Task Optimization utilizing Digital Transformation Concepts - Automation Project Execution via AGILE Methodology

Anthony Steven Maiello
Florida Institute of Technology, amaiello2003@my.fit.edu

Follow this and additional works at: https://repository.fit.edu/etd

Part of the Operational Research Commons, Systems Engineering Commons, and the Systems Engineering and Multidisciplinary Design Optimization Commons

Recommended Citation
https://repository.fit.edu/etd/1398

This Thesis is brought to you for free and open access by Scholarship Repository @ Florida Tech. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholarship Repository @ Florida Tech. For more information, please contact kheifner@fit.edu.
Task Optimization utilizing Digital Transformation Concepts - Automation
Project Execution via AGILE Methodology

by

Anthony Steven Maiello

A thesis submitted to the College of Engineering and Science of
Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Systems Engineering

Melbourne, Florida
December 2023
We the undersigned committee hereby approve the attached thesis, “Task Optimization utilizing Digital Transformation Concepts - Automation Project Execution via AGILE Methodology” by Anthony Steven Maiello

Juan C. Avendano Arbelaez, Ph.D.
Graduate Faculty
Mathematics and Systems Engineering
Major Advisor

Luis Daniel Otero, Ph.D.
Associate Professor
Mathematics and Systems Engineering

Mary Ann Gaal, Ph.D.
Assistant Professor
Mechanical and Civil Engineering

Adrian M. Peter, Ph.D.
Associate Professor
Mathematics and Systems Engineering

Gnana Bhaskar Tenali, Ph.D.
Professor and Department Head
Mathematics and Systems Engineering
Abstract

Title: Task Optimization utilizing Digital Transformation Concepts - Automation

Subtitle: Project Execution via AGILE Methodology

Author: Anthony Steven Maiello

Advisor: Juan Avendano Arbelaez, Ph.D.

Task Optimization via the use of automated process improvements is becoming more widespread as more industries lean into the concepts surrounding digital transformation. This shift also necessitates a complementary adaptation in project management methodologies to support the rapid and ever-changing environment, requirements, and innovations. This thesis examines the effectiveness of Agile methodology in managing digital automation projects, with a specific focus placed on process improvements with systems engineering. It accomplished this by contrasting the original model, designed and derived utilizing traditional project management techniques, with the proposed model which is a direct result of the application of Agile project practices. This comparative analysis aims to highlight the contributions of Agile methodologies in facilitating the digital transformation process. It examines a case study of the use of Agile methodology to manage the development and implementation of an internal process improvement via the use of scripted automation. The Product Management and Sustainment
(PM&S) team has a monthly deliverable consisting of upwards of 150 discrete zip packages that must be delivered via a Customer Resource Management (CRM) System. The zip packages contain multiple workflow documents and asset inventory spreadsheets that are automatically updated for content. The current established process is to manually create each discrete zip package, as the content varies and is categorized by a customer identifier (referred to as a ‘node_ID’). Historical data analysis shows that creation of the zip packages takes approximately an hour or more of uninterrupted work depending on who is completing the task. Assigned day to day workload and tasking results in zip generation over the course of a week. This thesis presents a comparative analysis of the manual process versus the automated process, highlighting the impact on operational efficiency and error reduction.

Various applications domains exist that employ similar activities that could benefit from the use of automation. Examples of these use cases across these domains serve to demonstrate the potential scalability and adaptability of the developed utility across various industries with an emphasis on the broader implications of the proposed research. Some examples are:

• Tax Firms - Zip up completed tax packages for customers by customerID
  o Ideally for larger firms that may work on multiple packages at once
Utility would allow for package generation of multiple customers at once as opposed to as they are worked on

- Field Engineering team creates similar discrete packages for delivery to customers via the same CRM system
- Education - Student assignments where there are multiple files per week (folders by week) and discrete zip packages are required on submission as opposed to a single zip containing multiple directories
- Any need to create multiple zip packages
  - More than 3 zip packages on average to benefit from the time savings

The proposed solution involves the development of a PowerShell utility to automatically create the discrete zip packages. Ideally, the scripted solution will be generic enough for re-utilization on separate applications requiring similar capability. The development, implementation, and outcomes of this proposed utility are evaluated, providing insights into the broader application of digital transformation initiatives and tools in systems engineering. Creation and use of an automated utility significantly decreases user error in package generation and manual execution time, as well as removing the need for package validation. The research findings underscore the utility’s role in supporting the company’s digital transformation efforts while also contributing to the field of systems engineering.
# Table of Contents

Abstract ................................................................................................................................. iii
List of Figures ....................................................................................................................... viii
List of Tables ......................................................................................................................... ix
Acknowledgement ............................................................................................................... x
Dedication ............................................................................................................................... xi
Executive Summary ............................................................................................................. 1
Chapter 1: Introduction ........................................................................................................ 3
Chapter 2: Literature Review ............................................................................................... 11
  Digital Transformation – Automation .............................................................................. 11
  Agile Processes in Systems Engineering ........................................................................ 13
Chapter 3: Methodology ....................................................................................................... 17
  Introduction to the Case Study ....................................................................................... 20
    Why this Case Study is an Ideal Candidate ............................................................... 20
  Selection for Analytical Comparison ........................................................................... 21
  Description of the Case Study and Process Improvement ......................................... 21
    Agile Methodology's Role in Enhancing Efficiency ................................................... 22
  Technical Section ............................................................................................................. 23
    Research Methodology and Value .............................................................................. 24
    User Requirements – The Foundation of Research Contribution ............................. 24
    System Requirements – Validating the Research Objective ...................................... 26
Chapter 4: Method Comparison .......................................................................................... 28
  Research Methodology and Value .................................................................................. 28
  Importance of User Requirements in Research Contribution .................................... 28
  Implementation Comparison: Current vs Automated .................................................... 29
    Current Implementation ............................................................................................... 29
    Automated Implementation .......................................................................................... 29
  Beyond Software Tool Creation – Agile’s Broader Impact ............................................ 30
Chapter 5: Functional Section ........................................................................................... 31
 Workflow (CONOPS) .................................................................31
 Utility Work Instruction..........................................................33
 Utility Execution Paths ............................................................35
 Chapter 6: Results ......................................................................51
 Metrics .......................................................................................52
 Data Analysis .............................................................................55
 Results and Comparative Analysis of Project Management Methodologies ....62
 1. Operational Efficiency Improvement ...................................62
 2. Error Reduction .................................................................62
 3. Agile vs. Traditional Project Management ...........................63
 Chapter 7: Conclusion ...............................................................64
 Future Work and Applications .................................................67
 References .................................................................................70
 Appendix: Zip Utility PowerShell Code .....................................73
List of Figures

