requirements [63]. This data demonstrates that the proliferation of requirements has a highly nonlinear effect on the effort required to complete an engineering project. Hazelrigg also demonstrated that the act of decomposing requirements, a central theme to the “V” model, lowers the probability of meeting the requirements [36]. The “V” model also reserves system validation and verification efforts to the end of the design cycle. Doing this can cause issues with scaling the process to large systems since the opportunity to catch mistakes that may change the design are done very late in the design process. Issues with the system meeting its intended purpose are not discovered until late in the design cycle, potentially requiring significant rework. The systems engineering “V” process is often implemented as a waterfall process with each of the steps in the process identified in Figure 2-4 (concept of operations, system requirements, etc.) performed in a serial manner. Waterfall processes are described below.
2.7.2 Waterfall

One type of design synthesis process is waterfall. The waterfall process consists of developing an engineering solution in a series of successive steps. These steps can align with the steps in the systems engineering “V” model or the steps in ISO/IEC/IEEE15288. The waterfall method was first formally described by Winston Royce [13]. In his description, Royce mentions that a straight waterfall implementation is “risky and invites failure” [13]. He describes the need to plan for adequate documentation and a prototype as means of avoiding failure in use of the waterfall process. The waterfall model is known to provide good results when requirements are well understood and the work to be accomplished is very similar to an existing product [4]. This is due to waterfall processes assuming perfect information about how the project will execute [11]. This assumption is closest to being valid when the work is well understood, which is not typical of many large system developments.

Waterfall project planning has fallen out of favor with many software development projects [24]. Despite this, it remains a mainstay of many large and complex system development efforts [64]. Using waterfall on these large and complex programs at the beginning of a project requires advanced detailed planning. On large, multi-year development projects, it may not be feasible to predict task plans accurately at the beginning of the project and the issues with waterfall suggested by Royce [13] can manifest. Most of these programs result in not meeting technical, cost, or schedule requirements [9].

2.7.3 Agile

Agile processes, described in section 2.2, are another means of performing design. While originally focused on software design and development, their use on a wider variety of systems has shown to be feasible. Specifically, the Scaled Agile Framework (SAFe) has been created to provide system designers with Agile methods that can be used for software development, hardware development, and productions situations [27], [65]. SAFe provides a selection of Agile processes like Scrum and Kanban that can be tailored to suit the needs of individual projects [27]. It is built around the need to continually release designs and is structured in a way to support continuous design, development, test, and release [27]. Most case studies on SAFe are for software-only systems [66].
There are some examples where hardware and software systems have successfully used SAFe, but they are not large and complex systems [67].

The Design Analysis Cycle (DAC) approach is an Agile-type approach that has been used on large and complex systems [68]. It is not a typical Agile approach, but embraces certain aspects of Agile, with defined periods of work followed by meetings to reflect, learn, and plan the next period of work. It may be thought of as a hybrid approach that uses some aspects of Agile, while still having relatively short periods of waterfall work.

Waterfall and Agile design synthesis processes were researched to gain an understanding of how they can be simulated for the agent-based simulation used to answer research question 2. Both processes are situation-dependent, with Agile being quite vague in how it should be executed. Published examples of design processes are used to develop the simulation logic and execution flow [24], [56].

2.8 Model-Based Systems Engineering

Research into Model-Based Systems Engineering (MBSE) is used in answering both research questions. For question 1, MBSE research was used to support the ability to perform Agile systems engineering on a wide variety of systems, not just those that can be rapidly reconfigured. For the simulation used in question 2, MBSE was assumed to be part of the design process. This is viewed as a simplifying assumption as the agents will have fewer interfaces between them, communicating through models, than if agent-to-agent communication was modeled.

The International Council on Systems Engineering (INCOSE) defines MBSE as “the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. MBSE is part of a long-term trend toward model-centric approaches adopted by other engineering disciplines, including mechanical, electrical, and software. In particular, MBSE is expected to replace the document-centric approach that has been practiced by systems engineers in the past and to influence the future practice of systems engineering by being fully integrated into the definition of systems engineering processes [69].”
Traditional systems engineering practices rely heavily on documents such as specifications, design descriptions, drawings, plans, and interface documents to record and communicate characteristics of a system between members of the development team and to stakeholders. This practice results in a chronic problem of document maintenance during a development effort. Each design change may affect multiple documents and multiple stakeholders of the system. One task of system engineers is to keep this documentation up to date and ensure that each stakeholder in the design is aware of changes. This can prove daunting and creates risk that the systems engineer may miss something. Things that are missed often are found late in the program during the integration and test phase, when the cost to correct the problem is much higher [70].

The intent of MBSE is to facilitate traditional systems engineering activities through a more efficient means. A system model is created where engineers, stakeholders, and other interested parties can view and develop the system. The system model can present tailored views of the system to stakeholders while maintaining the full complexity of the system within the model itself. This single interface allows design decisions and changes to be quickly understood by affected parties and their impacts to be assessed[71]. Models also grant the ability to perform virtual validation and verification, lowering the risk of finding an issue during system validation and verification.

Current MBSE practice utilizes models, developed through graphical languages, to record details associated with the requirements, design, analysis, verification, and validation of a system. These models create representations of the behavior, architecture, and requirements of a system. This approach is centered on the System Modeling Language (SysML), which is emerging as a dominant language for MBSE, although other languages and approaches exist [72]. SysML is based on the Unified Modeling Language (UML), which is a general-purpose software modeling language [73].

The INCOSE MBSE roadmap [74] outlines the overall vision for systems engineering in 2020. Per the roadmap, MBSE standards begin to emerge in the 2010 timeframe, and formal MBSE theory begins to be defined closer to 2020. This approach could be considered backwards from a traditional approach. Rather than waiting for an established theory for MBSE to emerge and start practicing MBSE per that theory,
MBSE has started in an ad-hoc fashion across the engineering industry with a variety of different methods and uses for MBSE being explored throughout industry [75]. MBSE is thus being experimented with to determine how it may work without a theoretical basis for how it should be performed. While this approach allows rapid use of MBSE throughout industry, it creates the risk that emerging MBSE methods may not actually be producing the results that it advertises and may not lead to better products. This in turn may turn individuals and organizations away from MBSE.

MBSE is seen as a key enabler of Agile processes for systems engineering [11]. Engineering with models is much akin to software development. SysML has evolved from a software modeling language used for software development [76]. Both software development and engineering with models allow rapid examination of design concepts and the ability to constantly validate the design.

MBSE is used in this dissertation as a way to enable an Agile approach to systems engineering. It is also seen as a simplifying assumption for the agent-based simulation as agents are given information to make decisions as soon as it is available rather than modeling the complexity of communication between the agents. The integration of MBSE with Agile is a recent area of research and this dissertation is working to contribute to that body of knowledge.

2.9 Value-Based Engineering

Value-Based Engineering (VBE) is used in this research as part of the agent-based simulation. The design agents require a goal to work towards, and VBE concepts are used to establish those goals. This research supports research question 2.

VBE arose from the need to find a new basis for performing systems engineering over requirements-based methods. Expensive failures of large programs provided motivation to find a new approach that would result in better systems engineering performance [40]. VBE is enabled by modeling the attributes of a system. Surrogate or meta-models greatly enhance VBE since they are usually continuously differentiable (meaning gradient-based optimizers are able to operate on them) and they greatly reduce computational load. VBE works to analyze the attributes of the system through these models and understand how they drive the overall value of the design [40].
VBE creates a framework that allows systems engineering to specify the system in terms of its extensive attributes and its components. Examples of extensive attributes are performance attributes: weight, reliability, safety, etc. VBE does not apply requirements to these attributes, which is the traditional practice of systems engineering. Instead, a value function (an objective function with a meaningful, single unit of measure) is created that converts the full set of system attributes to a measure. The design team's goal is to maximize this measure while meeting requirements for non-extensive attributes such as software language or color [40].

For this research, design is defined as a purposeful activity to fulfill stated goals or needs. Since humans drive this process, design is essentially a decision-making activity that may be characterized as driven by economic preferences (value maximization) and psychological aspects (human decision-making). For this research, the goal of design is to maximize value, which elicits a value-driven approach. Traditional systems engineering aims to satisfy stakeholder demands which results in a requirements-driven approach [77].

VBE is based on decision analysis as it provides a basis for examining rational decision making under uncertain conditions found in the design process [77]. For VBE to be effective, the value models must be consistent with von Neumann and Morgenstern’s utility theory [38], [78]. The models must result in a scalar metric. In addition, uncertainty about the value metric must be accounted for.

VBE is a rapidly evolving field of research. Current research areas include examining how to perform VBE on a real system [38], how to deal with systems that cannot be optimized for Net Present Value (NPV) to a company [39], [79], addressing risk in decision making [80], examining coupling of design attributes [81], [82], and examining how to use VBE with Multidisciplinary Design Optimization (MDO) [83]. VBE has promise with its basis in utility theory to provide a superior means of performing systems engineering.

This research includes value-based engineering principles in design team simulations. These principles are used to give design teams, using both Agile and waterfall methods, an objective to achieve as a part of their design effort.
2.10 Summary

The areas reviewed to support this research aid in providing a foundation to base the research on. They include areas relating to a variety of theories that support answering research question 1 as well as modeling and simulation principles and guidance that support answering research question 2. This background is key to ensure that the research in this dissertation is both current with research in similar areas and that it agrees with the findings in the modeling and simulation and systems engineering communities. The review also focused on finding research gaps in existing work on these topics. These research gaps were used as a basis for the research questions and objectives in this dissertation. The primary gaps found were:

- Lack of an agent-based simulation of design teams using the FBS model
  - Similar research is based on simulating the design synthesis process only [3], [48], [50]
- Lack of the use of theories to justify the use of Agile processes
  - Agile research is largely based on empirical evidence [17]
CHAPTER 3

3.0 RESEARCH METHODOLOGY

3.1 Overview

The research methodology presented in this chapter draws on the topics presented in chapter 2 to create a list of tasks needed to address the two research questions posed in chapter 1. Objectives are defined that will create the basis for answering these questions. Tasks are developed to meet the objectives.

The research goal of the proposed work is to develop an understanding of how Agile methods can be used in the development of a complex system. This objective, like many systems engineering research questions, is much too broad to solve for all possible systems and can take many years, perhaps decades, to gather real-world data. For this reason, only a subset of the whole problem is being addressed in the proposed research to prove this conjecture.

First, the history and development of Agile methods are examined to ensure that the methods have a basis in theory. This is important to confirm that the methods can be applied across the many types of systems encountered in systems engineering. Without a theoretical underpinning, the methods may not be a viable systems engineering process.

The answer to the first research question is determined through a literature review of relevant topics. The answer to the second research question is determined through development of a novel agent-based simulation of a design team used to examine and analyze both waterfall and Agile processes. Simulation is an established methodology for examining theorems and has extensive use in the academic community [41]–[43], [84].
The proposed simulation is novel in that it uses the FBS model to represent agents in a simulation of design teams. The simulation is also novel in that it is based on a cognitive model of design rather than statistical models of the time needed to complete tasks [19], [20]. Real-world data from an industry design team working on a major element of a complex system is used to calibrate the model. These data consist of probabilities of being able to move from one state of the FBS model to the next after a given amount of time has passed. The development of two types of systems will be performed: a space-launch system, and a large software project. Simulations will be performed using Monte-Carlo techniques and results will be assessed using statistical methods. Statistically significant differences between Agile and waterfall methods will be examined to determine if the Agile method provides any advantage in improving either cost or schedule performance. Data from these simulations will also be used to identify any other characteristics of the Agile process that may be beneficial or a detriment to systems engineering. Both Agile and waterfall simulation will assume the use of MBSE techniques. Some research shows that MBSE is beneficial when performing Agile systems engineering [11], [85].

Figure 3-1 shows the research methodology as a mind map [86]. The two research questions are the primary blocks in blue. Direct decompositions of the questions (shown in black line relationships) result in the tasks needed to answer the questions shown in green. The next level of decomposition of tasks is objectives (shown in dark blue-green) that need to be accomplished to complete the task. The objectives have additional relations in solid blue and pass data between each other with a dashed blue line. These relationships show how research on question 1 informs the work needed to answer research question 2. Specifically, the literature review for question 1 informs the modeling and simulation planning for the proof of concept simulation. For research question 2, the proof of concept simulation refines the initial modeling and simulation plan before the more complex simulations are created. The data from the more complex simulations performed are passed on to an analysis task, which produces results that will answer research question 2.
The following sections present the detailed objectives and tasks associated with answering each research question.

3.2 Research Question 1: Do theories exist that support the use of the Agile Scrum framework for systems engineering?

3.2.1 Objective 1: Investigate the Theoretical Roots of Agile Scrum
This objective is meant to determine the basis for using Scrum as the preferred Agile method for systems engineering. Understanding the history behind Scrum, and how it was developed, will help show whether or not theories exist that support Scrum. This objective is not meant to fully document and determine all theoretical foundations for Scrum, but rather, to show that theories exist that support using scrum for systems engineering. Future work to further define the theories that support scrum will be recommended as well.

Tasks:
1. Perform literature review of papers outlining the history of scrum
2. Identify and analyze papers outlining the theories behind Scrum and theories that may relate to Scrum
3. Perform analysis to relate Scrum to theories not previously identified in literature
4. Summarize the research findings, outlining the theoretical basis for scrum

3.2.2 Objective 2: Investigate How Scrum Accomplishes Systems Engineering Tasks
This objective is meant to determine the potential for using scrum to perform systems engineering development on a complex system. The process within Scrum used for performing system design will be examined and compared to traditional approaches. The main product will be a written summary of findings.

Tasks:
1. Analyze the systems engineering process and determine how the scrum methodology accomplishes systems engineering tasks required by the process or if it does not support performing the systems engineering process.
2. Review literature to identify examples of scrum being used for systems engineering
3. Summarize findings
3.3 **Research Question 2:** What benefits does the Agile Scrum framework have over waterfall processes? In what ways does the Agile Scrum framework fall short of waterfall processes?

3.3.1 **Objective 1: Establish Foundation for Modeling and Simulation Effort**
This objective is accomplished by reviewing literature on relevant topics that will provide information needed to create the model and simulation.

1. Perform a literature review of Agile development processes
2. Perform a literature review of Model-Based Systems Engineering
3. Perform a literature review of decision analysis
4. Perform a literature review of agent-based modeling
5. Perform a literature review of design models
6. Research how design models can be mechanized for waterfall and Agile processes
7. Select a model for use in simulations
8. Develop conceptual design of a design team simulation

3.3.2 **Objective 2: Develop a Simulation of a Simply Supported Beam Design**
This objective uses the FBS model in a simulation of a simply supported beam development. It is meant as a proof-of-concept to ensure that the modeling methodology is sound prior to using this approach on more complex systems. Its product is a simulation that works for both Agile and waterfall approaches.

1. Design simulation
2. Gather FBS model data from literature
3. Code simulation
4. Perform simulation V&V

3.3.3 **Objective 3: Develop a Simulation of a Launch Vehicle Development**
This objective develops a simulation of a launch vehicle development. Its main product is a simulation that works for both Agile and waterfall approaches.

Tasks:
1. Design simulation
2. Gather agent model calibration data from a real-world project
3. Code simulation
4. Perform simulation V&V

3.3.4 Objective 4: Develop a Simulation of a Large Software Product Development
This objective develops a simulation of a large software development project. Its main product is a simulation that works for both Agile and waterfall approaches.
Tasks:
1. Design simulation
2. Gather agent model calibration data from a real-world project
3. Code simulation
4. Perform simulation V&V

3.3.5 Objective 5: Analyze Simulation Data
The purpose of this objective is to analyze the simulation data created in the three simulations above. Its main product is the results of analyzing the simulation output data.
Tasks:
1. Perform a literature review of appropriate data analysis techniques including classical statistics and robust statistics.
2. Analyze simulation data to find significant differences between Agile and waterfall approaches, specifically in total time spent and resources expended

3.3.6 Objective 6: Develop Rationale for Significant Differences in Waterfall and Agile Simulation Results
For this objective, the results from the simulation data will be used to postulate reasons for significant differences.
Tasks:
1. Gather all significant differences from 3.3.5
2. Perform a causality analysis for each of the items

3.3.7 Objective 7: Examine the Results to Determine What Significant Difference Exist between Agile and Waterfall Approaches
This objective will develop the final rationale stating the differences between Agile and waterfall approach based on analysis of the simulation data.
Tasks:
1. Develop final rationale stating differences between Agile and waterfall methodologies
2. Document summary of findings for research question 2

3.4 Summary

The two research questions are being addressed through two different methodologies. First, a detailed literature review and comparisons of the findings are being used as the primary method to address research question 1. Second, simulation is being used as the methodology to address research question 2. Each question has been broken down into objectives with objectives further broken down into tasks. Completion of the tasks is meant to create the data needed to answer the research questions.
CHAPTER 4

4.1 THEORETICAL ROOTS OF AGILE SCRUM

4.1 Overview

Chapter 4 focuses on answering research question 1. Chapter 4 investigates the history of Scrum, examines how the attributes of Scrum have allegorical ties to theory, and examines how Scrum can be used for systems engineering. The history of Scrum is important to understand as it reveals how the Scrum process was formulated and identifies theories that were referenced in its development. The links between Scrum and theory are identified and analyzed in section 4.4 per question 1, objective 1. The use of Scrum for systems engineering is also explored to demonstrate the feasibility of using Scrum for non-software projects in section 4.5 for question 1, objective 2. An overview of the research into the theoretical roots of Agile Scrum is shown in Figure 4-1.
Agile methods have become popular for developing software and have begun to be advocated as being applicable for systems engineering [11]. Agile methods tend to be based on the principles of the Agile manifesto, including welcoming change, delivering frequent working product iterations, and allowing teams to self-organize [25]. While Agile methods have proven successful in software development their effectiveness in other disciplines is unclear.

4.2 Methodology

This chapter examines the relationships Scrum has with established theories and examines if Scrum could be used for systems engineering. A literature review of Scrum and a comparison of Scrum to theories and systems engineering processes are used as the methods for research. This research is performed to verify that Scrum ties to theories that would support its use for systems engineering and that the systems engineering process could be accomplished with Scrum.

4.2.1 Literature Review of Scrum

A literature review researching the characteristics of Scrum is performed. This review determined the fundamental definitions for the steps and characteristics of the
Scrum process. This review focused on articles that detailed the history of Scrum and those that define the Scrum process. The literature focused on journals related to software development, business management, and engineering management. Keywords searched for included Agile, Scrum, and the names of the founders of Scrum. References on the development of the Scrum process were used to trace back to original source material used to define Agile processes and Scrum. From this original source material, theories used to justify aspects of Scrum were identified.

4.2.2 Literature Review of Theories Related to Scrum

A literature review is performed on theories identified in the previous literature review which were used or alluded to in the development of Scrum. Articles are reviewed to find the axioms of the theories and the fundamental definitions of the axioms. The literature review focused on journals related to complex system science, complexity, and systems engineering. Keywords used to search for the articles included complexity, complex system science, decision analysis, and normative decision theory.

4.2.3 Relationships between Scrum and Complex System Science

Principles from Scrum and complex system science were compared to see if they align as suggested in the Scrum literature. This examination uses fundamental definitions of the axioms of complex system science and fundamental definitions of the characteristics of Scrum. Definitions were determined from the literature review mentioned above. The two sets of definitions were compared to determine similarity in the content. Substantial similarity in content is determined by examining the characteristics of Scrum and seeing if the characteristics, as described, use processes, descriptions, or definitions that are close to those of the axioms. If the definitions were assessed as being substantially similar to one another, then a relationship between the two were identified. A discussion on the relationships, describing the similarities, is then given.

Theories from complex system science, not explicitly referenced in the development of Scrum, were also examined. These theories provide additional axioms of complex system science that could be related to Scrum that are not mentioned in the Scrum literature. Using the same methodology described above, the fundamental
definitions of these axioms are compared to the fundamental definitions of the characteristics of Scrum to see if there is substantial similarity between the two.

4.2.4 Relationships between Scrum and Decision Analysis

Finally, an analysis of theories and principles in decision analysis is conducted. Decision analysis is explored due to the relationship between decision-making and systems engineering, as described in Chapter 2. A literature review of decision analysis is performed per the literature review of theories related to Scrum mentioned in 4.2.2. The fundamental definition of decision analysis axioms and Scrum characteristics, taken from the literature reviews mentioned in 4.2.2, are compared for similarities. This content comparison is the same process described above for complex system science, but for decision analysis. If the definitions were substantially similar to one another, it was determined that there was a link between the two.

4.2.5 Scrum in Systems Engineering

A literature review of current industry standards for systems engineering and an examination of how the standards could be utilized in Scrum is performed. The examination consisted of determining the fundamental definition of the steps in the systems engineering process through a literature review of systems engineering standards and using the definitions of the steps in the Scrum process taken from the literature review above. The two sets of definitions were then compared to find if there was compatibility or incompatibility between the two processes based on the fundamental description of the steps of the process. Compatibility is determined if the definition of a process step from the systems engineering process would allow the fundamental work of that step to be accomplished in a new way, by a step in the Scrum process. Incompatibility is determined if the systems engineering process, as defined by leading standards, require a step that could not be accomplished with in Scrum.

4.3 History of Scrum

Takeuchi and Nonaka described the first concept of what would later become known as Scrum in “The New New Product Development Game”[12]. This original description of the Scrum approach was a general approach for any kind of development work, and not exclusive to software. Takeuchi and Nonaka contrast a traditional development approach as a relay race against a new development approach termed
"rugby.” Rugby, the game, evolved from English football, known as soccer in the United States, when, per a possibly incorrect tradition, William Webb Ellis, a 17-year-old playing soccer, made a defiant move to attempt to win a game. With little time left in a match and his team behind, Ellis picked up the ball and ran towards the opposing team’s goal. This started a new sport, rugby, first codified in 1839 [87].

Takeuchi and Nonaka describe a traditional development approach as a relay race where each discipline in an organization takes their turn, in serial, to develop a product [12]. In the rugby method, a "scrum" of disciplines move the ball downfield all working simultaneously to achieve a common goal [24]. Takeuchi and Nonaka interviewed six design teams and leaders from multinational corporations to learn how they implemented processes to develop breakthrough products [12]. Six characteristics of these teams were identified and form the basis of scrum: built-in instability, self-organizing project teams, overlapping development phases, multilearning, subtle control, and an organizational transfer of learning [12].

Built-in instability comes from top-level management starting the project without detailed work plans. Instead, they create challenging goals which also grants the team great freedom [88]. Teams become self-organizing when they are driven to a state referred to as "zero information" [12], where prior knowledge does not apply. Left on its own, the process begins to take on its own order as needed by the team. Overlapping development phases are used to integrate many of the different points in the development phases into a holistic approach. A serial approach creates bottlenecks, which controls risk, but also prohibits integration. Overlapping the phases increases flexibility and development speed [88], [89]. Multilearning is defined as learning across multiple domains and across multiple levels, e.g. individual, group, corporation, etc. [12]. Takeuchi and Nonaka [12] provide examples of learning across multiple levels where groups were put into difficult circumstances, given new environments, or use a new methodology. Subtle control is a construct defined as creating enough checkpoints to prevent teams from becoming unstable, meaning to lose control and have their work turn into chaos. Transfer of learning is the final characteristic of the rugby development process proposed by Takeuchi and Nonaka [12] when organizations take lessons learned and apply them throughout the company.
Limitations of the approach are noted by Takeuchi and Nonaka [12]:

- It may not apply to all situations
- It may require extraordinary effort from team members, requiring long hours during peak development times
- It may not apply when a revolutionary breakthrough is needed.
- It may not work where lots of face to face interaction is not feasible.

Takeuchi and Nonaka argue that this new approach can produce constant innovation that is needed in a world of constant change [12]. Basically, the nearly chaotic conditions of technology evolution, market changes, and emerging needs are creating the need for a development process that can accommodate them. The idea that requirements can be set, technology can be chosen, and a development effort executed over the course of years is not feasible. Technology and stakeholder needs change too rapidly and more agility is needed. These conditions create the need for Agile processes like Scrum.

The overview of the history of Scrum shows that it was described based on the needs to quickly develop systems and its characteristics have some relationship with theories. These relationships will be further analyzed in the subsequent section.

4.4 Links between the Attributes of Scrum and Established Theories

In this section, we analyze the relationship Scrum has with established theories. This analysis is performed to verify that Scrum ties to theories that would support its use for systems engineering. This is being performed in support of research question 1, objective 1. First, a comprehensive literature review is performed to find all theories used in the development of Scrum. Next, theories from complex system science not explicitly used in the development of Scrum are analysed. Finally, an analysis of theories and principles in Decision Analysis is conducted. The analyses present the relationships Scrum has with theories and how they relate to the ability to use Scrum for Agile systems engineering.

4.4.1 Previously Identified Relationships between Scrum and Complex System Science

The rugby process described by Takeuchi and Nonaka [12] was codified by Ken Schwaber and Jeff Sutherland into a process they called Scrum [14]. When Scrum was
formalized into a process, a relationship to complexity theory was made [24]. Specifically, the work of Chip Langton [90] was used by Ken Schwaber to illustrate how complex system science supports development teams following an evolutionary approach [24]. These include relationships to chaos theory and the self-organization principle of complex system science.

The closer the team can operate to the “edge of chaos” while still maintaining overall control, the more flexibility they will have to create the best solution [28]. This flexibility is meant to increase the probability of success by allowing teams to be flexible in responding to environmental changes such as changes in key personnel, changes in tools, changes in methods, changes in requirements, or other external influences [88]. Therefore, Scrum advocates for overlapping development phases [89].

Scrum directs that teams be multidisciplinary and be allowed to determine the role of each team member themselves. By directing this, Scrum has self-organizing project teams that are setup for multilearning. The self-organization principle of complex system science were used to justify this aspect of Scrum [24].

Transfer of learning is implemented in Scrum through the multidisciplinary team that is allowed to organize itself. This has a relationship to complexity theory in that simple rules are used to control an organization to help drive it to self-organize [33]. These rules may need to change over time with new ones being learned and old ones being unlearned to maximize the efficiency of an organization. The relationships identified to theories are metaphorical relationships, not mathematical proofs of the use of theory. These relationships are based on similarities in logic and similar application of principles. Given this, we say that Scrum is loosely based on theory.

### 4.4.2 Newly-Identified Relationships between Scrum and Complex System Science

More relationships between Scrum and complex system science exist than have been previously identified. These relationships are important to establish as they provide the start of a theoretical basis for Scrum, albeit a weak and loose one. The relationships established in this dissertation relate to the Scrum characteristics of built-in-instability and subtle control.

The built-in-instability concept from complex system science is executed in Scrum through the lack of a detailed execution plan. The plan is purposefully left vague
by engineering management so that the development team has more flexibility to
determine what needs to be done to meet the goals set by management. This mimics the
emergence trait in complexity science where the whole becomes more than the sum of its
parts [32]. This also follows established principles of complexity science where allowing
some amount of chaos results in complex and useful behavior [33].

Industrial process control techniques were added to Scrum in order to control the
“black-box” development process [24], [85], [91]. This is a means of achieving subtle
control. Schwaber argues that development is inherently a “black-box,” meaning that the
process is not definable prior to executing it. This assumption has major implications for
engineering management as it means the process that is being managed is not definable a
priori. Unpredictable events like requirements changes, technology change, and scope
creep often plague system development efforts [5]. If the system development process is
treated as a fully-defined process, but in fact, is not, unpredictable behavior can result
[11]. A full explanation of this concept is described below to expand on previous work
[85], [91]. An illustration of this issue is provided by applying a controller to a system as
shown in Figure 4-2. The controller uses a model (plant) of how the system performs and
reacts to control inputs. If the model is incorrect then unpredictable and chaotic system
behavior can result. This is an allegory for controlling a development with a mismatching
plan and having the development produce undesirable results.

To illustrate the control model to system mismatch issue, a simple Proportional –
Integral - Derivative (PID) controller and process was generated using a template
available at Engineers-Excel.com [92]. The control system and plant (process) are shown
in Figure 4-2. The process is meant to be controlled to output a process value that
matches a set point value input into the controller. Control of the process is accomplished
by the controller. When simulated with the proper control model, the system responds
appropriately by quickly bringing the process value to the set point as seen in Figure 4-3.
If the process changes by modifying the gain, delay, and time constant parameters, but
the control model remains unchanged, chaotic behavior ensues as seen in Figure 4-4. This
simulates the danger of assuming a control model for a given system when, in fact, the
system is not well understood.
Figure 4-2: Simple Process with PID controller

Figure 4-3: Process behavior with matched process and control model

Figure 4-4: Process behavior with mismatched process and control model
Backlogs, create the subtle control in Scrum, and are used as a control system to manage the chaos of development. In the allegory to a PID controller above, this would be represented as changing the plant to obtain the level of control desired. Engineering managers use the backlog to control the process by listing tasks that need to be performed in priority order to direct the development team on what they should do. These tasks can include correcting errors, addressing new requirements, adding in new functionality, and upgrading technology. In this way, engineering managers use the backlog to control each sprint by keeping the team focused on short-term goals while avoiding scope changes. In this way, Scrum utilizes control theory in its implementation. Another form of subtle control in Scrum are sprint cycle durations. Keeping sprint cycles to a short amount of time allows for maximum flexibility in addressing changes by allowing them to be put into the backlog and then quickly worked on [24]. This relates back to chaos theory where the maximum efficiency is gained by allowing as much change as possible without drifting into complete chaos [28]. This "floating on the edge of chaos" is a common principle of complex system science [28], [33].

**4.4.3 Newly Identified Relationships between Scrum and Decision Analysis**

While clear links exist between Scrum and complex system science, identification of links between Scrum and other theories would strengthen the argument of a theoretical foundation. One area of research and its associated theories, decision analysis, has already been identified as having a strong link to systems engineering [36], [40], [93] and will be investigated here for its links to Scrum.

Scrum and other Agile approaches were developed on the assumption that the development plan cannot be known a priori [24]. This is supported by the suggested axiom of decision analysis that all the information needed to make a decision may not be available [93]. This is in contrast to traditional systems engineering approaches that assume the development model is known ahead of time [11]. The process of development is too chaotic and unpredictable to be able to establish long-term development plans. Rather, Scrum focuses on small, increments of product development, with pauses for learning and redirection after each increment. This axiom of decision analysis (all the information needed to make a decision may not be available) supports the built in instability characteristic of Scrum.
The multilearning characteristic of Scrum is implemented by having multidisciplinary teams that are able to pull information from many sources to enact a decision. Teams are also small, consisting of members that are able to work on multiple aspects of the design rather than experts for each aspect. The benefits of this concept can be traced to Hazelrigg's rational design theory, where minimizing the number of designers needed increases the rationality of the design [36].

The self-organizing teams and organizational learning characteristics of Scrum create small, multidisciplinary teams that are given latitude to determine their organization and how they interact with one another. The decision analysis axiom that decisions are multileveled and multidimensional [93] support these characteristics of Scrum since the multidisciplinary team has all the resources it needs to make multileveled and multidimensional decisions.

4.4.4 Summary of Relationships between Scrum and Theory

Scrum contains links to complex system science and decision analysis. Scrum was originally designed based on empirical evidence gathered from successful projects. As it was formalized into a process, it was linked to key traits of complexity science. These previously identified links include self-organization and chaos theory. New links to control theory and chaos theory in Scrum were identified in this chapter. We also identified elements of decision analysis in Scrum. A summary of these links is given in Table 4-1. Figure 4-5 provides a graphical representation of the relationships between Scrum and theory that the literature review found to be existing and the additional relationships that this dissertation has established.

For these reasons, we state that Scrum has relationships to established theories. It was not expressly built on theory, but principles from theory have been used to justify aspects of Scrum. These loose ties are not necessarily detrimental to its use as a process, but should be kept in mind as Scrum is further matured and modified as it invites the possibility to further improve the method as more relationships are made to theory.
Table 4-1: Summary of relationships between scrum, complex system science, and decision analysis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Built-in instability</td>
<td>Sprint backlog</td>
<td>Chaos theory</td>
<td>Information needed to make a decision may not be available</td>
<td>Game Theory</td>
</tr>
<tr>
<td>Self-organizing project teams</td>
<td>Team organizes itself</td>
<td>Self-organization*</td>
<td>Decisions are multileveled and multidimensional</td>
<td>Organization Theory</td>
</tr>
<tr>
<td>Overlapping development phases</td>
<td>Sprint</td>
<td>Chaos theory*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multilearning</td>
<td>Cross functional team</td>
<td>Self-organization*</td>
<td>Information for decisions comes from different sources and disciplines; Rational design theory</td>
<td>Learning Theory</td>
</tr>
<tr>
<td>Subtle Control</td>
<td>Sprint, daily Scrum meeting</td>
<td>Chaos theory, Control theory</td>
<td></td>
<td>Game Theory</td>
</tr>
<tr>
<td>Organizational transfer of learning</td>
<td>Cross functional team</td>
<td>Self-organization*</td>
<td>Decisions are multileveled and multidimensional</td>
<td>Organization Theory</td>
</tr>
</tbody>
</table>

* Previously identified link between Scrum and theory [24]
Figure 4-5: Summary of Relationships between Agile Scrum and Theories

4.5 The Use of Scrum for Systems Engineering

It is worth noting that Scrum was not originally viewed as a software development method. Per Takeuchi and Nonaka, Scrum was about cross-functional teams working in a “dynamic conflict” to generate ideas [94]. This was viewed as key to the approach that certain engineering managers took to make their team successful. Takeuchi and Nonaka’s ideas were applied to software as part of the Agile initiative to find better ways of handling the highly dynamic environment of software development, which resulted in the Scrum process [17]. Systems engineering can also be a highly dynamic environment, and thus Scrum principles are also of interest to systems engineers. Using Scrum for systems engineering is examined in this section in response to research question 1, objective 2.

4.5.1 Relationship with INCOSE handbook

The INCOSE systems engineering handbook defines processes involved in developing a new system [95]. The systems engineering processes outlined by INCOSE are summarized in Figure 4-6 as a traditional systems engineering “V”. A comparison of Scrum to the systems engineering processes in the INCOSE systems engineering handbook is shown in Table 4-2 and described below.

A key difference between the INCOSE processes and Scrum is that Scrum does not expressly require detailed documentation as suggested by INCOSE. Additional
processes defined by INCOSE represent processes used to transition, analyze, operate, maintain, and dispose of a system, which we consider out of scope for this analysis as they are support functions for system development or are post-development actions.

Figure 4-6: Traditional Systems Engineering Process

The business or mission analysis process in the INCOSE handbook aligns well with the planning and systems architecture phase in Scrum. Both are performed at the start of the project to outline broad goals on what the project needs to accomplish. Scrum does not enforce the strict documentation requirements of the INCOSE handbook though.