Figure 1 – Jira Timeline View ................................................................. 8
Figure 2 – Confluence Meeting Notes Example .................................... 20
Figure 3 – Utility Workflow .................................................................. 32
Figure 4 – Utility Extraction ................................................................. 34
Figure 5 – Sample Directory View ....................................................... 34
Figure 6 – Utility Info & Caution .......................................................... 35
Figure 7 – Launcher Button .................................................................. 36
Figure 8 – Utility Running with File Browser Pop-up ............................ 37
Figure 9 – Example of Parent Directory structure with sub-folders ....... 38
Figure 10 – Utility Running Pop-Up ..................................................... 38
Figure 11 – User Prompt ....................................................................... 39
Figure 12 – ‘Yes’ selected output ......................................................... 40
Figure 13 – Utility Exiting ..................................................................... 40
Figure 14 – ‘Yes’ Timing Report ............................................................ 40
Figure 15 – ‘Yes’ Completed Utility Run .............................................. 41
Figure 16 – ‘Yes’ Removed Files ........................................................... 42
Figure 17 – ‘No’ selected output ........................................................... 43
Figure 18 – ‘No’ Timing Report ............................................................. 43
Figure 19 – ‘No’ Completed Utility Run ............................................... 44
Figure 20 – Risk Analysis Table ............................................................ 46
Figure 21 – Task Distribution ............................................................... 53
Figure 22 – Workload by label .............................................................. 54
Figure 23 – Total Extrapolated Data Set (Summary) ............................... 56
Figure 24 – Extrapolated Data Set Runs (a) .......................................... 57
Figure 25 – Extrapolated Data Set Runs (b) .......................................... 58
Figure 26 – Utility Data (Sampled) ....................................................... 60
List of Tables

Table 1 – Traditional vs Agile Comparison

.................................................63
Acknowledgement

I want to offer my sincere thanks to every person that helped me on my way to achieving this goal and to all the professors that imparted their knowledge and guidance throughout this journey. Thank you for taking the time; I hope you know how meaningful that is. Thank you, Dr. Juan Avendano, for your guidance as my advisor and for the belief you hold in my abilities and the support in allowing me to tread new ground.

I am also incredibly thankful for my family, friends, and co-workers for their understanding and support as I pursued this Master’s degree.
Dedication

Dedicated to my wife, dog, and cat for being patient through all the long hours.
Executive Summary

This thesis offers a comprehensive exploration of the application of Agile methodologies in the domain of software systems engineering, with a particular focus on a specific implementation within the defense industry. The research conducted serves to bridge significant and crucial gaps that exist within the current literature, notably the dearth of empirical studies pertaining to the incorporation of Agile methodologies in the digitalization of specific legacy procedures within large-scale software engineering initiatives.

Within the literature, a significant inadequacy lies in the lack of comprehensive analyses regarding the application of Agile methodologies in the digital transformation of manual processes. Notably, the scarcity of studies providing quantitative insights into the efficiency of Agile implementation, especially when compared to traditional manual processes, is evident. This research takes a proactive stance in identifying and addressing these literature gaps. It achieves this by conducting a meticulous exploration and application of Agile methodologies in a real-world example of the digitization process. The study includes a comparative analysis, focusing on both the implementation of Agile methodology and the resulting digital tool and new process, demonstrating considerable optimization.

This research significantly contributes to the academic literature by conducting an exhaustive analysis of Agile methodologies and delineating a comprehensive framework for their application in the digitization and transformation of manual, non-automated processes. A key aspect of this contribution involves a detailed comparative analysis between traditional methodologies and Agile approaches. This analysis not only reveals the strengths and weaknesses associated with the implementation of Agile methodologies but also delves into the optimization outcomes achieved in the processes.
In addition, this research derives substantial value from a real-world case study conducted within the defense industry. This case study holds exceptional significance due to its focus on the pragmatic application of Agile methodologies in the realm of digitalization and automation of manual processes. It is essential to note that this research employs a distinctive methodology, deviating from conventional project management and requirements analysis techniques. This unique approach results in the development of an automated tool, offering a tangible, real-world solution. Notably, this tool can be objectively compared to solutions generated through traditional management techniques, effectively showcasing the impact of Agile methodologies in this specific context. Furthermore, it accentuates the noteworthy efficiency of procedures and digital tools developed through Agile methodologies when compared with traditional or standard methods.

Additionally, this thesis establishes a platform for future research undertakings, creating a basis for potential expansions into various sectors beyond the current study's boundaries. The capability to compare the efficiency of processes before and after digital transformation with Agile methodology creates opportunities for researching and classifying digital transformation tools based on the efficiency of the new processes or tools generated. Subsequent research could explore the integration of artificial intelligence (AI) in systems engineering tools and engage in longitudinal studies to assess the sustainability of Agile methodologies in digitizing similar processes in other industries. Moreover, the in-depth and systematic comparison presented for the development of digitized and automated processes using both Agile and traditional systems models could be extended to more intricate processes in similar industries. Notably, a controlled experiment involving separate teams attempting process digitization using multiple methodologies is worth considering as a significant avenue for future research.
Chapter 1: Introduction

It may not be readily apparent, but the inclusion of a well thought out project management plan is a paramount component of successful systems engineering. Often times, systems engineering initiatives focus more on core principles such as properly derived requirements and sufficient test evidence, with aspects such as schedule and resource loading being considered an afterthought or being relegated to a traditional project engineering role. In contrast, this thesis posits that project management, specifically Agile methodologies, plays a critical role in the efficiency and success of systems engineering projects. The focus of this research extends beyond the traditional academic setting, instead examining a real-world case study. The focus of this case study examined a project that implemented digital transformation via the introduction of an automated utility as a process improvement to a man-hour intensive exercise, the creation of a large quantity of unique zip packages. This research specifically aimed to investigate the integration of digital automation tools with Agile methodologies and how this combination could impact operational efficiency and lead to a measurable error reduction in systems engineering tasks and processes, addressing a significant gap in the current literature. The primary criterion for the project was that it needed to satisfy coverage of the core subject areas of the Systems Engineering degree program; beyond that, execution was at the sole discretion of the engineer in regards to all phases of the project including the project management. At a high level, the project consisted of weekly meetings over Zoom with the project advisor, at which point weekly deliverables and schedule were to be reviewed. These meetings also provided a forum for academic discourse and inquiry, essential for the research process.
Informed by the comprehensive analysis presented in the literature review, the decision to adopt an Agile strategy for the execution and implementation of this case study, which includes the use of tools like Jira and Confluence, was made with specific objectives in mind:

1. **Responsiveness to Change**: Agile methodologies are renowned for their adaptability and responsiveness to changing project dynamics. This flexibility is essential in projects involving digital automation, where evolving requirements and continuous feedback are integral to success. Agile's iterative nature allows the project to pivot and adapt as needed, ensuring that the developed utility remains aligned with the ultimate goal of process improvement.

2. **Emphasis on Continuous Improvement**: The literature emphasizes the value of continuous improvement inherent in Agile methodologies. In the context of this project, the ability to iteratively develop and refine the utility ensures that each phase of the project builds upon the learnings of the previous, culminating in a more effective and efficient final product.

3. **Enhanced Collaboration and Transparency**: Agile methodologies facilitate improved collaboration and transparency within project teams. Utilizing tools like Jira and Confluence enhances this aspect, ensuring all stakeholders are consistently engaged and informed about the project's progress, thereby fostering a collaborative environment essential for innovative projects.

4. **Bridging Software Development and Systems Engineering**: Although this case study involves the development of a software utility, its application extends beyond typical software development. The utility is designed as a tool for process improvement within systems engineering, specifically for automating the creation of zip packages. This project, therefore, operates at the intersection of software development and systems engineering, showcasing the
application of Agile methodologies in a unique context. The use of Agile here aims to highlight its effectiveness not just in software creation, but in employing software as a means to optimize traditional systems engineering processes.

5. Project Management and Tracking: The use of Jira and Confluence aligns with Agile’s focus on efficient project management and tracking. These tools support a structured yet adaptable framework for task organization, documentation, and progress monitoring, crucial for maintaining rigor and consistency in a research-driven project.

The selection of an Agile strategy for this case study is a deliberate response to the insights from the literature review. It reflects a strategic choice to employ a methodology that is adaptable, collaborative, and iterative, making it well-suited for addressing the unique challenges and objectives of this specific project in digital automation within the realm of systems engineering. Jira was used to build out the schedule and discrete tasks necessary to complete the project. It offered multiple views – Summary, Board, List, Calendar, Timeline – as well as integration with Confluence for presentation of the final deliverable and all supporting documentation and artifacts. These resources allowed for the appropriate tracking of the effort and provided at-a-glance progress reporting for the project advisor to review. Of the available views, the timeline one was the most important in the accelerated project timeframe, as it represented the information typically seen in a Gannt chart or an Integrated Master Schedule (IMS) with the added benefit of real-time updates. This visibility was critical in a fully self-managed project.

The current landscape of literature on Agile methodologies in systems engineering exposes notable gaps that this research aims to fill. Notably, there is a deficiency in comprehensive studies concerning the integration of Agile methodologies in non-software
engineering environments, particularly in large-scale industrial settings. Additionally, existing research provides limited insight into the practical challenges and solutions encountered when applying Agile principles to complex engineering projects. This thesis actively contributes to addressing these gaps by conducting a detailed analysis of the application of Agile methodologies in the case study. In doing so, the research not only enhances the existing literature but also imparts practical insights and recommendations for the application of Agile methodologies in diverse systems engineering contexts.

This thesis contributes to the academic field by providing a comprehensive case study that applies Agile methodologies and a digital automation tool - a PowerShell utility - in a systems engineering context. The research goes beyond the conventional boundaries of software development, exploring how these methodologies and tools can optimize a specific systems engineering process: the creation of zip packages. This case study not only enriches the existing body of knowledge, but also offers practical insights as well as a model that can be replicated or adapted in similar systems engineering tasks, thus bridging the identified gap in the literature.

**Main Research Objective:**

The main research objective of this thesis is to assess the impact of integrating Agile methodologies and digital automation tools on operational efficiency and error reduction in a specific systems engineering process. This objective is derived from the need to understand how Agile practices, often confined to software development, can be effectively translated into the systems engineering domain, particularly in enhancing routine operational tasks.
Research objective for the Case Study:

The case study meticulously documents the development and implementation of the PowerShell utility, focusing on its design, deployment, and the efficiencies gained from its usage. The study employs quantitative methods to measure the improvement in operational efficiency and the reduction in error rates before and after the utility's implementation. This empirical evidence not only validates the utility of Agile methodologies and digital tools in a new context, but also provides a benchmark for operational improvements in similar engineering scenarios.
Figure 1 – Jira Timeline View
As indicated in the Jira timeline presented in Figure 1, the project would cover the following major categories, identified as primary tasks, with a set of sub-tasks breaking down the primary ones into achievable and trackable goals. This breakdown supports the agile value of “responding to change over following a plan” by limiting the breadth and scope of potential changes into something more easily managed, especially given the reduced timeline, and working hour availability.