The stakeholder needs and requirements definition, systems requirements definition, and architecture definition processes in the INCOSE handbook align with the planning and systems architecture phase in Scrum. The INCOSE design definition, implementation, integration, verification, and validation processes are performed during the sprint phase in Scrum. Scrum prescribes accomplishing these tasks incrementally. In addition, verification is largely bypassed in Scrum with direct validation by the stakeholders being preferred. Once stakeholders are satisfied, the Scrum process moves to closure and the design effort is complete.
### Table 4-2: Comparison between the INCOSE and Scrum processes

<table>
<thead>
<tr>
<th>Process Step</th>
<th>Similarities</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business or</td>
<td>Performed at start of project to define overall goals</td>
<td>Details of process largely undefined in Scrum where INCOSE gives specific requirements for the step. Scrum does not require documentation. The concept of operations and operational concept, which are key products for INCOSE are not expressly defined in Scrum.</td>
</tr>
<tr>
<td>mission analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder Needs</td>
<td>Directly worked with stakeholders to elicit their needs</td>
<td>Scrum allows this process to be continually revisited. INCOSE states this process occurs early in development.</td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Requirements</td>
<td>Stakeholder needs are realized as system capabilities</td>
<td>Scrum allows for requirements changes at any point in development. INCOSE has this process occurring early in development</td>
</tr>
<tr>
<td>Definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>Performed at the start of a project to roughly define the system</td>
<td>Scrum allows architecture to evolve throughout development rather than INCOSE’s practice of trading architectures early in development to find the best option</td>
</tr>
<tr>
<td>Definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design Definition</td>
<td>Both processes generate design detail</td>
<td>Scrum develops the design incrementally. INCOSE treats this as a single phase.</td>
</tr>
<tr>
<td>Implementation</td>
<td>Process details are purposely vague as they vary by product</td>
<td>Scrum implements the design incrementally. INCOSE treats this as a single phase.</td>
</tr>
<tr>
<td>Integration</td>
<td>Join the parts of a system into a single product</td>
<td>Scrum performs integration incrementally. INCOSE treats this as a single phase.</td>
</tr>
<tr>
<td>Verification</td>
<td>System capabilities are checked</td>
<td>Scrum performs verification incrementally; step is largely bypassed by Scrum for direct validation to stakeholders. INCOSE treats this as a single phase.</td>
</tr>
<tr>
<td>Validation</td>
<td>Checks that stakeholders are satisfied with the system</td>
<td>Scrum performs this incrementally. INCOSE treats this as a single phase.</td>
</tr>
</tbody>
</table>

Figure 4-7 overlays the INCOSE systems engineering process with the Scrum process in a way that allows the execution of the INCOSE processes in an Agile manner. The INCOSE process steps are shown in blue rectangles and the Scrum process steps in black.
rectangles. The process starts by performing the first 4 steps of the INCOSE process to sufficient detail to allow initial work to begin. The next five steps are then performed for small increments of added capability into the system model. At the end of each sprint, changes to the stakeholder requirements, system requirements, or architecture definition can be planned for the next sprint. Once the design is ready for release, the process closes.

**Figure 4-7: Overview of INCOSE Systems Engineering Processes executed through Scrum with key MBSE enablers**

### 4.5.2 Difficulties in Implementing Scrum for Systems Engineering

There are some possible shortcomings with Scrum when it is used for systems engineering. These shortcomings trace back to the need to rapidly change and evaluate a system in Scrum. For software systems, the typical application of Scrum, this is not an issue. Object-oriented programs can rapidly be changed and reconfigured to add new functions [85]. They can also be tested incrementally as they are developed [24].

To use Scrum for systems engineering requires the system to itself be Agile [85]. This means the system can be rapidly reconfigured and altered to support the timelines associated with Scrum. It also means that the system can be incrementally verified and validated. Complex systems do not present the agility needed to support this type of
work. They tend to have complex and custom solutions that can be difficult to quickly modify and they may not have intermediate versions for testing [5]. Thus, using Scrum directly for systems engineering is not feasible for all systems.

4.5.3 Agile SE Enabled Through MBSE

Not all systems lend themselves to being changed rapidly and to continuously being verified. For these types of systems, a surrogate is needed that can be rapidly changed and can provide a means for continuous verification to enable Agile processes. The system models used in MBSE can serve as this surrogate for the system hardware and software that has yet to be built. By doing this, the system model provides a way to continuously integrate the system and check final functionality. The system model provides a needed Agile architecture to facilitate using Scrum [85]. The use of models for systems engineering is supported by the decision analysis axiom that productivity can be increased by using analysis, visualization, and synthesis [93]. MBSE also enables continuous verification of the system through system models [11]. Continuous verification is a key aspect of Scrum, so this aspect of MBSE supports the use of Scrum [11], [96]. Traditional systems engineering methods that rely on paper documents may not have an architecture surrogate that is Agile enough to facilitate performing Scrum.

Figure 4-7 includes three points in the proposed Scrum/INCOSE Processes merger where MBSE enables some aspect of Scrum. These points include: rapid changes to the system model, continuous integration and testing, and system model demonstrations. These points show how Scrum requires a model representation of the system where the model functions as the Agile architecture of the system. Without system models, the applicability of Scrum to complex systems common to the aerospace and mechanical engineering fields would be severely limited.

Given the aspects of MBSE that support Scrum described above, it is concluded that MBSE is a key enabler of doing Agile systems engineering. This is not without difficulties and challenges. Some system engineering activities require time beyond a traditional sprint timeline, and the sprint may need to be adjusted to last longer when doing systems engineering [11], [68]. The MBSE surrogate is also only as good as the level of fidelity in the model. All models require some level of abstraction. It is important
that models used for MBSE do not make an important aspect of the system overly abstract and the Agile development effort use the correct level of abstraction.

Foundational issues for using MBSE while doing Scrum need to be addressed, such as the size of Scrum teams, how to integrate multiple teams through a Scrum of Scrums [24], creating the definition of done for MBSE work, and stopping criteria to hand off the design to fabrication [11]. The definition of “done” is a key aspect of Scrum where team members decide a priori what the closure criteria is for a task [24]. The definition of done could be related to decision analysis as the decisions that shape this definition define system characteristics and how they should perform, essentially defining the function of the system.

4.6 Summary

Chapter 4 has shown much of the history behind Scrum. This includes the original findings by Takeuchi and Nonaka that were used to describe a highly productive development process. These findings were used to create Scrum. The findings and how they were incorporated into Scrum were analyzed to show how they trace to principles and characteristics of theories. This demonstrated that the major aspects of Scrum have analogous aspects in established theories, including complex system science and decision analysis. Finally, the use of Scrum for systems engineering was examined. This showed that it is possible to line up the Scrum process with established standards for systems engineering. MBSE was also analyzed to show how it is an enabler for an Agile systems engineering process. The ability to rapidly reconfigure and test models of a system developed through MBSE enables Agile-type development on a wide variety of systems.
CHAPTER 5

5.0 MODELING AND SIMULATION PROOF OF CONCEPT

5.1 Overview

This chapter outlines the work done to develop the proof of concept for an agent-based model and simulation of design teams performing waterfall and Agile work. The chapter outlines the model used for the agents, and how the simulation utilizes, learning, waterfall, and Agile concepts. The proof-of-concept simulation consists of a two-person team designing a beam. This proof-of-concept provides an initial understanding of how to construct an agent-based simulation of design teams using FBS. The work is valuable in determining the construct for more complex simulations detailed in chapter 7. The work in this chapter indirectly supports research question 2 by determining concepts that can be used in a more complex agent-based simulation to determine the benefits and drawbacks of Agile vs. waterfall. An overview of this chapter is provided in Figure 5-1.
5.2 The Agent Model

The agent model was developed to support research question 2, objective 1 to establish the foundation for the modeling and simulation effort. In this research, designers are modeled as agents using the FBS model described in section 2.5. The FBS model was mechanized in this research as a first order Markov process where transition from one state to the next is governed by a probability. These probabilities were set based on reported values from empirical studies of FBS [3]. The same initial probabilities were used for both pseudo-waterfall and pseudo-agile processes. This represents the same agents being used for both processes. The Bs and S states of the FBS model were collapsed into a single state for purposes of modeling, which is represented by the dashed box in Figure 5-2. This collapses the synthesis, analysis, and evaluation activities into a single activity. This was done so that the FBS model could be represented as a Markov process. As-is, the FBS model could not be represented as a Markov process since the path from F to Bs has no return path. Also, the evaluation process does not actually result in changing states. This may cause the FBS model to lose some amount of fidelity, but it still allows the overall FBS flow to be represented so long as the analysis and evaluation
activities are captured as part of a modified synthesis activity. An example of the model with notional transition probabilities is shown in Figure 5-2.

![Figure 5-2: FBS Model Setup for Markov Simulation with Example State Transition Probabilities](image)

Agents are represented in the simulations with the FBS Markov process. The model keeps track of the agent’s current state. Each of the agents are simulated as transitioning through the implemented FBS model using random draws to calculate a value that is compared to the transition probabilities to determine the state for the next time step.

Three FBS models are being tracked in the waterfall and agile simulations, one FBS model for each of the two designers (agents) and one FBS model for the entire system design team. The FBS model of the system design team is tracked in the pseudo-waterfall simulation but doesn’t play a role in decision making. In the pseudo-agile simulation, it does play a decision making role as it represents the data that the two agents use to make decisions representing the collaborative nature of the agile process. The FBS model of the entire system design team is at the lowest state of the FBS models of the two engineers.
Table 5-1 examines the attributes of agent-based models to the one used in this simulation. It can be seen that the main attributes of agent-based models have been incorporated into this simulation.

**Table 5-1: Summary of Agent-Based Model Properties**

<table>
<thead>
<tr>
<th>Attributes of Agent-Based models from Perrone [43]</th>
<th>Corresponding attribute in the subject model</th>
</tr>
</thead>
<tbody>
<tr>
<td>A single discrete entity with decision making ability</td>
<td>Each engineer has the ability to make decisions to set design variables and proceed through a system design process</td>
</tr>
<tr>
<td>Contained in an environment with other agents</td>
<td>There are two agents in this simulation</td>
</tr>
<tr>
<td>Directed to meet a goal, but not necessarily to maximize an objective</td>
<td>The agents drive to an acceptable design, one that results in positive net profit</td>
</tr>
<tr>
<td>Able to function autonomously in its environment</td>
<td>The agents function independent of one another with separate transition matrices for each</td>
</tr>
<tr>
<td>Has the ability to change and adapt based on its experience</td>
<td>Each agent learns from past design choices based on the learning algorithm described in section 3.3.</td>
</tr>
</tbody>
</table>

**5.3 Proof of Concept Simulation Design**

A proof of concept simulation was built in accordance with research question 1, objective 2. This proof of concept tests the ability to model and simulate a design team using FBS as an agent model. A small design team consisting of two engineers, a materials engineer and a mechanical engineer, was modeled. Their purpose is to develop a beam that would create a profit for their company. The beam is assumed to have a square cross section and could be made from four different types of materials. The Factor of Safety (FOS) of the beam relative to a given load is a measure of effectiveness used to determine the value of the beam. FOS is used in a Hyperbolic Absolute Risk Aversion (HARA) function to create the revenue equations showing in Equation 1. This HARA
functions is used to determine revenue the beam could generate as shown in Figure 5-3. The HARA function is built around the assumption that FOS is the primary means of a customer gaining utility for a beam of this type. HARA was chosen as it traces well to utility theory [78]. The specific parameters of the HARA equation shown in Equation 5-1 were chosen to create a constrained design environment for the agents in which only a small subset of the design space yields designs that result in a net positive profit. The cost of the materials subtracted from the revenue determined the profit of the beam as shown in Equation 5-2. Creating a positive profit is the goal of the agents.

\textbf{Equation 5-1:} \quad \textit{Revenue} = d \left[ \frac{1-\gamma}{\gamma} \left( \frac{af_{\text{FOS}}}{1-\gamma} + b \right)^\gamma \right] + c

Where $\gamma = 18$

$b = 1.4$

$a = 1$

$c = 604745$

$d = 1500$

\textbf{Equation 5-2:} \quad \textit{Profit} = \textit{Revenue} - \textit{Cost of materials}

Two simulations are used in this scenario, one to simulate a pseudo-waterfall design process and another to simulate a pseudo-Agile design process. In both
simulations, the beam design variables (material type and cross section area) are chosen using a random draw from a uniform random distribution.

5.3.1 Pseudo-Waterfall System Design Process Simulation

The simulation was created with the FBS model to march the two agents through a serial, waterfall-like process to design a beam that produces positive net profit for the company as a satisficing criterion. The waterfall process was setup as represented in Figure 5-4. The process starts with the materials engineer performing his/her work to determine the material type for the beam. After the material type is determined (represented as the FBS model transitioning through all states from R to D), the mechanical engineer begins work. The mechanical engineer begins in state R and will eventually transition to state D. There is a possibility that if the materials engineer selected a certain material, that the mechanical engineer will determine that the design goal is not reachable (exhaust all possible design options). The process then starts over with the materials engineer reworking his solution to find a new material.

![Figure 5-4 Pseudo-waterfall beam design process](image)

5.3.2 Pseudo-Agile System Design Process Simulation

Since the proof-of-concept beam design is simple, the pseudo-Agile process consists of a single sprint, which is reasonable since the design effort for such a design artifact is expected to take much less than a month. The Agile process is started by
having both the materials engineer and the mechanical engineer work in parallel as shown in Figure 5-5. This represents them working together in a Scrum process. The simulation requires the two agents to work together by not allowing the agents to ever be more than one state apart. The agent that is falling behind will be tasked with an extra work cycle until he/she reaches the same state as the leading agent and they can work together.
Figure 5-5 Pseudo-Agile beam design process

These simulated design processes are very rudimentary and do not capture all the complex interactions and work that will be performed by design teams doing actual waterfall and Agile work. The processes do attempt to capture Takeuchi and Nonaka’s description of a traditional development approach where each discipline in an
organization takes their turn, in serial, to develop a product and their description of the “rugby” method, where a "Scrum" of disciplines moves the ball downfield all working simultaneously to achieve a common goal [12]. The main difference between the pseudo-waterfall and pseudo-Agile efforts for the proof of concept is that the engineers work in serial for the pseudo-waterfall effort and in parallel for the pseudo-Agile effort. It is recognized that parallel and serial efforts do occur in both approaches in the real world, but given the simple design effort being modeled and the small team (two engineers), this is assumed to be the main difference in process between the Agile and waterfall design methods.

The purpose of this proof of concept was to prove that an agent-based simulation of the FBS model was feasible, and not to make any conclusions about the differences between Agile and waterfall processes. The pseudo-processes are meant to represent sufficient complexity to prove the ability to make a simulation of the FBS framework.

5.3.3 Learning

Bayesian learning [97] was incorporated into the simulation to model changes in transition probabilities resulting from learning by the agents. The learning is based on the principle of bivalence, i.e. a chosen design variable is either good or bad. It is also assumed that knowledge is available on how to improve the design by either increasing or decreasing a given design variable. Version space learning [98] is used to implement Bayesian learning in the simulation. Version space learning works by truncating the available variables that can be picked, based on what has been learned. For example, if a design variable were picked that caused a reformulation, the values of that variable that make the design move further away from its optimal value would be truncated. This leads to the probability, $P(d|R_1)$, of choosing a design variable, $d$, given that reformulation 1 occurred. This can be repeated for further reformulations until a value is chosen that allows the design to move to the documentation phase. Figure 5-6 is an example of this process with FBS states that can be entered from state “S” listed across a uniform probability distribution that represents the possible design choices. The size of region “D” is based on the number of design alternatives within the bounded design space that produces a positive net-profit. The other 6 regions were sized equally according to the remaining design space.
The example in Figure 5-6 represents a design space with 1 design variable and it begins when the FBS state “S” is first entered. The first value causes a type 1 reformulation and the values that are lower than the chosen value are truncated from the uniform distribution. The overall probability density remains constant, so the remaining uniform distribution grows in height. In the second distribution, a 2nd value is chosen, which also causes a type 1 reformulation, and the portion of the distribution to the right of this value is truncated due to learning. The 3rd value chosen (illustrated in the 3rd distribution) is one that allows the FBS to transition to state “D” representing the completion of the design.

**Figure 5-6 Version Space Learning Example**

Figure 5-7 illustrates the concept of a design space with 2 design variables. The red region represents the region that design variables are picked from if a type 3 reformulation (RF3) is picked from a random draw. If an agent picked from the RF3 region, then the agent would revert to the Function (F) state. The yellow region represents the design space that the design variables are picked from if a type 2 reformulation (RF2) is picked from a random draw, then the agent would move back to the expected behavior.
state (Be). The green region represents the design space that would be used if a type 1 reformulation was chosen and the agent would remain in the structure (S) state. The blue region represents the design space that satisfies the design closure criteria, in this simulation positive net profit, and the agent would advance to the documentation state (D).

![Design space example with 2 design variables](image)

**Figure 5-7  Design space example with 2 design variables**

Figure 5-8 shows two example design spaces along with the corresponding distributions for one of the design variables. The dashed lines in the figures at the left indicate the value of $X_1$ that is being used for the $X_2$ distributions on the right. The upper figure shows most of the design variables are distributed to S, Be, and F spaces on the left side of the design space. The lower figure shows a large space where the design meets goals in the D space in the center.
Learning is slightly different in the pseudo-Agile simulation versus the pseudo-waterfall simulation. In the pseudo-waterfall simulation, each engineer learns by himself/herself based on the results of their design work. Each engineer can encounter a type 1, type 2, or type 3 reformulation when performing design work. In the proof-of-concept, only if the mechanical engineer exhausts all his/her design space does the materials engineer get tasked with learning. In the Agile simulation, it is a single process that involves both engineers. This was done to represent the collaborative nature of the Agile process.