**Maiello Capstone (MC)**

- MC-1: Topic Submission
- MC-4: Plan Proposal
- MC-5: Requirements Analysis
- MC-2: Scope definition (CONOPS)
- MC-3: Literature Review
- MC-6: Systems Architecture (Simulated)
- MC-7: Research and Selection
- MC-8: Risk Assessment and Mitigation
- MC-9: Implementation and Test
- MC-14: Validation and Verification
- MC-17: Reporting
- MC-18: Final Presentation

Confluence was utilized to deliver the overall project content in a centralized, integrated, and easy to consume manner. The interoperability between the two allowed for seamless integration of project components from either Confluence (such as meeting notes and work
instructions) or Jira (such as timelines and task tracking) to each other. This interconnectivity increased project visibility and transparency which in turn yielded a more efficient usage of time. The primary elements managed in Confluence were a high-level design plan, the action items or decisions from the weekly meetings, a work instruction for the utility along with test evidence, SWOT, risk analysis tables, and any additional artifacts and references.
Chapter 2: Literature Review

Digital Transformation – Automation

The concept of utilizing digital transformation to improve upon internal systems engineering processes is not a new one. In Kuhn’s 1994 “Automating systems engineering,” it is discussed that the “need for effective automation in systems engineering has been a subtext of nearly every book on this fascinating subject since the early 1960s” (Kuhn, 1994). At the time, automation in systems engineering was a still a fairly new concept. The article went on to highlight that the three necessary conditions for developing an automation tool in systems engineering had finally been met. Those conditions included the ability to effectively describe the systems engineering process, the increasing complexity of systems and the availability of sufficient computing power. However, Kuhn’s analysis stops short of exploring the practical application of these automation tools in real-world systems engineering tasks, which this thesis aims to address through a contemporary case study. Fast forward to today, digital transformation in systems engineering is much more prevalent, especially with the available computing power today. This is especially relevant to tasks that are well defined and repeatable in nature. From a company perspective, often a member of a systems engineering team will be tasked with manually performing a repetitive task; however, literature on the subject often overlooks the specific challenges and nuances involved with the automation of such tasks in a real-world environment. The potential for task fatigue often drives engineers to explore methods to reduce the amount of time spent on performing said task, usually through digital automation. This example is another sub-set of automation as it relates to engineering processes, those innovations associated with internal processes. Unfortunately, there is a gap in academic publications in this realm. This is likely due to restrictions placed by the corporations, the specificity of tasks that
may not have applicability outside of the given task, program, or company or a general lack of analysis and reporting done on these efforts. This thesis contributes to filling this gap through the documentation and analysis of a specific case of digital automation in systems engineering.

Digital automation can improve the efficiency and effectiveness of systems engineering by providing repeatable frameworks for automation and information technologies. This includes the use of model-based systems engineering (MBSE) to create a roadmap for digitalization (Arquimedes, Gustavo Arturo, Georg, Elisabeth, & Jörg, 2020). While the literature addresses the theoretical framework of MBSE, there is a need for empirical evidence demonstrating its practical applications, which this thesis provides. In this thesis, we will explore the impact of digital automation tools on operational efficiency in systems engineering and how these tools not only have the potential for scalability and adaptability across diverse systems engineering processes, but also how the adoption of these automated tools can lead to reductions in error rates as they apply to routine engineering tasks.

 Another important analysis of the digital transformation boom in systems engineering is the risk of human errors as they relate to the introduction of new technologies. If new technologies are not implemented correctly, human error in operation can quickly manifest. The pattern of human performance degradation when novel technologies are introduced has been found in a wide range of endeavors (Corrado, 2022). The aforementioned study aimed to address the negative effects of advancing technologies on nuclear plant operations by discussing concepts such as operator involvement in the systems engineering process, human performance integration with defined systems requirements, and developed standard operating procedures and training as it pertained to the creation of training programs that aligned with operational requirements integrated with human performance requirements to aid in the reduction of human
error associated with incorporating advanced technology in nuclear facilities. This thesis further extends this discussion by exploring how digital automation tools, when properly implemented and integrated with Agile methodologies, can mitigate the risk of human error in systems engineering processes, a topic not extensively covered in the current literature. Another paper proposed automating error processing at the system side, thus reducing the need for manual intervention and ultimately leading to a reduction in manual intervention and consequently a decrease in development and maintenance manhours. In the context of the paper, “when a software error except those related to the processing contents of the application program decided previously occurs, this error is processed in the operating system. Thus the manhours in the development and manhour can be reduced” (Yamazaki & Nomura, 1991). What methods are available to measure this type of reduction or improvement? This thesis aims to provide a quantitative analysis of this aspect, which is often lacking in existing literature.

**Agile Processes in Systems Engineering**

Agile methodologies in systems engineering and project management offer several benefits. These include the ability to quickly respond to changes in the external and internal environment, allowing for adaptation and transformation of processes and products (Альмухаметов, Дмитриев, Almukhametov, & Dmitriev, 2023). The article goes on to discuss several different agile methodologies that provide effective ways to manage projects due to the focus placed on continuous improvement, flexibility, and collaboration and how these concepts can aid in efficiently managing groups working on myriad different tasks while ensuring a measure of trackable coordination throughout the project. However, there is a need for empirical research to substantiate these claims, specifically in the context of systems engineering beyond
software development, which this thesis aims to address. For many years, Agile has been limited to software development, but we have witnessed a shift in recent times due to the adaptability and effectiveness of agile methodologies across various contexts. We often see a combination of Agile with more traditional or structured project management approaches, to take advantage of the benefits offered by each. This thesis aims to contribute to the literature by examining this shift in systems engineering context, providing insights into the practical application and outcomes of integrating Agile with traditional project management approaches.

Based on an article in the *Journal of Systems and Software (Elsevier)*, the authors present a systematic review of the modifications made to *Scrum*, a sub-method of Agile, in different instances and analyzing the reasons behind these modifications as well as the strategies used to implement. The study aimed to provide a basis for future research by identifying common modification objectives and strategies, while also highlighting research gaps. This thesis extends the discussion by exploring how these modifications apply and function in the specific context of digital automation in systems engineering, thereby serving to address some of the identified gaps.

Another study into the use of Agile methods aimed to analyze theories that support the use of these methodologies as they apply to systems engineering, including non-software systems. It examined how the methodologies employed “promote transparency and visibility, as progress is regularly tracked and shared with stakeholders, ensuring that everyone is aware of the project’s status” (Bott & Mesmer, 2020). Additionally, the authors highlight that the use of Agile in systems engineering also allows teams the ability to deliver high-quality products within reduced timeframes while being able to effectively respond to changes in customer requirements.
While this literature establishes the theoretical benefits of Agile, this thesis contributes by presenting a case study that demonstrates these benefits in action, particularly in enhancing operational efficiency and error rate reduction in systems engineering tasks.

A significant transformation has taken place in recent years, as Agile methodologies have expanded their reach beyond software development to encompass complex systems engineering contexts. This thesis contributes empirical research and real-world case studies, with a specific focus on the in-depth analysis of Agile implementation in a specific industry case study, validating this paradigm shift.