In the real world multiple types of learning and collaboration are possible. The method used in this simulation is a simplification of the real world and is meant to represent a way to differentiate the two approaches with one possible type of learning. The learning is based on the type of reformulation drawn. For the proof-of-concept, Type 3 reformulations are assumed to automatically require a change in material type. Type 2 reformulations require the mechanical engineer to narrow his/her design variable. Type 1 reformulations require the mechanical engineer to narrow his/her design variable, and if the mechanical engineer exhausts all design options, the materials engineer is tasked with finding a new material to use for the beam and the design process starts over with the learning incorporated.
5.4 Simulation Verification and Validation

The simulation was developed in MATLAB® a proprietary program language. It was chosen due to the familiarity the author has with it and its inherent ease of analyzing output data and plot generation. The code was verified through several steps:

- The code was debugged using the MATLAB® built-in debugger
- Transition times were verified to match the closed-form solution for the Markov process
  - Transition probabilities of 0, 1, and 0.5
  - Learning algorithm turned off
- The learning algorithm verified by stepping through execution of the algorithm with several different examples and observing that the algorithm correctly narrowed the design space as expected

Simulation validation was handled three ways. First, the FBS model used in the simulation is widely accepted as a valid, first-order model governing the behavior of designers [47], [48]. It has been validated through case studies of designers [3], [53]. Second, input data to the model was calibrated with values obtained from FBS case studies [3]. This resulted in the model being adjusted so that it matched observed behavior of designers (the simuland) [84]. Finally, predictive validation [84] was used to compare the simulation output with reported data. The results of this comparison are given in section 5.5, and shows that the simulation output matches the expected range of data seen in literature. This conclusion is not meant to validate that Agile or waterfall is superior for beam design as the pseudo-processes used are too simple to draw this conclusion, but rather to show that these simple representations produce an output within expected bounds, showing that the modeling and simulation approach is reasonable.

5.5 Proof of Concept Results

The results show that the individual effort for each of the two engineers is dependent on the system design process used. This can be seen in Figure 5-9 through Figure 5-12. Figure 5-9 shows a histogram of how many 4-hour design sessions are required to complete the material design using the pseudo-waterfall process. Figure 5-10 shows the same information for the pseudo-Agile process. Both histograms use the same bins size
and number. The average number of 4-hour design sessions needed to complete the material design decreases from 19.3 sessions with the pseudo-waterfall process to 14.7 sessions with the pseudo-Agile process. The shape of the histogram for the pseudo-Agile process is also more compressed towards the left side of the plot, showing how the pseudo-Agile process tends to drive the material design to completion with fewer design sessions. The pseudo-Agile process also has less variability for the materials engineer, with the standard deviation of the number of 4-hour design sessions needed to complete the materials design decreasing from 11.1 to 6.9 sessions when using the pseudo-Agile process.

![Histogram of waterfall simulation times for material design](image)

**Figure 5-9** Materials engineer effort for pseudo-waterfall process. Average is 19.3 four-hour sessions
Figure 5-10: Materials engineer effort for pseudo-Agile process. Average is 14.7
four-hour sessions

Figure 5-11 shows a histogram of how many 4-hour design sessions are required to complete the mechanical design using the pseudo-waterfall process. Figure 5-12 shows this information for the pseudo-Agile process. Both histograms have the same bin size and number. The average number of 4-hour design sessions needed to complete the mechanical design increases from 13.5 sessions with the pseudo-waterfall process to 17.6 sessions with the pseudo-Agile process. The shape of the histogram for the pseudo-Agile process shows there is significantly more variability in the number of design sessions required to complete the mechanical design. The standard deviation for the number of 4-hour design sessions needed to complete the mechanical design increased from 6.1 to 8.9 when using the pseudo-Agile process. This is not unexpected as Agile processes are known to require a significant time commitment from team members [12].

The approach to collaborative learning used in the pseudo-Agile simulation is the reason for the difference in effort for both engineers. In the pseudo-waterfall simulation, the materials engineer spends some effort learning before the mechanical engineer starts work. In the pseudo-Agile simulation, this learning is done simultaneously. Agile processes tend to require the mechanical engineer to spend more time looking at different possible solutions while the materials engineer learns faster and spends less time performing work. The overall amount of effort (effort-hours) between the two engineers
is similar, with 32.8 sessions being required on average to complete the design with the pseudo-waterfall process and 32.3 sessions being needed for the pseudo-Agile process. Given the satisficing nature of the design goal, there is no improvement in performance of the beam design for the pseudo-Agile simulation versus the pseudo-waterfall simulation.

Figure 5-11 Mechanical engineer effort for pseudo-waterfall process. Average is 13.5 four-hour sessions

Figure 5-12 Mechanical engineer effort for pseudo-Agile process. Average is 17.6 four-hour sessions
The total amount of time needed to complete the beam design was examined for both pseudo-waterfall and pseudo-Agile processes. Efficiency gains from waterfall to agile processes in literature tend to be 36% to 50% in most cases [27], [31], [67]. Outlier examples show gains as high as 600% and other outliers show a productivity loss [24]. The simulation showed a 44% gain in efficiency when the designers changed from a pseudo-waterfall to a pseudo-Agile process. Since results are within the range reported in case studies, the simulation is determined to have reasonable results. Figure 5-13 and Figure 5-14 show a scatterplot of the time needed to complete the beam design. The average time to complete the design and standard deviations from the average are also denoted on the figures. Of note, the standard deviation decreased from 15.5 four-hour design sessions for the pseudo-waterfall process to 9.2 four-hour design sessions for the pseudo-Agile process. This indicates the pseudo-Agile process tends to have less variation.

Figure 5-13 Total number of 4-hour design sessions needed to complete a beam design using a pseudo-waterfall process. Average is 32.9
Figure 5-14 Total number of 4-hour design sessions needed to complete a beam design using a pseudo-Agile process. Average is 18.4

5.6 Summary

The proof-of-concept simulation exercise resulted in several findings that were incorporated into the follow-on waterfall and Agile simulations of a software design effort and a launch vehicle design effort. First, the methodology to develop such a simulation was determined to be valid. The simulation accurately predicted the difference between the Agile and waterfall process efficiencies. Second, the complexity of the process simulation needed to be increased to better approximate real-world processes when modeling more complex design efforts. Third, after discussion with subject matter experts, the learning algorithm was not carried into future work. There were concerns that not enough research into human learning for complex engineering exists to justify the algorithm, and that the stochastic nature of the FBS model could provide enough approximation of learning until further research is performed. These findings were incorporated into the software design team and launch vehicle design team simulations described in the following chapters.
6.1 Overview

Chapter 6 provides an overview of the work done to calibrate the FBS model for use in simulations. The FBS model is a conceptual model, which requires calibration to mechanize into a model that can be simulated. This chapter covers the work to mechanize the model into a first order Markov process, calibrate the model, and verify that the calibration is correct. Figure 6-1 provides an overview of the chapter.
6.2 FBS Model Calibration Methodology

A method to calibrate the FBS model was performed in support of research question 2, objective 1. The FBS model requires transition probabilities for it to be used in simulation of how designers work on real problems [3]. Existing data from FBS validation efforts is based on mock design sessions for simple systems [3], [48]. For this dissertation, data from a development program utilizing a systems engineering “V” model approach with incremental Design Analysis Cycles (DAC) for a complex aerospace system is used to calibrate the FBS model and derive transition probabilities [68]. This is performed by a four-step process:
1. Scheduled tasks in the complex aerospace system development are analyzed and assigned to an appropriate state transition in the FBS model
2. Task durations are normalized to a single engineer working full-time
3. The normalized duration of the tasks is analyzed to determine a distribution and fit for the distribution
4. The distribution is used to determine transition probabilities

Figure 6-2 illustrates the methodology using the 4 steps above. Scheduled tasks are assigned an appropriate state transition in the FBS model per the first step. The type of task is used to determine which FBS transition represents the task. Example tasks include developing performance requirements for a computer, developing a calibration approach for an instrument, and designing a structural member. This is done by reviewing the task description and matching it to a transition in the FBS model. It is worth noting that in our execution of the FBS model, we have collapsed the analysis, evaluation, and synthesis transitions into a single transition as denoted by the dotted box on the FBS model in Figure 6-2. This was done so that the FBS model could be represented as a first order Markov process. Data on how many hours each task takes is used to normalize the duration of the task to a single performer. This normalizes tasks to be representative of how long they would take a single individual to complete since the FBS model is built around a single, full time designer. This corrects the data for tasks that may be worked by multiple people simultaneously, or tasks that may be worked part-time by a single person. For example, if a task was shown to have a duration of 2 weeks and required 40 hours to complete, it would be normalized to a 1-week task. The data points that correspond to each transition type were then grouped and analyzed to determine an appropriate probability distribution and fit the data to that distribution. These distributions are then used to derive transition probabilities based on the cumulative distribution function (CDF).

Any model consists of assumptions. These assumptions were made during this initial effort and could be addressed with future work. The assumptions made in this approach are:

- The duration of tasks is invariant with respect to the type of system being developed for efforts analyzed with the calibrated model.
• The duration of tasks is invariant with respect to the type of system engineering process used.
• Tasks can be uniquely assigned to one of the FBS state transitions.
• Normalizing task duration to a single performer is representative of the effort it would take to complete the task.
Figure 6-2: Example of FBS Calibration Methodology
6.3 Results

Data from the development of a complex aerospace system consisting of task durations and hours to complete the tasks was used as an input to the methodology described in section 6.2. The raw data is proprietary and details of it cannot be provided. However, the results of the analysis are provided in this section. The task durations are in calendar days, including days where work is not performed (weekends and holidays). Hence, outputs from the calibrated model should be considered in terms of calendar days, not just work days. Table 6-1 provides a summary of the number of samples of each transition type that were derived from the data.

Table 6-1: Number of Samples Derived from Source Data

<table>
<thead>
<tr>
<th>Transition Type</th>
<th>Number of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-&gt;F</td>
<td>36</td>
</tr>
<tr>
<td>F-&gt;Be</td>
<td>22</td>
</tr>
<tr>
<td>Be-&gt;S</td>
<td>34</td>
</tr>
<tr>
<td>S-&gt;D</td>
<td>17</td>
</tr>
<tr>
<td>Reformulation Type 1</td>
<td>17</td>
</tr>
<tr>
<td>Reformulation Type 2</td>
<td>10</td>
</tr>
<tr>
<td>Reformulation Type 3</td>
<td>25</td>
</tr>
</tbody>
</table>

After the data was normalized to a single performer, the distribution of the data was examined. Each of the durations to complete a transition to a new state were distributed in a way that consisted of an initial peak followed by a roll-off to a very low level. Several different distributions for fitting were examined and the Gamma distribution was chosen because it generally modeled the data well and is often used to model the time between events [99]. Distributions examined but did not exhibit good fits include normal, log-normal, Poisson, and exponential. The Maximum Likelihood Estimate (MLE) method was used to derive the parameters of the gamma distribution, the shape parameter, and the scale parameter, to fit the data for each transition type. A Chi-squared goodness of fit test was used to evaluate if the gamma distribution fit was reasonable for the data. The test concluded that a null hypothesis that the data fit the
gamma distribution could not be rejected at a 5% significance level. This test was performed for the fit of the data on each transition type with the same result, that the null hypothesis could not be rejected.

The following figures are histograms of the data grouped by their transition type, with a blue line showing the corresponding gamma curve fit of the data. The left Y-axis corresponds to the curve fit, showing the Probability Density Function (PDF) of the gamma distribution. The PDF was discretized for each figure into 1-hour intervals with the total number of intervals equaling the size of the maximum histogram bin. For example, if the maximum histogram bin was 150 hours, the PDF was discretized into 150 parts. The right Y-axis illustrates the number of samples in each bar of the histogram. The X-axis shows the time needed to complete the state transition.

Figure 6-3 below shows the results of the R->F transition probability analysis. The data indicates that the majority of these types of tasks are completed in the 9-27 hour timeframe. After that, there is a sharp decline in the number of tasks that take longer, which quickly normalizes to a near steady-state value. The gamma distribution fit captures this characteristic of the data. The fit may overestimate the data in the 30-60 hour region and under predict in the very high hour region, but overall, it provides a reasonable estimate of the underlying data per the analysis of goodness of fit. A Chi-squared goodness of fits test for the gamma distribution is not rejected at a 5% significance level.
Figure 6-3: R->F task histogram and gamma distribution fit

Figure 6-4 shows the results of the F->Be transition probability analysis. The data indicates most tasks are completed in under 30 hours with a few tasks taking much longer. The gamma distribution fit to this data captures this trend and may somewhat under predict the 20-30 hour range, but overall, given the small number of samples, the fit is reasonable given the results of a goodness of fit test. A Chi-squared goodness of fits test for the gamma distribution is not rejected at a 5% significance level.
Figure 6-4: F->Be task histogram and gamma distribution fit

Figure 6-5 shows the results of the B_{e->S} transition probability analysis. Like the previous sets of data, this data shows a sharp drop in task time after a certain amount of time has passed. In this case about 90 hours. After 90 hours, the data indicates a near steady set of probability for other tasks with one or two samples showing up in each bar as task duration increases. A Chi-squared goodness of fits test for the gamma distribution is not rejected at a 5% significance level.
Figure 6-5: Be-S task histogram and gamma distribution fit

Figure 6-6 shows the results of the S->D transition probability analysis. A significant number of these documentation tasks take a rather short amount of time, less than 25 hours, after which the data indicates a sharp decline in the number of tasks as task duration increases. Given the small amount of data, the gamma distribution fit appears to capture the characteristics of the data, with a sharp rise and gradual decrease over the interval the data indicates. A Chi-squared goodness of fits test for the gamma distribution is not rejected at a 5% significance level.
Figure 6-6: S->D task histogram and gamma distribution fit

Figure 6-7 shows the results of the Type I reformulation transition probability. The data indicates an odd trend of a slight decline to low levels with a high number of tasks taking much longer in the 108 – 144 hour bin. This could be due to the low number of samples, 17, in the dataset. The gamma distribution fit does not capture the feature of the large number of tasking in the 108 - 144 hour region. Despite the small sample size, a Chi-squared goodness of fits test for the gamma distribution is not rejected at a 5% significance level.
Figure 6-7: Type I Reformulation task histogram and gamma distribution fit

Figure 6-8 shows the results of the type II reformulation transition probability analysis. The data indicates that most of the type II reformulation tasks take less than 22 hours to complete, with only a few samples at a higher task duration. Like the type I reformulation, there is a small sample size for type II reformulations. The small sample size of type I and type II reformulations is likely due to requirements instability in the project the data was gathered from. The project experienced several changes in system requirements, which required designers to readdress the basic functions of their designs rather than make smaller changes. Due to the small sample size, the gamma distribution fit to the data is a rough estimate of the underlying full population of data. The gamma distribution fit does capture the characteristics of the data well, namely the sharp drop in the number of tasks with a duration greater than 22 hours. A Chi-squared goodness of fit test for the gamma distribution is not rejected at a 5% significance level.
Figure 6-8: Type II Reformulation task histogram and gamma distribution fit

Figure 6-9 shows the results of the Type III reformulation transition probability analysis. The data shows a gradual decline in the number of tasks as task duration increases up to 65 hours. After that, the tasks intermittently take longer, up to 200 hours. The gamma distribution fit to this data captures the data characteristics and appears to provide a reasonable estimate of the characteristics of the data. A Chi-squared goodness of fits test for the gamma distribution is not rejected at a 5% significance level.
Overall, the gamma distribution provides the ability to fit the available data reasonably. This was confirmed with a Chi-squared goodness of fit test, which did not reject the null hypothesis at the 5% significance level. Some of the task transitions suffered from a low number of samples and possible outliers. We plan to account for this in simulations that use this data through a Monte-Carlo approach where we can analyze the uncertainty in the gamma distribution parameters to capture their effects on the performance estimates generated by the calibrated FBS model.

The transition probabilities for an 8-hour work period with +/-90% confidence intervals are shown in Table 6-2. The FBS model with the calibrated transition probabilities for an 8-hour work period is shown in Figure 6-10. A 90% confidence interval for the gamma distribution parameters was used to drive variability into the agent model so that it could be used in a Monte-Carlo simulation. Based on the data from this confidence interval, the lowest and highest possible performing agents were determined and are shown in Figure 6-11 and Figure 6-12, respectively. These two agents represent
the extreme values that could be possible in the simulation, although values that are closer to the mean performer are more likely.

Table 6-2: Mean and 90% Confidence Interval (CI) Transition Probabilities for an 8 Hour Work Period

<table>
<thead>
<tr>
<th>Transition Type</th>
<th>90% Lower CI Probability</th>
<th>Mean Probability</th>
<th>90% Upper CI Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-&gt;F</td>
<td>0.0097</td>
<td>0.3922</td>
<td>0.4019</td>
</tr>
<tr>
<td>F-&gt;Be</td>
<td>0.0214</td>
<td>0.6104</td>
<td>0.6318</td>
</tr>
<tr>
<td>Be-&gt;S</td>
<td>0.0056</td>
<td>0.2658</td>
<td>0.2713</td>
</tr>
<tr>
<td>S-&gt;D</td>
<td>0.0111</td>
<td>0.1816</td>
<td>0.3269</td>
</tr>
<tr>
<td>Reformulation Type 1</td>
<td>0.0186</td>
<td>0.1853</td>
<td>0.3336</td>
</tr>
<tr>
<td>Reformulation Type 2</td>
<td>0.3297</td>
<td>0.3571</td>
<td>0.6516</td>
</tr>
<tr>
<td>Reformulation Type 3</td>
<td>0.1849</td>
<td>0.2759</td>
<td>0.6406</td>
</tr>
</tbody>
</table>

Figure 6-10: Mean Performing Agent Model
A verification of the calibrated model was performed to ensure it was calibrated properly. Per the standard definition of verification, this effort was a check to ensure that the model was built correctly with the input data [84]. The calibrated model was used to calculate a mean first passage time [100] for the design effort with transition probabilities based on 8 hours i.e. the probability of transitioning from one state to the next in 8 hours.
or less. The mean performing agent from Figure 6-10 was used in this analysis. The input data has 79 individuals in it, which this analysis has reduced to a model of the average of the 79 individuals. The 79 individuals each worked on different parts of the same complex aerospace system to develop a design iteration in a 6-month period. The mean first passage time when using the mean performing agent, is 190 calendar days, meaning the average performer will take 6.3 calendar months to complete their design effort. This compares well with actual performance data from the subject project where the 79 individuals developed design iterations on a 6-month design cycle. There is a 4% difference between the calibrated model and the actual program schedule used as input data, showing that the model is accurately representing the actual cycle time of the underlying data. This shows that the calibrated model is in agreement with the data used to calibrate it and the calibration is acceptable.

The data also shows significant variability. With the best-case performance for each transition found in the 79 individuals put into a single FBS model, the data indicates that a design can reach the documentation state in 5 days. Doing the same exercise with the worst-case data indicates the design will reach the documentation state in 9 years. These extreme positions are unlikely to occur. When implementing this calibrated FBS model in a simulation, a random draw from a normal distribution is used to determine each of the transition probabilities. There are six random draws needed to generate an FBS transition matrix (they correspond to each potential transition from one state to another), and these extreme performance levels are 2.48 standard deviations from the mean since they are the worst in 79 samples for each transition. Thus, the extreme positions cited above represent cases that are 14.88 standard deviations from the mean, indicating they are very unlikely to occur.

6.5 Summary

The work documented in this chapter shows that it is possible to translate real-world performance data of an engineering team into a calibrated FBS model. This was performed through an analysis of task duration and task type data from a real-world complex aerospace system program. The calibrated model was shown to have a mean first passage time that agrees well with the actual performance data from the project. The
ability to calibrate the FBS model to reflect the work of the engineering team is useful so that the design team simulation used to answer research question 2 is built upon a validated model, calibrated with real-world data that is representative of the work being simulated.
7.1 Overview

Chapter 7 documents the work done to develop agent-based simulations of design teams using waterfall and Agile systems engineering processes. Two cases of these simulations are used. One is a simulation of a design team developing a complex software program. The second is a simulation of a design team developing a launch vehicle. The primary difference between the launch vehicle and software design simulations is the modeling of the couplings in the design of the launch vehicle. The software design simulation was built with the assumption that each subsystem design team could be built in isolation from one another with internal subsystem characteristics not affecting other subsystems. This is not assumed to be true for the launch vehicle, where decisions for each subsystem have an effect on the system performance. Figure 7-1 provides an overview of the chapter.
7.2 Design Team Structure

The agents are arranged in a hierarchical structure representing the teams that build the various parts of the system. In the case of the software system development, the computer program has 4 major modules. Each module is comprised of 2 – 4 sub-modules.
that are developed by individual teams. This is represented in Figure 7-2 with each development team represented by a box in the figure.

**Figure 7-2: Representation of team and program structure for software program design simulation**

At the integrated program and module level, only one agent is used to represent the integrating functions. These agents are equivalent to team leads in the waterfall construct, or product owners in the Agile construct. The teams at the sub-module level each consist of eight designers. Eight was chosen as the team size to match scrum recommendations [24]. The sub-modules are further divided into ten individual functions or features that the design teams create at the sub-module level.

For the launch vehicle design simulation, the agents are arranged in a hierarchical structure representing the teams that build the various parts of the launch vehicle. Team 1 is for the first stage with sub teams developing the engine, propellant tanks, structure, and defining requirements. Team 2 is for the second stage with the same sub teams as the first stage. There is also a 3rd team that defines the payload and mission profile with a single sub team. Each of the teams have a lead that participates in planning and performing the design work. There is also an overall lead for the design effort. Each of these individuals are represented as an agent. Each sub team has 8 members based on recommendations for the size of scrum teams [27]. This organization is shown in Figure 7-3 below with the groups in the Scrum of Scrums shown horizontally across the organization. The same structure is used in the waterfall simulation as well. The top scrum group along with the intermediate scrum group set the initial goals for the project and monitor progress. The lowest “Scrums” group is where actual design work is performed. In the scrum simulation, these higher level leaders have minimal participation in the design work per scrum guidance [27]. In the waterfall simulation, these leaders are more involved and participate in the lower level design work.
7.3 Waterfall Process

A waterfall simulation was developed in support of research question 2, objective 1. The waterfall simulation was created to follow a typical development program through several milestones. These are typical design milestones and are advocated for in traditional systems engineering standards such as MIL-STD-499 [57]:

- System Requirements Review (SRR)
- System Functional Review (SFR)
- Preliminary Design Review (PDR)
- Critical Design Review (CDR)

For the waterfall simulation, these design phases were mapped to the FBS model as follows:

- **SRR** – corresponds to the transition from R to F
  - A goal of SRR is to translate customer requirements into system specific functions [57]. This aligns well with the FBS transition of formulation where functions are formulated from requirements.

- **SFR** – corresponds to the transition from F to Be
  - A goal of SFR is to create a design approach that performs in such a way to accomplish required functions [57]. This aligns well with the FBS transition of creating expected behavior from functions.

- **PDR** – corresponds to the transition from Be to S
  - PDR is meant to show that the detailed design approach for the system satisfies functions [57]. The Be to S transition is meant to achieve this as the expected behavior (low level functions) is used to derive the design.
• CDR – corresponds to the transition from S to D
  
  CDR is meant to show the total system design is complete, it meets requirements, and it is ready to be built or coded [57]. This is represented by the transition from structure to documentation. In order to make this transition, the design must be complete, which is met by completing the structure phase. The design must meet requirements, which is met by completing the Bs to Be comparison (part of the structure phase in the implemented FBS model). Also, the design must be ready to be created, which is represented by completing the documentation, this is where the design is handed off to manufacturing or coders to be created.

All agents start the simulation in the requirements state, R. The work to get to SRR begins with a hierarchical flow down of requirements. The agents are arranged into three tiers as described in section 5.2. The engineer at the top level provides information to engineers at the next level down. At this middle level the information is further refined into lower level functions, which are then provided to the teams at the lowest level to develop functions from. The work to get to SFR (or transition of Be) follows this same hierarchical construct as the agents go from F to Be. The work to get to PDR only consists of the lowest-level teams transitioning to S as they are the ones that develop the design. The final step involves the lowest level agents transitioning to D, which is also where rework may occur since the behavior of the structure of the design is compared with expected behavior and reformulations may occur. The waterfall simulation process is illustrated in Figure 7-4, where the subject state transition is highlighted in the FBS model. After all agents have transitioned to the documentation phase, the simulation ends.
An Agile simulation was developed in support of research question 2, objective 1. In the Agile simulation, the agents move through the FBS states per the scrum process. The top and middle level agents begin the process with a planning phase. To represent this planning phase, these agents must transition from the R state to the F state, representing the work to develop high-level requirements into functions and creation of the first tasks in the work backlog. From here the work is passed directly to the lower level teams to perform sprints.

Each of the teams at the lowest level perform sprints to develop functions within their subsystems. The goals of each sprint are to accomplish the work needed to move to the next state in the FBS diagram. Reformulations represent rework on the product that is performed according to the product backlog. The product backlog can be populated with fixes to the structure, behavior, or function depending on the type of reformulation.
Figure 7-5 outlines the scrum process and its ties to the agents in the simulation. The process starts with an initial planning phase where the top two tiers of agents must transition from R to F. After this transition, the lower level teams work through sprints to accomplish their work. The lower level teams go through sprints that can notionally last from 10 to 30 days with a daily scrum. At the end of the sprint, a product iteration is produced. The product backlog is then consulted to determine what features/fixes need to be put in the product. After all agents have transitioned to state D, the simulation ends as the documentation of the design has completed. Figure 7-6 shows the notional Agile process which starts with key agents going through initial system definition by going from the R to the F state. Lower level agents then develop functions in a serial manner by moving through all the FBS states.

![Figure 7-5: Agile Process Overview](image-url)
Figure 7-6: Notional Agile Design Process

7.5 Simulation Design

The simulation design was developed in support of research question 2, objective 1. The simulation was developed in MATLAB®, a proprietary program language. It was chosen due to the familiarity the author has with it and its inherent ease of analyzing output data and plot generation. The language includes a large library of functions that are useful for Monte-Carlo simulations, such as the multiplicative lagged Fibonacci generator [101] used for pseudo-random number generation. The multiplicative lagged Fibonacci generator is one of the native pseudo-random number generators available in MATLAB®. The multiplicative lagged Fibonacci generator was chosen as it supports multiple simultaneous random number streams, simplifying execution of the simulation as well as its performance for random number generation [101].

The agent model is implemented in a 5x5 array, which tracks the transition probabilities between the FBS states. The mean FBS performer and an array of the difference between the mean performer and the high side of the 90% confidence interval on transition times are used with the pseudo random number generator to create all the
individual agent models. This results in multiple arrays of FBS transition probabilities that range between the lowest and highest possible performing agents shown in Figure 6-12 and Figure 7-7. The function *FBS_Designator.m* is used to generate these agent models.

The agents are then assigned to teams in the hierarchy structure for the team shown in Figure 7-2 or Figure 7-3. Certain agents are assigned to be team leads, the overall program lead, or individual designers. These agents maintain their position in both the waterfall and Agile simulations, simulating the same team being used for each process.

### 7.5.1 Waterfall Simulation Design

For the waterfall simulation, all the agents perform the following steps in a sequential fashion (FBS state transition shown in parentheses):

- Requirements development (R->F)
- Function development (F-> Be)
- Preliminary design (Be->S)
- Detailed design (S->D)

To step through each of these steps, a random draw is used with the transition probabilities in the agent model to determine if the agent advances to the next state in the FBS matrix. The function *FBStrans.m* is used to compare the random draw to the agent’s transition probability. If the random draw is greater than the transition probability, this indicates that the agent has completed the activity needed to transition to the next state. If the random draw is less than the transition probability, the agent remains in the state it is in until simulation time advances and a new transition probability is created to check if the transition has occurred.

In order to move from one step to the next, all agents have to complete the corresponding state transition for all functions they are responsible for. After each time step advancement, agents that have not yet transitioned to the next state are tested to see if they advance with a new random draw. Once all agents have advanced, the simulation gathers metrics about the current step the design is at. These include total effort hours expended by all agents advancing the design, total time needed to complete the step, and
for the critical design phase, total time spent in rework. After all steps are complete these metrics are compiled into metrics for the entire simulation.

The simulation is repeated 10,000 times as a Monte Carlo simulation [102]. Data from all the simulations are compiled together to analyze the performance of the design team using waterfall processes. Average time needed to complete the design, average number of effort hours needed to complete the design, and average amount of time spent in rework are calculated from the compiled simulation metrics.

7.5.2 Agile Simulation Design

The Agile simulation starts with a planning phase where the program leader and team leaders (top and middle tier of the organization hierarchy) perform initial requirements development going from state R→ F. The agents advance through the FBS states in the same manner as is used in the waterfall simulation using the FBStrans.m function.

Next, each of the design teams (lowest level of the organizational hierarchy) perform Agile sprints. They move through the entire FBS model, starting at state R and ending at state D for each of the design functions they are responsible for in a serial manner. Each team works in parallel to one another without the need to synchronize work between them at major reviews like the waterfall team. Instead, when functions are ready, they are added to the product incrementally and released, eventually building a complete product.

Like the waterfall simulation, the Agile simulation is repeated 10,000 times as a Monte Carlo simulation. The same performance metrics are gathered during each iteration and compiled into metrics that span the 10,000 iterations. Averages of these metrics are also calculated.

After the Agile simulation completes, the waterfall and Agile simulation metrics are compared to each other. This is done by determining the ratio of average time to complete the design, ratio of average effort to complete the design, and ratio of time spent in rework.

7.6 Modeling and Simulation Assumptions

Any model and simulation consist of assumptions. The model and simulations used in this research, along with all others, are approximations of reality [103]. This
section consists of a list, in no particular order, of simplifying assumptions used in the software program and launch vehicle design simulations. The reasons for these assumptions are also listed. These assumptions were needed to make this particular effort tractable as representing all aspects of a large team of individuals developing a complex design in simulation is viewed as a nearly impossible task.

- Designers can be represented by the FBS model. The FBS model is an abstracted model of designers. While it does not represent all aspects of designers, it contains enough information to represent the states that the designers go through during the design process [3].
- The synthesis, analysis, and evaluation activities were combined into a single activity. This was necessary so that the FBS model could be represented by a first-order Markov process.
- The system being designed is unprecedented. The designers do not know a-priori the optimal design solution.
- Design teams work in parallel. Teams do not wait for other teams to perform their work before starting.
- Reformulations caused by design incompatibility are type I reformulations. It is assumed that these types of reformulations are caused by incompatibility in the structure of two different parts of the design.
- The simulation is designed assuming that the agents interact with one another through a model of the system they are developing. Communication between the agents is not modeled due to this assumption. Rather, agents learn of the design decisions and implications of those decisions on the system as soon as their peer agents learn them. Thus, information learned about the design of the system is given to agents with zero time lag.
- Team leaders (agents above the lowest level in the team hierarchy) do not contribute to design work. This was done to represent the roles of these leaders primarily in the planning of the design through requirements and function derivation.
Idle time is not modeled. It is assumed that agents that complete their work early have other projects they can work on and their idle time does not count towards the total number of effort hours needed to complete the design.

Agents understand the coupling in a design. When coupling forces redesigns of subsystems, the minimum number of subsystems are redesigned.

The software and launch vehicle systems are simple versions of these types of systems. This assumption was needed to ensure the simulation development effort was tractable as a fully defined development process for these large and complex systems is difficult to define and fully simulate. This makes the simulation results not necessarily representative of real-world performance outright. They are still considered valid for comparison purposes, which is the primary objective of this research.

### 7.7 Software Program Development Simulation

A software program development simulation was created to support research question 2, objective 4. The software design simulation was created following the design outline in section 7.5. Agents were arranged in teams per the hierarchy shown in Figure 7-2. Each team was assigned to develop a single module of a software program. The exact purpose of the software is not necessary to model so long as the complexity of the work for the software team is properly modeled.

Each of the agent teams has eight members. The teams each have a software sub-module to develop with equal complexity to one another. Each software sub-module has 10 functions that require requirements development, design, and evaluation. This means that designs must visit all FBS states for each function to adequately design it. In total, the software program has 120 functions that need to be developed. The total number of individuals represented by the simulation is 101.

The sub-modules are assumed to have a low amount of coupling [104], and the design of one submodule does not affect other submodules. Coupling is the implicit dependency of two pieces of the system on one another [105]. Developing computer
programs with a low amount of coupling is standard design practice for software systems [105]. This is not true for the launch vehicle development simulation.

### 7.8 Launch Vehicle Development Simulation

A launch vehicle development simulation was developed to support research question 2, objective 3. The launch vehicle design simulation was built using the same simulation structure and techniques as the software program development simulation, but it contains an aspect that made it more complex than the software program development simulation. This is the coupling of design choices between design teams. The software program development simulation assumed a low amount of coupling between modules of the software program, a reasonable assumption given standard software design practices. This same assumption is not true of a launch vehicle, where there is significant coupling between subsystems in the design. To simulate this, design variables were added to the simulation to represent the design choices of the agents. The choices of these variables can introduce the need to reformulate a design that is working fine by itself but does not integrate with the rest of the launch vehicle. A simulation of the ascent of a launch vehicle was created to determine what combinations of design variables resulted in a successful design. This section details the effort that went into adding these capabilities to the simulation.

The random draws that govern the agents transitioning through the FBS model are used to pick design variables related to the launch vehicle. This is performed to model the highly-coupled nature of a launch vehicle design. As part of the evaluation step of the FBS model, the design structure is compared, not only to its expected behavior, but how it impacts the expected behavior of the system given the design choices from the other design teams. This provides some insight into how a highly coupled design can impact the work of design teams. The concept is illustrated in Figure 7-7, which shows two design variables represented as two random variables. The intersection of the two variables that creates a valid design, which for a launch vehicle is a design that can take a designated payload to orbit, is shown in blue as the “D” space. Type I reformulations occur in space outside of the valid design space and require rework to reformulate the design to the point that it could be a valid design. In the launch vehicle design simulation,
a total of 9 design variables are needed to align to a space where a design is valid. The effort to develop this valid design space is described below.

![Design Variables Diagram](image)

**Figure 7-7: Illustration of how design variables are tied to FBS states**

Nine total design variables are used to design the launch vehicle. They belong to different engineering disciplines represented by different agents in the FBS model:

- Stage 1 Thrust – determined by the Stage 1 rocket engine team
- Stage 1 Propellant mass – determined by the Stage 1 tank team
- Stage 1 Structure mass – determined by the Stage 1 structural team
- Stage 1 Diameter – determined by the Stage 1 mechanical team
- Stage 2 Thrust – determined by the Stage 2 rocket engine team
- Stage 2 Propellant mass – determined by the Stage 2 tank team
- Stage 2 Structure mass – determined by the Stage 2 structural team
- Stage 2 Diameter – determined by the Stage 2 mechanical team
- Payload Mass – determined by the systems analysis team

The teams are broken out by major subsystems of the launch vehicle. Figure 7-8 shows an $N^2$ diagram of the teams and the two aspects of the simulation: trajectory and structural evaluation. Each team is responsible for a single design variable and each receives feedback from one or both simulations. This creates a learning feedback loop for the design where the designers can improve their designs if orbit isn’t achieved or if there are negative margins of safety.
Figure 7-8: N² diagram of launch vehicle design team simulation

7.8.1 Launch Vehicle Physics model

The launch vehicle is a two-stage vehicle with a payload it is trying to get into orbit around the Earth. The design is represented simply, with a minimal number of subsystems. This abstracted design will be used for comparison purposes for evaluating systems engineering methodologies and not necessarily a predictor of actual design team performance. A notional depiction of the launch vehicle is shown in Figure 7-9. Figure 7-10 depicts a free body diagram of the launch vehicle during ascent. The velocity vector, \( V \), is shown with a significant angle to the vehicle centerline for illustration purposes. The simulation assumes this angle is small, which is true for all but a short period of the trajectory, during the time after stage 1 pitch over. Since this time is short and the vehicle velocity during this time is relatively small and drag forces are low, the assumption is considered valid for this first-order simulation. The forces on the launch vehicle are modeled as follows:

- **T** – Thrust – the force of thrust from either the stage 1 or stage 2 rocket engine. This is a constant force during the burning of the stage until the stage’s propellant is expended. After stage 1’s propellant is expended, stage 1 is jettisoned and stage 2 thrusting begins. After stage 2 propellant is expended, the vehicle’s trajectory is set, placing the vehicle into orbit if the design is successful.
• **G – Gravity** – the force of gravity is modeled using a simple point-mass constant acceleration assumption with no regards for vehicle altitude, i.e. a constant 9.81 m/s\(^2\). Higher order gravity terms, such as J2, are not modeled due to the simplicity of the simulation.