This case study serves as a prime example, demonstrating the adaptability and efficacy of Agile methodologies in managing large-scale, intricate engineering projects. It provides tangible evidence of how Agile practices seamlessly integrate into traditional systems engineering processes, enhancing efficiency, flexibility, and responsiveness to change.

Furthermore, additional case studies and examples from prominent engineering firms serve to corroborate these findings, highlighting the versatile application of Agile principles across various industries. These real-world examples not only strengthen the theoretical foundations of Agile methodologies but also furnish practical insights and guidelines for their effective implementation in diverse engineering scenarios.

From the literature review, it has become apparent that a complete set of relevant comparison criteria between Agile and Systems Engineering does not exist, likely due in part to the vast differences in project types, scope, and industries. This thesis aims to help fill this gap by providing a detailed comparative analysis of Agile methodologies in the context of a specific
systems engineering project, thus contributing a new perspective to the body of literature. These gaps will diminish as the methodologies are more widely adopted, especially given the versatility and adaptability afforded to the methodologies; this adaptability allows for an ever-increasing breadth of usability. Through this thesis, we explore and assess the effectiveness of Agile methodologies in managing and implementing digital transformation projects in systems engineering, thereby enhancing the understanding of application of Agile in a broader range of engineering disciplines.
Chapter 3: Methodology

This research utilizes a mixed-methods approach, integrating both qualitative and quantitative analyses to thoroughly evaluate the application of Agile methodologies in systems engineering. The rationale for adopting this approach lies in the pursuit of offering a holistic perspective on the subject matter, encompassing both theoretical insights and practical implications. Qualitative analysis incorporates detailed case studies, with a specific focus on the representative case study, to illustrate real-world applications. Simultaneously, quantitative methods involve data analysis to objectively measure the impact of Agile practices. The integration of these methods directly tackles the identified gaps in the literature, specifically the demand for empirical studies on Agile methodologies in complex engineering environments. This methodology goes beyond enriching the prevailing body of knowledge, offering tangible practical insights for industry practitioners.

This research approach makes a substantial contribution to the field of systems engineering, offering fresh perspectives on the implementation of Agile methodologies and showcasing their adaptability and efficiency in diverse engineering contexts. Utilizing both qualitative and quantitative methods, it offers valuable insights into the practical implementation and challenges of Agile in this specialized field. Beyond exemplifying mixed-methods research, this case study significantly enhances the understanding of Agile's broader potential in complex, technology-driven industries.

Systems Engineering is an interdisciplinary and integrative approach. It enables a successful realization of systems, by taking customer needs, requirements and system validation into account (John Wiley and Sons Inc, 2015). Over the past few years, there has been a
significant shift from traditional project management strategies to Agile, continuous integration and continuous delivery, the combination often referred to as DevOps. This shift is crucial in addressing the gaps in literature, particularly the application of Agile methodologies beyond software development, which this thesis investigates. The significant shift happening today towards more connected, more automated, and more autonomous systems is bringing software inside all systems, and at the same time agile practices (Alt & Le Mouëlï, 2023). This integration presents a unique opportunity for academic exploration, aligning with the research objective presented in this thesis to assess the impact of Agile methodologies in systems engineering processes. Predominantly, these management strategies have been limited to software development, but the practices have a much greater breadth of applicability upon closer inspection, especially once combined with Systems Engineering principles. All methods are centered around working in short iterative cycles, whereby each method uses a different name for those work increments (sprint, iteration, etc.). However, the core process of all methods follows the cycle of iteration planning, iteration work, demonstration, review, and retrospective (Humpert, Mundt, Bretz, & Anacker, 2022). A key academic contribution of this thesis is demonstrating how these iterative cycles, typically associated with software development, can be effectively applied in systems engineering tasks, addressing a notable research gap. A key benefit of having selected Agile for this project was that there are several different strategies within the overall Agile methodology that could be adapted. Most authors and advocates of the Agile methodologies encourage changes in the methodology itself to suit your development team and/or company (Lindblom, 2015). This adaptability of Agile, applied in a systems engineering context, is central to this research, offering a novel perspective on Agile’s flexibility and utility in diverse project environments. Agile does not need to be limited to large-scale software projects.
It is this combination of systems engineering and the adaptability of Agile principles that was examined, selected, and ultimately implemented for this project and the basis of the case study.

The choice of Agile for this case study directly addresses the research objectives by providing a framework for evaluating its effectiveness in operational efficiency and error reduction in systems engineering. Jira and Confluence were selected as the vehicles/tools to accomplish this. One of the primary drivers for using Jira and Confluence was the familiarity with the applications as they are widely used across many teams, including the Product Management & Sustainment team. This visibility is crucial for the research, as it allows for clear documentation and tracking of the project’s progress, an essential component for the empirical analysis required in academic studies. It was previously stated that Confluence was not only used for the presentation of findings and artifacts, but it also served as the ‘system of record’ for the meeting notes and actions captured throughout the project. These notes are pivotal for the academic integrity of the case study, providing transparent and detailed documentation of the project’s evolution, key decisions, and outcomes. These notes included but were not limited to the generation of goals, high-level questions for the project advisor, a projected timeline mapping elements of the systems engineering curriculum to the facets of the proposed project, and actions for both the project advisor and the project engineer/manager. One high-level note from one of these meetings can be seen in the figure below:
Introduction to the Case Study

The case study featured in this thesis represents a pivotal element of our research, offering a practical, real-world example to study the application of Agile methodologies in systems engineering processes. The insights garnered from this study not only showcase the transformative impact of Agile in large-scale systems engineering but also open avenues for potential extrapolations to other sectors within the aerospace and defense industries. Additionally, the principles and outcomes observed in this study hold substantial relevance and applicability to various other industries where innovation, efficiency, and adaptability play a crucial role. The selection of this particular case study is grounded in its potential to address significant gaps identified in the existing literature and to contribute meaningful insights towards the practical implementation of Agile methodologies.

Why this Case Study is an Ideal Candidate

The case study revolves around a systems engineering task that is routine yet critical in nature: the creation of discrete zip packages. This task was traditionally executed manually, requiring significant man-hours and being prone to errors. The simplicity yet fundamental nature
of this task makes it an ideal candidate to study. It provides a clear and tangible context to explore the effectiveness of Agile methodologies and digital automation tools, which are often discussed in theory, but lack substantial empirical evidence in systems engineering contexts. By focusing on a specific, relatable, and measurable task, the case study directly addresses the literature gaps concerning the practical application of Agile outside traditional software development environments.

Selection for Analytical Comparison

The case study serves as a tool for analytical comparison between the traditional method of executing the task and the new method developed using Agile methodologies. This comparison is crucial to evaluate the research objective: assessing the impact of Agile methodologies on operational efficiency and error reduction in systems engineering processes. The case study provides a framework to analyze improvements quantitatively and qualitatively in process efficiency, error rates, and overall workflow effectiveness resulting from the application of Agile principles.

Description of the Case Study and Process Improvement

The case study details the process of automating the creation of zip packages. Initially, this task involved locating specific data files, manually creating zip packages, renaming them as directed per customer identifiers, and conducting manual validations — a time-consuming and error-prone process. The introduction of an automated solution, developed through Agile project management methodologies, revolutionized this process. Agile's iterative approach, with its emphasis on continuous improvement, adaptability, and collaboration, was instrumental in
developing a solution that not only automated the task, but also made it more efficient and less susceptible to errors.

**Agile Methodology's Role in Enhancing Efficiency**

The application of Agile methodologies in this case study facilitated a rapid and responsive development process. It allowed for regular feedback, rapid adaptations to challenges, and continuous refinement of the utility. This not only accelerated the development timeline, but also ensured that the final product effectively met the user requirements. The result was a significant improvement in process efficiency, as evidenced by reduced time for package creation, decreased error rates, and enhanced workflow automation.

This case study is not only an example of process improvement in systems engineering, but also a critical examination of how Agile methodologies can be effectively applied in areas beyond their traditional scope. The findings from this case study provide valuable insights into the research objectives and contribute to the broader understanding of Agile's application in diverse engineering contexts.
Technical Section

The technical section of this thesis forms the core of our empirical research, providing a detailed account of the design, development, and evaluation of the automated utility. The focus on user and system requirements is crucial, as these requirements not only guide the technical development of the utility, but also serve as fundamental elements in assessing the research contribution of this study. The development of the PowerShell utility in this thesis presents an innovative approach, markedly different from existing methodologies in systems engineering. In contrast to traditional methods that often rely on manual processes and are prone to human error, the PowerShell utility utilizes automation to enhance accuracy and efficiency. This automated approach proves especially beneficial in complex systems, where precision and reliability are paramount.

The utility's modular and scalable design marks a substantial improvement over more rigid and less adaptable existing systems. Its ability to seamlessly integrate with various engineering environments not only showcases the tool's versatility but also highlights its potential applicability across different industry sectors. Critically, the development of this utility directly addresses the research gaps identified in the literature review. It furnishes empirical evidence of the practical application and benefits of Agile methodologies in systems engineering, making a significant contribution to both academic research and industry practices. By providing a tangible example of innovation in system management, this work emphasizes the potential of Agile approaches in enhancing operational efficiency and adaptability in diverse engineering contexts.
Research Methodology and Value

The development of the utility is grounded in a systematic and methodical approach, reflecting a blend of theoretical knowledge and practical application. By meticulously documenting each step - from initial user requirements to final system requirements - this section offers a transparent and comprehensive view of the research process. The detailed enumeration of requirements serves as a baseline against which the effectiveness and efficiency of the utility are measured, providing tangible evidence of the utility's impact on operational processes.

User Requirements – The Foundation of Research Contribution

The identification of user requirements is integral to this research, as it ensures that the utility is not only technically sound, but also aligned with the actual needs and challenges faced by users in systems engineering tasks. Each requirement is carefully chosen to address specific aspects of the operational inefficiency and error prone areas identified in earlier parts of the thesis. This alignment with user needs not only enhances the practical relevance of the utility, but also contributes significantly to the academic understanding of how digital tools can be tailored to improve specific engineering processes.

Within the realm of systems engineering challenges, user and system requirements assume a pivotal role. These requirements, derived from the identified literature gaps, address critical issues in systems engineering, including flexibility, scalability, and integration challenges. For instance, user requirements for this case study underscored the necessity for agile, adaptable systems to navigate rapidly changing technological and market conditions. Concurrently, system requirements were formulated to ensure robustness and efficiency, directly confronting the inefficiencies and rigidities often found in traditional systems engineering approaches. The direct
correlation between the identified literature gaps and the specific requirements set forth in this study underscores the practical relevance and applicability of our research in advancing the field of systems engineering.

The documentation details the development of a PowerShell utility designed to automate the creation of zip packages, a task traditionally carried out manually and susceptible to errors. The user and system requirements for this utility align with the operational inefficiencies and error-prone areas identified in the literature review. This alignment ensures that the utility not only meets technical standards but also addresses specific challenges in systems engineering, contributing significantly to the field by offering practical solutions derived directly from the gaps found in the literature. This approach emphasizes the practical application of digital tools in improving specific engineering processes, bridging a gap in academic publications.

**User Requirements**

1. **Utility to Automate the Process of Creating Discrete Zip Packages**: Central to the research, this requirement addresses the primary objective of improving operational efficiency.

2. **User Guide**: Ensuring the utility is accessible and user-friendly aligns with the goal of broad applicability and ease of adoption in systems engineering contexts.

3. **Testing within a Sprint with Provided Test Evidence**: This requirement underlines the Agile approach adopted and provides empirical data crucial for the research.

4. **Data Analysis Representing the Value Add of Automation**: Critical for evaluating the research hypothesis and quantifying the utility's impact.
5. **Option to Clean Up Parent Directory (Source Data):** Reflects a real-world application scenario, adding depth to the research.

6. **Utility in a Scripting Language:** Emphasizes ease of development and deployment, relevant to the research context of Agile methodologies.

**System Requirements – Validating the Research Objective**

The system requirements further build on the user requirements, offering technical specifications that ensure the utility's functionality aligns with the identified user needs. These requirements are instrumental in the empirical evaluation of the utility, providing a structured framework to assess its effectiveness and efficiency.