• **D – Drag** – the force on the vehicle due to atmospheric drag. This is represented by the coefficient of drag of the rocket and the vehicle frontal area in the vehicle design. The coefficient of drag is assumed at a constant value of 0.7 [106]. The force of drag varies with the atmospheric density (per the 1976 standard atmosphere model) and velocity of the vehicle. The force acts in the opposite direction of the vehicle’s velocity vector.

![Figure 7-9: Notional Depiction of Launch Vehicle](image-url)
All forces act through the center of mass of the vehicle. The simulation contains equations for 3 degrees of freedom of the vehicle: vertical motion, horizontal motion, and out of plane motion. In the simulation, out-of-plane motion is neglected since the vehicle is stationed on the equator and achieves an orbit with 0 degrees of inclination. This simplifies modeling to only 2 dimensions, or a perifocal analysis, of vehicle motion.

The vehicle starts on the equator of a spherical Earth in a vertical position before launch. At the beginning of the simulation the vehicle ascends for a short vertical rise under power from the first stage lasting 10 seconds. At the end of the vertical rise the vehicle pitches over at a constant rate for 5 seconds and then begins a gravity turn [107]. After leaving the atmosphere, it begins to further pitch over to insert its payload into an orbit. This pitch maneuver is performed at a constant rate for remainder of powered flight. The fairing surrounding the payload is jettisoned upon leaving the sensible atmosphere as defined by the 1976 standard atmosphere [106]. When stage 1 propellant is exhausted, stage 1 is jettisoned and the second stage begins firing. When the second stage propellant is spent, the second stage and the payload continue to coast until a stopping point is reached in the simulation.

The simulation checks the rocket through 10,000 seconds of flight and stops with a failure if the payload impacts the surface of the Earth. Testing done with the simulation
showed that for the largest vehicle simulation, 10,000 seconds was sufficient time to prove that the vehicle either reached orbit or impacted the Earth.

Maximum dynamic pressure is calculated in the simulation and is used to determine structural capability of the vehicle against maximum dynamic pressure loads. There are other load cases of interest for a launch vehicle, but buckling under maximum dynamic pressure is typically one of the most stressing load cases and is used here as a first order check on vehicle buckling capability [108]. The equations from NASA-SP-8007 are used to determine the buckling capability of the vehicle design and compared against the buckling load imparted to the vehicle during maximum dynamic pressure [108]. This is expressed as the margin of safety, as defined in Equation 5-1.

\[
Margin\ of\ Safety = \frac{Buckling\ Capability}{Buckling\ Load} - 1
\]

Equation 5-1: Margin of Safety

This simulation was run with four values for each of the nine design variables listed above. This gives 262,144 possible vehicle configurations. Of these only 1,517 were found that reached orbit with positive margin of safety. This then formed a response surface that is used as a Kringing model [109]. This Kringing model is used in the waterfall and Agile design simulations to represent the choices of the design team based on random draws during the design simulation.

The coupled nature of the design choices is also modeled. This is accomplished by forcing type I reformulations when design coupling forces a change to a part of the design. If the design variables chosen by a team of agents, when coupled with the variables chosen by the other teams, results in a design that does not reach orbit with positive margins of safety, a Type I reformulation is forced on one of the teams. This simulates that while the subsystem design was viable, when coupled with the rest of the system, the system did not meet objectives. It is assumed that the agents understand the coupling and react to design changes driven by the coupling in such a way as to minimize the number of subsystems that need to be redesigned.

Figure 7-11 shows an example of this process for the first 3 of the 9 design variables in the simulation. The first agent team selects the payload mass. Their design
selection is compared to the designs in the Kringing model. If their selected variable is available in at least one of the viable designs, they are allowed to keep their design point (stay at state D in the FBS model) and the next team’s results are checked. If their design variable is not in at least one of the viable designs, then they must do a Type I reformulation to modify their design until they select a design variable that results in a viable design. The next team then repeats the process, but when they check their design variable, the team must also consider the previously chosen design variables. So, the check is that the current team’s selected design variable along with all previously identified design variables must all be part of at least one viable design point. If not, then the current team performs Type I reformulations until the design variable they choose, along with the previously selected design variables, are contained in at least one viable design point.
Figure 7-11: Design Variable Selection for the Coupled Launch Vehicle Design (3 of 9 design variables shown)

The launch vehicle simulation concludes with the 9 design teams choosing design variables that, perhaps after many reformulations, results in a design that successfully injects a payload into orbit. Metrics are gathered by the simulation to determine how long it took to develop the design, how many effort-hours were expended, and how much time was spent doing reformulations.
7.9 Simulation Execution

Each of the two simulation cases were executed in a similar manner. Each simulation performs 10,000 Monte-Carlo iterations each of the waterfall and Agile design approach. These simulations are run in accordance with the design described in sections 7.5.1 and 7.5.2. At the end of each of the simulations, metrics are gathered and tallied. The total time needed to develop the design, the total number of effort-hours expended to develop the design, and the time spent doing rework are all calculated for each of the simulations. The ratio of the average of these metrics between the Agile and waterfall simulations is also calculated so as a way to compare the two approaches.

7.10 Summary

The work documented in this chapter shows the design of the two agent-based simulation cases used to determine the benefits and drawbacks of Agile systems engineering processes compared to the waterfall process. The structure of the agent teams, the systems engineering processes, and the design of the simulations are defined in the chapter. The source code for the simulations and output data from the simulations are available online at https://github.com/monza66/Waterfall-Agile_ABS. This work defines the simulation that is used to answer research question 2.
CHAPTER 8

8.1 SIMULATION DATA ANALYSIS AND RESULTS

This chapter provides the results from the launch vehicle simulation and software development simulation. The methods used to analyze the data are presented in this chapter as well. The results are reviewed in order to answer the second research question and determine the applicability of Agile methods for systems engineering. This chapter specifically addresses objectives 5, 6, and 7 of the second research question. Building on the work of chapters 4, 5, 6, and 7, the results presented in this chapter provide the answer to research question 2. Figure 8-1 provides an overview of the chapter.
8.2 Data Analysis Methods

The primary method used to analyze the simulation data is statistical analysis of the results of the Monte Carlo simulation. The results are tracked through three metrics, time to complete the design effort, effort-hours expended completing the design effort, and time spent in rework during the design effort. Histograms are a popular way to represent Monte-Carlo simulation data and are used in this research [110]. The mean values for the team performance metrics are primarily used for comparison. The standard deviation of the team performance metrics are also calculated to examine the variability in performance of the different processes.

8.3 Software Development Simulation Results Analysis

The software development simulation yielded results for using a waterfall and an Agile development process. Figure 8-2 represents the time needed to complete the design when using a waterfall process. The average amount of time needed to complete the
design is 2,012 working hours, which is close to a year worth of effort assuming a 40-hour work week. The standard deviation of this parameter is 337 hours, representing around 2 months of time.

Figure 8-2: Histogram of time to complete a software program design using **waterfall processes**

Figure 8-3 represents the time needed to complete the software program design when using an Agile process. The histogram is represented in the same scale as Figure 8-2 for comparison purposes. The average amount of time to complete a design is 765 hours, or a little over 4 months with a 40-hour workweek. The standard deviation is 157 hours, slightly less than a month of effort. The mean is much less than that of the waterfall approach and the standard deviation is also slightly less than half of the waterfall approach.
Figure 8-3: Histogram of time to complete a software program design using Agile processes

Next, effort hours, defined as an hour of labor expended by a designer, are examined. Figure 8-4 shows the effort-hours expended in completing the design of the software program using waterfall processes. The mean of the data is 37,038 hours. The standard deviation is 1,225 hours. The histogram and standard deviation show that this metric is quite consistent across the 10,000 Monte-Carlo runs of the simulation with little spread in the data. This demonstrates that the total time needed to complete the design is consistent with randomized team performance capabilities. An Anderson-Darling test of the data at a 5% significance level does not reject the null hypothesis that the data is from a normal distribution.
Figure 8-4: Histogram of effort hours needed to complete a software program design using waterfall processes

Figure 8-5 shows the effort hours needed to complete the software program design using Agile processes. The mean of the data is 21,518 hours with a standard deviation of 1,091 hours. Like, the waterfall simulation, the Agile simulation showed a low amount of variability in the number of effort hours needed to complete the design. The mean of the Agile simulation is lower than the waterfall simulation by 42%. An Anderson-Darling test of the data at a 5% significance level does not reject the null hypothesis that the data is from a normal distribution.
Figure 8-5: Histogram of effort hours needed to complete a software program design using Agile processes

Rework is tracked in the simulation as the amount of time spent performing tasks that started due to any of the three types of reformulations. Figure 8-6 shows the amount of rework required to complete the software program design and how it varies across 10,000 Monte-Carlo runs of the software design simulation model. The runs result in a mean of 1,693 hours spent doing rework with a 332-hour standard deviation when using a waterfall process.
Figure 8-6: Histogram of time spent doing rework for a software program design using waterfall processes

Figure 8-7 shows the amount of time spent doing rework when using an Agile process. There is a substantial reduction in time spent doing rework with the Agile process. The mean across the 10,000 simulations is 724 hours and the standard deviation is 157 hours. This represents a reduction in time spent on fixing defects of 57% on average.
8.3.1 Software Development Simulation Results Comparison

Table 8-1 summarizes the results of the waterfall and Agile simulations of the software program design. The “expected result” column contains validation data explained in section 9.6. The difference between the means of the waterfall and Agile simulation data closely match the expected results from the dataset used to validate the simulation. The standard deviation of the effort hours expended is similar between the two simulations, showing that the amount of effort required to complete the design is fairly consistent between the waterfall and Agile processes. The standard deviation of the time to complete the design and the standard deviation of time in rework are both reduced significantly when using the Agile process. The simulation supports the claims of
benefits from using the Agile process on uncoupled systems, like the uncoupled software system modeled in this simulation.

Table 8-1: Summary of software design simulation metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Waterfall Simulation</th>
<th>Agile Simulation</th>
<th>Difference in Waterfall and Agile Means</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort-Hours</td>
<td>Mean: 37,038 hours</td>
<td>Mean: 21,518 hours</td>
<td>42% less hours in Agile mean</td>
<td>36%- 50% less</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation: 1,225 hours</td>
<td>Standard Deviation: 1,091 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total time expended</td>
<td>Mean: 2,012 hours</td>
<td>Mean: 765 hours</td>
<td>62% less hours in Agile mean</td>
<td>30% - 70% less</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation: 337 hours</td>
<td>Standard Deviation: 157 hours</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time spent in rework</td>
<td>Mean: 1,693 hours</td>
<td>Mean: 724 hours</td>
<td>57% less hours in Agile mean</td>
<td>~50% Less</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation: 332 hours</td>
<td>Standard Deviation: 157 hours</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.4 Launch Vehicle Development Simulation Results Analysis

The launch vehicle development simulation produced results from 10,000 Monte-Carlo runs for both waterfall and Agile processes. The time needed to complete the launch vehicle design using waterfall development processes is shown in Figure 8-8. The mean of this data is 1,836 hours, or roughly 10.5 months of time to complete the design of a simple launch vehicle. The standard deviation of the data is 405 hours.
When using an Agile process, the time to complete the design of the simple launch vehicle has a mean 1,608 hours with a standard deviation of 407 hours. There is a 12% reduction in the mean when using an Agile process. The standard deviation of the two processes is very similar. A histogram of the time of complete the launch vehicle design when using the Agile process is shown in Figure 8-9.
Figure 8-9: Histogram of time to complete a launch vehicle design using an Agile process

The effort-hours, or total number of hours expended by all the agents in the simulation, to complete the design when using a waterfall process across the 10,000 Monte-Carlo runs is shown in Figure 8-10. The data has a mean of 22,846 hours and a relatively low standard deviation of 1,693 hours. This maintains the characteristic seen in the software design simulation where the amount of effort hours needed to complete the launch vehicle design is relatively consistent between the Monte-Carlo runs.
Figure 8-10: Histogram of effort hours needed to complete a launch vehicle design using a waterfall process

Figure 8-11 shows a histogram of the effort hours needed to complete the simple launch vehicle design when using an Agile process. The data has a mean of 22,672 hours and a standard deviation of 1,724 hours. This data also continues the trend seen in the software simulation, where the effort-hours needed to complete the design do not show large variations across the 10,000 Monte-Carlo runs. The Agile process shows a modest 1% decrease in the mean effort-hours needed to complete the design when compared to the waterfall process. This represents only about 1/10 of a standard deviation, illustrating that the difference in effort is not significant.
Figure 8-11: Histogram of effort hours needed to complete a launch vehicle design using an Agile process

Rework for the launch vehicle design effort was tracked in the simulation as the effort hours expended doing design work related to type I, II, or III reformulations. It is tracked the same way as in the software program design simulation. Figure 8-12 shows the amount of time spent doing rework when using a waterfall process. This data has a mean of 1,519 hours with a standard deviation of 402 hours.
Figure 8-12: Histogram of time spent doing rework for a launch vehicle design using waterfall processes

Figure 8-13 shows the amount of time spent doing rework when using an Agile process. The mean of the data is 1,569 hours and the standard deviation of the data is 407 hours. This represents a 3% increase in the amount of effort expended on rework when using an Agile process.
8.4.1 Launch Vehicle Development Simulation Results Comparison

Table 5-2 shows a summary of the data from the launch vehicle design simulation. The benefits seen in the software program design simulation when using Agile over waterfall processes are not as substantial designing a highly coupled system. The effort hours and rework are very similar between the two efforts, with only a 1%-3% difference between the Agile and waterfall metrics, which, being within 1/10 of a standard deviation is not viewed as being significant. Time to complete the design does show a positive benefit, but only 12% better vs. 62% in the software design simulation.

Figure 8-13: Histogram of time spent doing rework for a launch vehicle design using an Agile process
Table 8-2: Summary of launch vehicle design simulation metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Waterfall Simulation</th>
<th>Agile Simulation</th>
<th>Difference in Waterfall and Agile Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort-Hours</td>
<td>Mean: 22,846 hours</td>
<td>Mean: 22,672 hours</td>
<td>1% less hours in Agile mean</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation: 1,693 hours</td>
<td>Standard Deviation: 1,724 hours</td>
<td></td>
</tr>
<tr>
<td>Total time expended</td>
<td>Mean: 1,836 hours</td>
<td>Mean: 1,608 hours</td>
<td>12% less hours in Agile mean</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation: 405 hours</td>
<td>Standard Deviation: 407 hours</td>
<td></td>
</tr>
<tr>
<td>Time spent in rework</td>
<td>Mean: 1,519 hours</td>
<td>Mean: 1,569 hours</td>
<td>3% more hours in Agile mean</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation: 402 hours</td>
<td>Standard Deviation: 407 hours</td>
<td></td>
</tr>
</tbody>
</table>

8.5 Summary

The process methodology between the software and launch vehicle design simulations is the same. The primary difference between the launch vehicle and software simulation is that the launch vehicle design simulation contains significant coupling between teams, which is not present in the software design simulation. Design coupling was found to create a significant amount of rework in the design. By forcing the simulation to ignore design coupling, rework was reduced by approximately 60% in both waterfall and Agile processes. The design coupling led to individual teams resolving their internal rework to create a workable subsystem design that, when integrated with other parts of the launch vehicle, resulted in a deficient system design. This tended to force teams to rework a design that was functional and spawned a significant amount of effort.

The results from the software simulation showed that the modeling and simulation approach matched expected data very well, showing validity of the simulation. It also shows that Agile methods offer significant benefits over waterfall methods for software development. The launch vehicle simulation extrapolates from this methodology to
simulate a highly coupled design. It shows that, for a highly coupled design, benefits from an Agile methodology may not be as substantial as seen on software projects. The simulation results show no significant benefits or drawbacks for Agile against waterfall for highly coupled systems such as a launch vehicle. The launch vehicle simulation shows the usefulness of the simulation for studying the effects of different engineering processes. Since there is little research on the use of Agile processes for large, coupled, hardware intensive systems, such as a launch vehicle, it is not possible to validate the results of the launch vehicle simulation. The results of the software simulation do provide some confidence in the overall methodology, showing that using Agile methods on highly coupled systems may require special provisions to account for the coupling in the design.
9.1 Overview

Verification and validation of the simulations was performed in support of research question 2. This was done to first, verify that the simulation was performing as intended, and second, to validate that the simulation output was representative of real design teams. This chapter covers the verification work and the four validation efforts. These consist of a review of the validation of the underlying FBS model, a validation of the transition parameters used in the FBS model, a face validation using a survey of subjective aspects of the simulation, and finally, a validation of simulation predictions against published case studies. Figure 9-1 provides an overview of the chapter and how it support verification and validation of the modeling and simulation effort needed to answer research question 2.
9.2 Verification

The simulations were verified through several steps. First, the code was debugged using the MATLAB® built-in debugger. Second, the simulations were tested using a simplified FBS transition matrix. The matrix was set to equal probability of transition between states and then the state transition times were examined. The first passage time on average should be the reciprocal of the probability of transition [100]. For example, in one case, the probability of going from R to F was set at 0.5. With 100 samples run, the average first passage time for the 10 agents at the top and middle levels was 2.051, very close to the theoretical answer of 2.0. Further checks of the simulation were performed to verify proper operation, including checking other first passage times with the notional FBS transition matrix and checks of the routines used to compute outputs of the program.

The number of Monte-Carlo runs was examined with 100 and 10,000 runs being performed with both simulations. The results from the 100-run simulation were found to be within 5% of the 10,000-run simulation. This provided evidence that 10,000 runs were sufficient to develop a consistent result as variation in results was not high between the
100 and 10,000-run simulations. Since data from the 10,000-run simulations was available, this data was used in the analysis. The model and simulation were validated using predictive validation [84] in four ways, described in the subsections below.

9.3 FBS Model Validation

The underlying FBS model is widely accepted in the systems engineering community. The model has been through empirical studies to confirm that it is a valid model of engineering processes as shown in [23], [48], [49] and discussed in section 2.6.

9.4 Model Parameter Validation

Model parameters are based on data that traces to real-world data from engineers. The model parameters were developed as described in section 6.1. The calibration effort was shown to correlate well with expectations, showing that the calibrated FBS model behaved as-expected when calibrated with the real-world data.

9.5 Face Validation of Simulation Characteristics

Face validation [84] is used to validate aspects of the simulation that cannot be validated through other methods. These consist of several subjective aspects of the simulation that are of interest for simulation validity and drawing conclusions from results. Ten aspects of the simulation were subjected to face validation. They consist of:

1. The amount of rework seen in the simulations
2. Whether or not there was evidence of unusually high productivity, or positive emergent behavior in the software design simulation
3. Whether or not there was evidence of unusually high productivity, or positive emergent behavior in the launch vehicle simulation
4. Whether or not there was evidence of unusually high rework, or negative emergent behavior
5. How well the calibrated FBS model captured the variability in individual engineers
6. How well the simulation implemented the waterfall process
7. How well the simulation implemented the Scrum process
8. If the size and composition of the software design team was appropriate
9. If the size and composition of the launch vehicle design team was appropriate

10. How well the FBS model was mechanized in the simulation

The face validation was performed through a survey of Systems Engineering Subject Matter Experts (SMEs) with experience in both waterfall and Agile engineering processes. The SMEs were recruited from the American Institute of Aeronautics and Astronautics (AIAA) Systems Engineering Technical Committee (SETC). The survey was built using Qualtrics™ software and the survey methodology was reviewed and approved by the University of Alabama in Huntsville Institutional Review Board (IRB). The survey is provided in APPENDIX 1: Face Validation Survey.

The data from the survey was assessed by looking at the median, mode, and range as the set of descriptive statistics for the data. The top two box score [111] is used to assess whether or not the majority of the SMEs agree that each question is valid or not. There are 4 possible results for each question based on the box scores:

- Validated – a majority of SMEs agree the aspect of the simulation is valid
- Incorrect – a majority of the SMEs agree the aspect of the simulation is not correct as stated
- Inconclusive – no majority position. The validity of the aspect of the simulation is unknown.
- Indifferent – a majority of the SMEs select the indifference response of 3.

The validity of the aspect of the simulation is unknown.

The response rate to the survey was lower than anticipated. Of the 53 participants invited to take the survey, only 4 completed the survey with two more completing part of the survey. The tally of responses and results of the face validation are shown below in Table 9-1.

<table>
<thead>
<tr>
<th>Count of Scores for Each Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question / Score</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
The results for each question, including a discussion on some of the written comments from the SMEs, are discussed below. The question is first stated then a discussion on results.

1. Rate how well rework (called reformulations) in the simulation matches expectations for the types of system developments simulated.
   - There was not a clear majority opinion from the SMEs. Some SME comments indicated that the results were well in line with expectations, where others felt that they were not. One SME that left a low rating stated that there wasn’t enough data to draw a definitive conclusions. Given that there was no clear majority consensus from the SMEs, the validation of question 1 is considered inconclusive.

2. Rate how well the simulation data shows evidence of positive emergent behavior (unexpectedly high productivity) in the design team
   - The majority SME opinion was that there was evidence of positive emergent behavior. Comments indicated that the amount of productivity seen in the Agile approach for software development was more than would be expected.

3. Rate how well the simulation data shows evidence of positive emergent behavior (unexpectedly high productivity) in the design team
   - The majority of SMEs selected the indifference option. One SME was surprised that Agile didn’t offer more benefits and several remarked that the data between Agile and waterfall were more similar than expected. It was expected that the majority of SMEs would have selected the option 1 or 2 since Agile did not appear to offer benefits over waterfall for the launch vehicle simulation.
4. Rate how well the simulation data shows evidence of negative emergent behavior (unexpectedly high amounts of rework) in the design team
   - There was not a clear majority SME consensus, leaving the validation inconclusive. Two of the SMEs felt they needed additional data to make a determination. One felt it was very clear that the rework was typical and negative emergent behavior was not present.

5. Rate how well the simulation numerically represents the variable performance of individual designers
   - This question asked if the FBS model captured variability in individual performance properly. There was not a clear majority SME option. One of the SMEs that gave a low rating felt that the high performer was very representative, but the low performer was unrealistically slow in transitioning between the FBS steps. The only other SME to comment felt that the FBS transition probabilities were reasonable.

6. Rate how well the above simulation represents the process followed
   - The majority of the SMEs felt that the simulation’s implementation of the waterfall process was reasonable.

7. Rate how well the above simulation represents the process followed
   - This question dealt with the implementation of the Scrum process in the simulation and was considered critical for validation of the simulation. The majority of the SMEs felt that the simulation’s implementation of the Agile process was reasonable.

8. Rate how representative the size and composition of the development teams used in the simulation are
   - The subject of this question was whether or not the size and structure of the software design team was reasonable. The SMEs did not reach a majority consensus in the summary of their responses. Responses by some SMEs indicated that teams may
have fewer or more members than the 8 used in the simulation and recommended by Agile practices.

9. Rate how representative the size and composition of the development teams used in the simulation are
   - This question dealt with the validity of the size and scope of the launch vehicle design team. The SMEs did not reach a majority consensus. One SME indicated in their written response that the team size and structure was reasonable while another disagreed and suggested a different type of structure.

10. Rate how well the modification to the FBS model preserves the original purpose of FBS model
   - This question asked about the validity of the mechanization of the FBS model in the simulation. Specifically about collapsing the Bs state and analysis process into the synthesis process. The majority opinion from the SMEs indicated indifference on the validity of the approach. One SME took issue with the FBS model itself in their comments, which was outside the scope of the question.

While the majority of the survey questions resulted in an inconclusive result on validating the more subjective aspects of the simulation, the two most important face validation questions were validated. This was the implementation of the waterfall and Agile processes in the simulation (questions 6 and 7). This validation was key to ensuring that the processes were properly simulated and that the simulation was creating valid outputs. The response to question 2 indicates that when switching to Agile, the software design simulation shows evidence of positive emergent behavior. This is a characteristic of the behavior of complex systems and shows that the simulation implements the Scrum process in a manner that creates positive emergent behavior. This supports the claim that Scrum results in positive emergent behavior when used for software system development. Creating a shorter survey that only addressed the key questions would likely have been helpful in obtaining additional responses. A worst-case survey time of 60 minutes was advertised to the subjects, which may have led to many of them not wanting to take the survey due to the time commitment.
9.6 Validation of Simulation Output against Case Studies

Finally, published literature on the benefits of using Agile scrum over waterfall processes for software development are used to determine if the simulation output is as expected. In this literature, case studies of projects using Agile compare productivity to traditional waterfall practices. These studies tend to define productivity as the amount of work that an individual can accomplish in an hour. Productivity gains tend to be 36% to 50% in most cases, with outlier examples showing gains as high as 600% and other outliers showing a productivity loss [24], [27], [31], [67]. Other metrics, with less case studies for support, include a 30-70% faster time to market, and an approximate 50% reduction in defects [27]. Table 9-2 below examines the simulation results against these metrics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Expected Result</th>
<th>Software Program Simulation Result</th>
<th>Launch Vehicle Simulation Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productivity</td>
<td>36%-50% gain</td>
<td>42% Gain</td>
<td>1% Gain</td>
</tr>
<tr>
<td>Time to Market</td>
<td>30%-70% Faster</td>
<td>62% Faster</td>
<td>12% Faster</td>
</tr>
<tr>
<td>Reduction in Defects</td>
<td>~50% Less</td>
<td>57% Less</td>
<td>3% More</td>
</tr>
</tbody>
</table>

The software program design simulation showed a productivity gain of 42% when using Agile. This is well within the expected range and since all four of the case studies cited above are for software development efforts, the case study data shows that the simulation accurately predicts the productivity gain software development teams can see when using Agile. The time to market, or overall length of time needed to develop the design, was 62% faster when using Agile, agreeing well with the published results of 30%-70%. Finally, defects were 57% less using the Agile process, close to the 50%-+ (no defined range in the cited case studies) expected. Overall, the software program design simulation produced differences in results between the waterfall and Agile simulations that agreed well with all the metrics cited for comparison.
The launch vehicle simulation results are outside of the expected range when compared to the metrics. This is not an unexpected result. All the available case studies to draw metrics from use software development programs as their basis. An effort comparable to a launch vehicle design was not found in any published case study on the benefits of switching to Agile. Thus, the comparisons in Table 9-2 are not necessarily valid for the launch vehicle design simulation. Given the positive validation results for the software design simulation, and the fact that the same methodology was used for the launch vehicle design simulation, the results from the launch vehicle design simulation are viewed as an extrapolation of the validated software design simulation results. Future case studies showing the productivity, time to market, and defect rate change when going from waterfall to Agile processes in the development of complex systems would be required to fully validate the simulation. Until such data is available, the results should be noted to be an extrapolation from a validated simulation.

9.7 Summary

Chapter 9 presents the verification and validation efforts showing the work performed to verify the simulations work properly and validate that they are an acceptable approximation of reality. Quantitative validation of the simulation results show results are within the expected range. Face validation was attempted to validate subjective aspects of the simulation and while critical aspects of the simulation were shown to be valid, many aspects were not able to be validated with the face validation data due to insufficient data or indifference among the SMEs that performed the face validation.
Chapter 10

SUMMARY, CONCLUSIONS, AND FUTURE WORK

10.1 Overview

This chapter contains the final conclusions of the research and areas where future work can further this research. The two research questions are discussed with associated future work on how to simulate design teams, and how Scrum could be used for systems engineering.

10.2 Conclusions

The two questions related to this research are stated again below:

1. Do theories exist that support the use of the Agile Scrum framework for systems engineering?

2. What benefits does the Agile Scrum framework have over waterfall processes? In what ways does the Agile Scrum framework fall short of waterfall processes?

For the first question, chapter 4 explored the history of Scrum and found evidence it was built around principles from complex system science and decision analysis. The tie between Scrum and these theories is through analogy. This means that the principles observed in these theories are applied in Scrum, in concept, and a strong mathematical link does not exist. It was determined that theories exist that support the use of Agile Scrum for systems engineering, but the support was only through these analogies. No strong mathematical tie was found.

To answer the second question, a modeling and simulation approach for design teams was created. Chapter 5 details the initial work to develop a proof-of-concept for the simulation. Chapter 6 discusses how the FBS model was calibrated for use in the simulation. Chapter 7 outlines the modeling and simulation approach used to make a simulation of a software program design team that produced data showing the benefits of
using Agile Scrum that matched published data for software design teams well. The modeling approach was extended to simulate a launch vehicle design team, simulating the work to create a highly coupled system. Chapter 8 discussed results from these simulations and chapter 9 reviewed the verification and validation work done on the simulations.

This work showed that agent-based modeling with the FBS framework is a valid approach to investigating the applicability of specific systems engineering methods. The results of the simulation of the software design team fell within the expected range gathered from case studies. The most critical of the subjective aspects of the simulation, the Agile and waterfall processes, were validated through a majority opinion of SMEs. Further validation is desired to strengthen these claims, but the results from the agent-based simulation show good agreement with the expected results.

For the software design simulation, it was found that the Agile Scrum framework provides positive benefits by reducing total effort-hours expended, reducing time to complete the design, and reducing the amount of time spent in rework. This assumes that the scrum teams can be mostly independent of one another, as is part of the Scrum-of-Scrums construct used in Agile Scrum. For the launch vehicle design simulation, it was found that the number of effort hours expended, and the time spent in rework were mostly unchanged between the waterfall and Agile processes. Agile processes did show a lower time to complete the design, but the benefit was substantially less than seen in the software design simulation. The launch vehicle simulation suggests that systems that have significant coupling in the design may not see benefits to the level the software community has seen. The systems with tight design coupling may require a modified Scrum process to realize any benefits. It should also be noted that the research found that Agile processes are not always beneficial as design efforts that are slight alterations of existing products may be more efficient with a waterfall process [4].

10.3 Future Work

This section identifies several areas of further research that could be performed to build on the work in this dissertation. These additional areas of work could help to understand how Agile and waterfall processes perform to build a stronger validate case for the modeling techniques in this research. The additional research would elaborate the
basic FBS model used in the agent-based simulations. Finally, additional research is proposed that could deepen understanding of how Scrum relates to systems engineering theories. These areas are explored in the subsections below.

### 10.3.1 Waterfall to Agile Transition for Development of a Complex, Coupled system

The validation of the modeling and simulation approach described in section 9.6 was based on teams transitioning from waterfall to Agile when developing software. Several case studies were available to examine how the change in engineering process affected the productivity of the design team. Comparable studies for a complex, coupled system were not found. These types of studies are needed to fully validate the results of the launch vehicle design simulation.

While the launch vehicle simulation was based on the validated software simulation, it added the aspect of design coupling. Design coupling was found to significantly alter the results of the simulation, where the software simulation showed significant benefits from using Agile, the launch vehicle simulation had mixed results with only time to complete the design showing improvement. Case studies from real-world projects involving complex and coupled systems would be useful to fully validate the design coupling methodology used in the launch vehicle design simulation.

### 10.3.2 Situated FBS model

Gero and Kannengiesser have extended the FBS model to a “situated” FBS model [50]. This is meant to account for the dynamic effects in the design process. These dynamic effects result from requirements not being perfectly defined. This extended model is meant to support modeling uncertainty in the design process.

The situated FBS model was created to better describe the interaction that designers have with the external world. It has added complexity to the FBS structure but allows for much richer explanation of how designers behave. The situated FBS model may add additional fidelity to the agent-based simulation approach used in this research. It does further complicate the model and does not have as much validation behind it, but it could offer a way to enhance a simulation of design teams.
10.3.3 Possible Relationships between Scrum and Other Systems Engineering Theories

While it can be shown that relationships exist between Scrum, complex system science, and decision analysis, there are many theories that may be useful to help explain Agile processes’ effectiveness or ineffectiveness in topics such as team interactions and knowledge generation. Theories such as organization theory, learning theory, and game theory may help build a foundation for when and how Agile development processes are effective. Figure 10-1 shows the relationships to theory established through this research and additional relationships that could be established in later work. Theories outside of complex system science exist with the potential to further support the use of the Scrum process. Further work could establish clear relationships to these theories, even if it means changes to the Scrum process, to further support the use of Agile Scrum for engineering efforts. In addition, these theories have relationships to other communities, outside of the complex system science community. Establishing these relationships would create a larger community support for an approach with (possibly) broad theoretical underpinnings. Some of the potential relationships to these other theories are shown in Figure 10-1 and are described below.

Organization theory is a collection of theories that attempt to study and predict how organizations of people will perform under different circumstances [112]. Organization theory may provide further support to Agile Scrum if it can be used to support how using the Agile Scrum process contributes to more efficient work. Specifically, organization theory could be used to understand the best composition of the Scrum team to enable the best system to be developed.
Figure 10-1: Identified and potential future relationships between Agile Scrum and theory. * denotes a relationship that was previously established.

Game theory studies mathematical models of interaction between rational decision makers [113]. As such, game theory could help provide a further mathematical foundation for studying rational agents performing engineering. This work could advance the theoretical basis that supports using Scrum to perform engineering work. Specifically, game theory could be used to understand how best to incentivize or develop “rules” for how individuals on a Scrum team interact. This would build on the decision analysis relationships previously identified to relate Scrum to rational decision interaction theories.

Learning theory, specifically Bayesian learning, could be beneficial in providing additional theoretical basis for Scrum. Bayesian learning uses data from past experience to make predictions about something being learned [97]. This allows uncertainty to be bounded and reduced at each step of learning. The sprint retrospective provides a forum for such learning, where the exact development process is refined as lessons from each sprint are incorporated. Learning theory may provide theoretical support to performing the sprint retrospective and its impact on refining the development process.
Theories originating in psychology and other social sciences may aid in establishing the basis for using Scrum for engineering. Research into these other areas of theory could help to bolster the theoretical foundation for Scrum.

10.3.4 FBS Model Calibration

Future work on this subject includes using the calibrated model in simulations of different engineering approaches and comparing the time and effort required to design a system to observe differences in the team’s performance when using different processes. The approach used in the dissertation examined differences in using traditional waterfall methodology and agile Scrum. This methodology has merit in being able to provide a cost-effective means of studying different ways for a team to execute a project.