**System Requirements**

1.1. Utility shall be able to automatically create discrete zip packages of the source data upon execution

1.2. Utility shall have a user display
   1.2.1. User display shall report zip package creation
   1.2.2. User display shall report execution time

1.3. Utility shall have user instructions
   1.3.1. Utility shall have a work instruction
   1.3.2. Utility shall prompt the end user for location of directories to be zipped
   1.3.3. Utility shall prompt the end user with the option to clean up parent directory (source data)
      1.3.3.1. Utility shall provide a record of removed files

1.4. Testing shall be completed on the developer’s system
1.4.1. Test evidence shall be provided

1.4.2. Testing shall have at a minimum been conducted over 5 sets of source data

1.4.2.1. Source data sets shall contain multiple file types

1.5. Utility shall be written using a scripting language (e.g., batch, Linux, PowerShell, Python, etc.)

This technical section is pivotal in demonstrating how the integration of Agile methodologies and digital automation tools can enhance systems engineering processes. By rigorously detailing and fulfilling each requirement, this study not only contributes a practical tool to the field, but also enriches the academic understanding of Agile's applicability in systems engineering.
Chapter 4: Method Comparison

Chapter 4 of this thesis presents a critical analysis of the current and automated implementations of the process for creating discrete zip packages. This method comparison section is central to the research methodology, as it provides a direct and empirical comparison between the traditional project management approach and the newly applied Agile project management methodology. This comparison is crucial in demonstrating how Agile methodology enhances this particular process beyond the mere creation of a software tool.

Research Methodology and Value

The research methodology employed in this section is comparative in nature, aiming to draw clear distinctions between the current and automated implementations. By quantifying and analyzing differences in efficiency, error rates, and overall effectiveness, this section adds significant value to the research. It moves beyond theoretical discussions of Agile methodologies, offering concrete evidence of their impact in a real-world systems engineering context.

Importance of User Requirements in Research Contribution

The user requirements, identified earlier in the thesis, play a pivotal role in framing this comparison. They serve as benchmarks for evaluating the effectiveness of the Agile methodology in addressing specific user needs and operational challenges, which the current implementation fails to meet adequately.
Implementation Comparison: Current vs Automated

Current Implementation

The traditional approach to creating zip packages is manual and time-intensive, involving locating data files, creating and renaming zip packages to match customer identifiers, and manual validation. This section will detail how this approach, derived from conventional project management techniques, aligns with or falls short of meeting the user requirements.

- Manually create each discrete zip package
- Locate data files and create zip
- Re-name file (if needed) to match customer identifier
- Historical data analysis indicates that the creation of the zip packages takes approximately 1+ hours of uninterrupted work depending on who is completing the task
- Each file needs to be manually validated further increasing required time

Automated Implementation

Contrasting the current method, the automated process, developed using Agile methodologies, simplifies and streamlines the task. This section will explore how pre-work requirements, the ability to generate discrete zip packages automatically, and the utility's generic nature for re-utilization significantly improve operational efficiency. It will also demonstrate how Agile methodologies contribute to a decrease in user error and manual execution time, eliminating the need for manual package validation.

- Automatic generation of discrete zip packages
- Pre-work only requires a parent directory with child directories already matched to customer identifier
• Generic to allow for re-utilization on separate applications requiring similar capability
• Decrease in user error on package generation
• Decrease on manual execution time and removal of the need for manual package validation

Beyond Software Tool Creation – Agile’s Broader Impact

The comparison extends beyond the technical aspects of software tool creation. It delves into how Agile methodologies transform the entire process, enhancing not just the technical task of package creation, but also impacting broader operational aspects like workflow efficiency, error reduction, and process adaptability.
Chapter 5: Functional Section

Workflow (CONOPS)

Figure 3 below demonstrates the utility execution at a pseudo-code level, illustrating the application of Agile methodology in developing a software tool for a systems engineering task. This approach aligns with the research objective of evaluating Agile methodologies in systems engineering processes beyond traditional software development. It progresses through script execution to the first graphical user interface (GUI) option of a ‘Folder Browser’ with decision catches. Based on end user input, the possible paths diverge. This divergence is critical in understanding how Agile methodologies enable adaptability and user responsiveness, a key aspect of this research. All end user actions are accompanied by a visual output to the end user requiring acknowledgment to proceed. If the end user opts to go through with the utility actions, a separate thread is initiated to capture execution times of both the zipping activity and the cleaning activity - if that option is also selected – for utility end reporting. Additionally, the cleaning activity has logging functionality as a supplemental validation check for the end user highlighting the Agile principle of continuous feedback and improvement.
Figure 3 – Utility Workflow
Utility Work Instruction

In addition to the workflow, a step-by-step work instruction was created to guide the end user through a set of actions to be completed using Graphical User Interface (GUI) elements. The inclusion of a comprehensive set of instructions reflects the user-centric approach of Agile methodologies and contributes to the research by demonstrating how user requirements are translated into functional specifications. These actions would allow for the selection of the parent directory of where the staged source sub-directories with the necessary content would be located. Initial instructions direct the end user to save the overall utility package to a folder local to their workstation and then proceed to instruct them on how to extract the necessary files for utility usage. This aspect of the study underscores how Agile methodologies can enhance the usability and effectiveness of tools developed for systems engineering tasks. The instructions included screenshots, seen in the two figures below, to aid in clarity of the written extraction and staging instructions.
Instructions for utility execution following extraction and staging, including warnings and decision point deviation, were presented for examination in the following sub-section.
Utility Execution Paths

The utility work instruction contained two paths, ‘Yes’ and ‘No’, for source cleaning that are documented in this Utility Execution Path section. The documentation of these paths serves as a practical example of how Agile methodologies facilitate flexible and user responsive tool development, addressing the research objective of evaluating Agile’s effectiveness in process improvement. Both available paths are preceded by the following informational and warning signs.

**Figure 6 – Utility Info & Caution**

- If sub-folders are not staged, the utility will indicate this to the end user via a pop-up and exit
- Re-running the utility will **FORCE** overwrite contents of previous zips with matching file names.
1. To execute the tool, simply double-click on the extracted **UtilityCaller** file and it will open the launcher GUI which includes a button for the **ZipUtility**.