The approach normalizes the input data to a single performer. This approach may not account for efficiencies gained by applying multiple individuals to a task or efficiencies gained by working a task part-time over a longer duration. The sensitivity of task loading on efficiency should be further examined to determine what, if any, effect it has on the FBS transition probabilities.

The approach uses a subjective analysis to group tasks into an FBS state transition category. This could be improved with a more rigorous approach such as verbal protocol analysis. The approach also assumes that task durations do not change when using alternate processes or when working on different systems. How engineering processes affect the duration of design tasks should be examined in more detail to determine the validity of this assumption. Different types of systems will obviously require a different amount of time to design between them. The exact variability between different types of systems needs to be better understood so that an FBS model with calibrated data is used appropriately when it is used to examine different efforts.

10.4 Summary

Chapter 10 concludes the dissertation and summarizes further work that can be done to expand on the modeling and simulation approach. This could involve using more elaborate models, such as the situated FBS model, gathering additional FBS model calibration information, and validating the launch vehicle design simulation with data from a complex and highly coupled system development. In addition, future work could be performed to find further theories that support the Agile Scrum process.
This work has made novel contributions to the modeling and simulation field by developing an agent-based simulation of designers that can be used to test the efficiency of using different systems engineering processes. It also has developed a method for calibrating FBS models with real-world data. In addition, this work has contributed to systems engineering by establishing links between Agile Scrum and theories as well as developing a method for using Agile Scrum with MBSE to perform systems engineering processes.
REFERENCES


[66] “Case Study – Scaled Agile Framework.”


[81] H. Kannan, C. L. Bloebaum, and B. L. Mesmer, “Incorporation of Coupling Strength Models in Decomposition Strategies for Value-Based MDO.”


APPENDIX 1: FACE VALIDATION SURVEY

Design Team Simulation Face Validation

CONSENT FORM Please be advised that you must be at least 18 years old to participate in this study.

PROCEDURE TO BE FOLLOWED IN THE STUDY: Participation in this study is completely voluntary. Once consent is given; you will review data from a simulation of a design team and fill out a questionnaire regarding the simulation and takeaways you gathered. The survey consists of scaled as well as open-ended questions. This session will take about 60 minutes.

DISCOMFORTS AND RISKS FROM PARTICIPATING IN THIS STUDY: There are no expected risks associated with your participation. You may experience light eyestrain from reading the survey and simulation data. There are no expected social, economic, or legal risks for participating in this study.

EXPECTED BENEFITS: Your responses will assist in assessing the validity of an agent-based model of a design team.

CONFIDENTIALITY OF RESULTS: No identifying information will be combined with your survey responses. Your responses will be tracked via a survey response number. The data from your survey will only be released to those individuals who are directly involved in the research and only using this number.

FREEDOM TO WITHDRAW: You are free to withdraw from the study at any time. You will not be penalized because of withdrawal in any form. If you agree to participate in our research, please click “I Agree” below.

This study was approved by the Institutional Review Board at UAH and will expire in one year from April 30, 2019.

☐ I Agree (1)
This survey examines the results of a simulation of design teams. The simulation is meant to represent the members of the design team. The simulation examines two design teams. One is tasked with developing a computer program. The other team is tasked with developing a space launch vehicle. In the simulation, each team attempts to develop their design using a waterfall process. A waterfall process is one in which team members develop the design all at once, in a serial fashion. It has large design review events where all designers synchronize their efforts. Also in the simulation, the design teams attempt to develop their design using an Agile process. In this process, the designers develop individual features of the design in a serial manner. Features are added one by one until the product is completed. The simulation compares the waterfall and Agile approaches. You will be presented with data from the simulation. The data is presented in histograms. The simulation is a Monte-Carlo simulation and the histograms represent the output from the simulation. You will be comparing the waterfall and Agile results and will be asked to provide your best assessment on if the simulation results align to your expectations of how design teams would perform when developing these types of systems. The first question is on rework encountered when creating a design. Rework occurs when an earlier part of the design process much be revisited since something in the design is found to be lacking.
The first question is on rework encountered when creating a design. Rework occurs when an earlier part of the design process must be revisited since something in the design is found to be lacking. You will be shown histograms from 10,000 Monte-Carlo runs of the simulation. The first two histograms are from the software program design simulation. The first represents how many effort-hours were spent doing rework on the software program design when using waterfall. The second represents how many effort-hours were spent doing rework on the software program design when using Agile. The third and fourth histograms are from the launch vehicle design simulation. The third represents how many effort-hours were spent doing rework on the launch vehicle design when using waterfall. The fourth represents how many effort-hours were spent doing rework on the launch vehicle design when using Agile. If needed, you can zoom in on the graphics using the "pinch-to-zoom" feature on a mobile device or "Ctrl+scroll" on a PC.
Effort-hours spent doing rework on the software design simulation when using Agile.
Effort-hours spent doing rework on the launch vehicle design simulation when using waterfall.
Effort-hours spent doing rework on the launch vehicle design simulation when using Agile.
Rate how well rework (called reformulations) in the simulation matches expectations for the types of system developments simulated on a scale of 1 to 5. 1 meaning a poor
match of real world reformulations  5 being a very good match of real world reformulations

☐ 5 (1)

☐ 4 (2)

☐ 3 (3)

☐ 2 (4)

☐ 1 (5)

Please explain the reason for your rating.

_________________________________________________________________
The second question is about positive emergent behavior. This is when a team has unexpected high productivity when creating a design. You will be shown histograms from 10,000 Monte-Carlo runs of the simulation. The four histograms are from the software program design simulation. The first represents how much time it took to create the software program design when using waterfall. The second represents how much time it took to create the software program design when using Agile. The third histogram represents how many effort-hours it took to complete the software program design when using waterfall. The fourth histogram represents how many effort-hours it took to create the software program design when using Agile.

How much time it takes represents the time that passes from when the design effort starts to when the design is complete. Effort-hours tracks the total number of hours the designers spend to complete the design effort. If needed, you can zoom in on the graphics using the "pinch-to-zoom" feature on a mobile device or "Ctrl+scroll" on a PC.
Time needed to create the software program design when using Agile.
Effort-hours needed to create the software program design when using waterfall.
Effort-hours needed to create the software program design when using Agile.
Rate how well the simulation data shows evidence of positive emergent behavior (unexpectedly high productivity) in the design team on a scale of 1 to 5

1 meaning no evidence of emergent behavior
5 being clear evidence of emergent behavior

- 5 (1)
- 4 (2)
- 3 (3)
- 2 (4)
- 1 (5)

Please explain the reason for your rating.

________________________________________________________________________

________________________________________________________________________
The third question is also about positive emergent behavior. This is when a team has unexpected high productivity when creating a design. This is similar to the previous question, but the data is specific to the launch vehicle design simulation. You will be shown histograms from 10,000 Monte-Carlo runs of the simulation. The four histograms are from the launch vehicle design simulation. The first represents how much time it took to create the launch vehicle design when using waterfall. The second represents how much time it took to create the launch vehicle design when using Agile. The third histogram represents how many effort-hours it took to complete the launch vehicle design when using waterfall. The fourth histogram represents how many effort-hours it took to create the launch vehicle design when using Agile. If needed, you can zoom in on the graphics using the "pinch-to-zoom" feature on a mobile device or "Ctrl+scroll" on a PC.
Time needed to create the launch vehicle design when using Agile.
Effort-hours needed to create the launch vehicle design when using waterfall.
Effort-hours needed to create the launch vehicle design when using Agile.
Rate how well the simulation data shows evidence of positive emergent behavior (unexpectedly high productivity) in the design team on a scale of 1 to 5

1 meaning no evidence of emergent behavior
5 being clear evidence of emergent behavior

- 5 (1)
- 4 (2)
- 3 (3)
- 2 (4)
- 1 (5)

Please explain the reason for your rating.

__________________________________________________________________________
The fourth question is on rework encountered when creating a design. Rework occurs when an earlier part of the design process much be revisited since something in the design is found to be lacking. For this question we are asking if there is evidence of negative emergent behavior in the results of the simulation. This is being assessed as unexpected high amounts of rework. You will be shown histograms from 10,000 Monte-Carlo runs of the simulation. They are the same histograms from question 1. The first two histograms are from the software program design simulation. The first represents how many effort-hours were spent doing rework on the software program design when using waterfall. The second represents how many effort-hours were spent doing rework on the software program design when using Agile. The third and fourth histograms are from the launch vehicle design simulation. The third represents how many effort-hours were spent doing rework on the launch vehicle design when using waterfall. The fourth represents how many effort-hours were spent doing rework on the launch vehicle design when using Agile. If needed, you can zoom in on the graphics using the "pinch-to-zoom" feature on a mobile device or "Ctrl+scroll" on a PC.

Effort-hours spent doing rework on the software design simulation when using waterfall.
Effort-hours spent doing rework on the software design simulation when using Agile.
Effort-hours spent doing rework on the launch vehicle design simulation when using waterfall.
Effort-hours spent doing rework on the launch vehicle design simulation when using Agile.
Rate how well the simulation data shows evidence of negative emergent behavior (unexpectedly high amounts of rework) in the design team on a scale of 1 to 5

1 meaning no evidence of emergent behavior
5 being clear evidence of emergent behavior

☐ 5 (1)

☐ 4 (2)

☐ 3 (3)

☐ 2 (4)

☐ 1 (5)

----------------------------------------------------------------------------------------

Please explain the reason for your rating.

----------------------------------------------------------------------------------------
For the next question, we will present the underlying Function-Behavior-Structure (FBS) model used in the simulation. The FBS model is a model of designers. The FBS model represents an individual designer. It maps the mental state that a designer is in at all points during the design process. The model consists of states and state transitions. A state represented by letters in the model as described below:

- **R** - Requirements
- **F** - Function
- **Be** - Expected Behavior
- **Bs** - Behavior of the structure
- **S** - Structure
- **D** - Documentation

The states describe what the designer may be working on at the various points in the design process.

Between states, certain state transitions occur where the designer can move from one state to another. The goal of the designer is to reach the Documentation state, where the design is finished. Most work moves towards this state except for three paths that rework can take. Rework starts at the structure state and can keep the designer in the structure state, move the designer back to the expected behavior state, or move the designer back to the function state. The FBS model is represented here as a 1st order Markov process. It is shown with transition probabilities between states. These transition probabilities represent the probability of moving to the next state or staying in the current state after 1 work day (8 hours). The mean performer model is shown along with the two extreme (low and high) performer models. You will notice that some of the FBS states have been collapsed into a single state shown as a dashed box.
Mean Performer Model

Q39
Rate how well the simulation numerically represents the variable performance of individual designers on a scale of 1 to 5. 1 meaning variability is poorly captured.
5 being variability is captured well

☐ 5 (1)

☐ 4 (2)

☐ 3 (3)

☐ 2 (4)

☐ 1 (5)

________________________________________________________________________

Please explain the reason for your rating.

________________________________________________________________________

________________________________________________________________________
Question #6: Design Processes

The next set of data shows the design of an agent-based simulation. In this simulation, each designer is an agent and is represented with an FBS model. The simulation drives the agents (designers) through a designer process to develop a design in both waterfall and Agile processes. Yellow highlights in the figures indicate which states of the FBS model the agent can be in at the different points in the design process.

Waterfall simulation design
uAgents step through their FBS states in such a way that they synchronize before moving to the next state
uEach function of a system is developed in parallel
uReformulations may require previous work to be revisited by certain agents
uCan slow project completion down
Rate how well the above simulation represents the process followed on a scale of 1 to 5

1 being a very poor representation
5 being a very good representation

- 5 (1)
- 4 (2)
- 3 (3)
- 2 (4)
- 1 (5)

Please explain the reason for your rating.

_________________________________________________________________

_________________________________________________________________

Agile simulation design

- Agents step through all FBS states as individual functions are completed and added to the system
- Functions are developed in a serial fashion by each team
- Agile teams work in parallel
- Once all functions are developed, the design effort is complete. After initial system definition is complete, agents can be in any state of the FBS states as individual functions are developed
- After initial system definition is complete, agents can be in any state of the FBS states as individual functions are developed
Rate how well the above simulation represents the process followed on a scale of 1 to 5

1 being a very poor representation
5 being a very good representation

○ 5 (1)

○ 4 (2)

○ 3 (3)

○ 2 (4)

○ 1 (5)

Please explain the reason for your rating.

__________________________________________________________________________

__________________________________________________________________________

__________________________________________________________________________
Question #7: Software Design Team

The next set of data shows the size and composition of the simulated software design team.

Each team has 8 software engineers
- Teams each develop functions in the software program
- Team leads and Program leader are each a single person
- Total of 101 individuals on the team

Rate how representative the size and composition of the development teams used in the simulation are on a scale of 1 to 5

1 meaning the size and composition of the development team is not representative of one needed for the subject system
5 being that the size and composition of the development team matches expectations for the subject system

- 5 (1)
- 4 (2)
- 3 (3)
- 2 (4)
- 1 (5)

Please explain the reason for your rating.

____________________________________________________________________

____________________________________________________________________

____________________________________________________________________
Question #8: Launch Vehicle Design Team

The next set of data shows the size and composition of the simulated launch vehicle design team.

Rate how representative the size and composition of the development teams used in the simulation are on a scale of 1 to 5

1 meaning the size and composition of the development team is not representative of one
needed for the subject system

5 being that the size and composition of the development team matches expectations for the subject system

- 5 (1)
- 4 (2)
- 3 (3)
- 2 (4)
- 1 (5)

Please explain the reason for your rating.

__________________________________________________________________________

__________________________________________________________________________
The next set of data shows the FBS model and how it was modified for use in the simulation. The Bs (Behavior of the Structure) state was combined with the Structure state. This is shown in the dashed-line box in the figure below. This combined the processes of deriving the structure (Be->S), determining the behavior of the structure (S->Bs), and comparing the behavior of the structure to the expected behavior (Be<->Bs). This was done to allow the FBS model to be used as a first order Markov process since this type of process only allows flow from state to state and comparing the behavior of the structure to the expected behavior (Be<->Bs) does not actually change the state in the model.
Rate how well the modification to the FBS model preserves the original purpose of FBS model on a scale of 1 to 5

1 meaning a very poor preservation of purpose

5 being a very good preservation of purpose

☐ 5 (1)

☐ 4 (2)

☐ 3 (3)

☐ 2 (4)

☐ 1 (5)

Please explain the reason for your rating.

________________________________________________________________
How many years of professional experience do you have?

- 0-1 (1)
- 1-5 (2)
- 5-10 (3)
- 10-20 (4)
- 20+ (5)

What is your year of birth?
What is the highest level of school you have completed or the highest degree you have received?

- Less than high school degree (1)
- High school graduate (high school diploma or equivalent including GED) (2)
- Some college but no degree (3)
- Associate degree in college (2-year) (4)
- Bachelor's degree in college (4-year) (5)
- Master's degree (6)
- Doctoral degree (7)
- Professional degree (JD, MD) (8)
Choose one or more ethnicity that you consider yourself to be:

☐ White (1)
☐ Black or African American (2)
☐ American Indian or Alaska Native (3)
☐ Asian (4)
☐ Native Hawaiian or Pacific Islander (5)
☐ Other (6) ________________________________________________

What is your sex?

☐ Male (1)
☐ Female (2)
☐ Other (3)
Which statement best describes your current employment status?

- Working (paid employee) (1)
- Working (self-employed) (2)
- Not working (temporary layoff from a job) (3)
- Not working (looking for work) (4)
- Not working (retired) (5)
- Not working (disabled) (6)
- Not working (other) (7)

________________________________________________

- Prefer not to answer (8)
How many employees work in your company?

- 1-4 (1)
- 5-9 (2)
- 10-19 (3)
- 20-49 (4)
- 50-99 (5)
- 100-249 (6)
- 250-499 (7)
- 500-999 (8)
- 1000 or more (9)
Where are you employed?

- PRIVATE-FOR-PROFIT company, business or individual, for wages, salary or commissions (1)

- PRIVATE-NOT-FOR-PROFIT, tax-exempt, or charitable organization (2)

- Local GOVERNMENT employee (city, county, etc.) (3)

- State GOVERNMENT employee; 5-Federal GOVERNMENT employee (4)

- Federal GOVERNMENT employee (5)

- SELF-EMPLOYED in own NOT INCORPORATED business, professional practice, or farm (6)

- SELF-EMPLOYED in own INCORPORATED business, professional practice, or farm (7)

- Working WITHOUT PAY in family business or farm (8)
Have you used waterfall processes in your career?

- Definitely yes (1)
- Probably yes (2)
- Might or might not (3)
- Probably not (4)
- Definitely not (5)

Have you used Agile processes in your career?

- Definitely yes (1)
- Probably yes (2)
- Might or might not (3)
- Probably not (4)
- Definitely not (5)
Debriefing

Thank you for participating in our study. The primary purpose of this study was to investigate the validity of certain aspects of a simulation of design teams. We will use your responses on the survey to determine if some of the subject aspects of the design team simulation are valid. The gathered data will aid in determining if the simulation requires updates before it produces valid results. The demographic information will be used to give researchers a better idea of the age range, proportion of males to females, and average education level of the people who participated in this study. All of this information will be completely anonymous and cannot be linked back to you as an individual. Finally, the responses that you have made during this survey will be used to help us improve the simulation. We hope to make the simulation accurate so it can be used in research to find better engineering processes. If participating in this study has led you to feel any discomfort, feel free to visit the UAH Counseling Center located at Wilson Hall 329. They can be contacted at 256-824-6203. If you have any questions about this study, feel free to ask the primary researcher, Mitch Bott, at mjb0021@uah.edu. Thank you for your help today.

If you would like your data to be excluded from our study, please check this box:

☐ Please exclude my data (1)