*Figure 7 – Launcher Button*
2. Click the ‘AutoZip Utility’ button. This will launch a ‘File Browser Pop-up’.
3. Navigate to and select the ‘parent directory’ containing all the user’s sub-folders then click **OK**

![Figure 9 – Example of Parent Directory structure with sub-folders](image)

4. The user will now see a pop-up indicating that the utility is running and will now create the discrete zip packages. Click **OK**

![Figure 10 – Utility Running Pop-Up](image)
5. The utility will now indicate on the open console window that the zip actions are complete and that the end user should review the zip files that have been created (indicated in **green text**) as well as prompt the user to choose to remove the child directories or to leave them.

![Source Clean Confirmation](source-clean-confirmation.png)

*Figure 11 – User Prompt*
**Source Clean Confirmation – Selecting ‘Yes’**

1. Selecting ‘Yes’ will result in the following:
   a. Indicate to the user their choice – ‘Cleaning source files’. Click **OK**.

   ![Figure 12 – ‘Yes’ selected output]

   *Figure 12 – ‘Yes’ selected output*

   b. Indicate to the user that the directories have been removed and a ‘RemovedFiles.txt’ file has been created, prior to the utility exiting. Click **OK**.

   ![Figure 13 – Utility Exiting]

   *Figure 13 – Utility Exiting*

2. On completion, the utility will indicate the amount of time taken to create the zip packages and to clean-up the source data.

   ![Figure 14 – ‘Yes’ Timing Report]

   *Figure 14 – ‘Yes’ Timing Report*
3. Click **OK** to fully exit the utility.

4. Parent directory will now contain zips of the child directories as well as the RemovedFiles *.txt file.

*Figure 15 – ‘Yes’ Completed Utility Run*
5. *RemovedFiles* will list all of the removed child directories and their contents.

![RemovedFiles window](image)

**Figure 16 – ‘Yes’ Removed Files**
Source Clean Confirmation – Selecting ‘No’

1. Selecting ‘No’ will result in the following:
   a. Indicate to the user their choice – ‘Zip Utility exiting’. Click OK.

   ![Zip Utility exiting](image)
   
   *Figure 17 – ‘No’ selected output*

   b. Indicate to the user the amount of time elapsed to create zip packages.

   ![Utility Execution Time](image)
   
   *Figure 18 – ‘No’ Timing Report*

2. Click OK to fully exit the utility.
3. Parent directory will now contain the original child directories as well as corresponding zip packages.
Risk

Risk for this project was minimal even given the shortened timeline to accomplish the scope defined and the related tasks. The effective management of risks using Agile strategies highlights the methodology’s robustness in project management, contributing to the thesis’ exploration of Agile’s applicability in diverse operational contexts. The most common risks had an impact associated with schedule slip, but the mitigation strategies coupled with the Jira toolset allowed for an Agile approach to addressing these. Some may see the execution of the entire project by a single person as a high risk for single point of failure that could result in delays, however, having full visibility into each activity in conjunction with full knowledge of task completion capability aids in minimizing potential schedule risks if an appropriate level of slack is built in on initial planning. This is made possible through experience and knowledge on which tasks may require more time due to complexity and required skill level relative to the associated work. Additionally, with the project being self-managed and self-executed, priority reassignment faced no roadblocks allowing for rapid adjustment of tasks to effectively manage the overall case study. Building the script itself and the associated testing were allocated the most time and resources. The presented risk analysis table - Figure 20 below – provides a systematic overview of potential challenges and mitigation strategies, reinforcing the research value of this case study in demonstrating Agile’s practical application in managing project risks.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MC-1</td>
<td>Submit Topic</td>
<td></td>
<td></td>
<td>Risk of approval of topic submissions</td>
<td>low</td>
<td>high</td>
<td>medium</td>
<td>Ensure topic deliverable meets necessary criteria</td>
<td>-</td>
<td>high</td>
<td>medium</td>
</tr>
<tr>
<td>MC-4</td>
<td>Proposed Plan</td>
<td></td>
<td></td>
<td>Scope of effort fitting within reduced timeline</td>
<td>medium</td>
<td>high</td>
<td>medium</td>
<td>Utilize AGILE principals to streamline project tasks and set achievable goals within the given timeline</td>
<td>-</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>MC-6</td>
<td>Systems Architecture</td>
<td>MC-12</td>
<td>Data analysis (simulated)</td>
<td>Risk of not having enough representative data</td>
<td>low</td>
<td>low</td>
<td>low</td>
<td>Extrapolate simulated data set</td>
<td>-</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>MC-9</td>
<td>Implementation &amp; Test</td>
<td>MC-23</td>
<td>Write Script</td>
<td>Risk of scope creep or unexpected failures</td>
<td>low</td>
<td>high</td>
<td>medium</td>
<td>Adhere to the decided upon elements and schedule</td>
<td>low</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>MC-9</td>
<td>Implementation &amp; Test</td>
<td>MC-24</td>
<td>Add GUI elements (optional)</td>
<td>Additional scope may exceed timeline availability</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>Research PowerShell GUI options available and categorize by complexity</td>
<td>low</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>MC-9</td>
<td>Implementation &amp; Test</td>
<td>MC-10</td>
<td>Testing</td>
<td>Risk of not encountering all possible failure cases</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>Incorporate error logging and test functionality as new elements are added</td>
<td>-</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>MC-9</td>
<td>Implementation &amp; Test</td>
<td>MC-11</td>
<td>Documentation</td>
<td>Complexity Risk</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>Use confluence to build a wiki how-to that includes screenshots as visual aids</td>
<td>low</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>MC-14</td>
<td>Validation &amp; Verification</td>
<td>MC-15</td>
<td>Verification (Simulated)</td>
<td>Risk that automated method does not have a marked improvement</td>
<td>low</td>
<td>high</td>
<td>medium</td>
<td>Test early and ensure execution metrics are analyzed</td>
<td>-</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>MC-17</td>
<td>Final Report</td>
<td>MC-26</td>
<td>Import final report into Confluence</td>
<td>Risk of poor formatting and legibility</td>
<td>medium</td>
<td>high</td>
<td>high</td>
<td>Include buffer time in schedule for peer/advisor review</td>
<td>low</td>
<td>high</td>
<td>medium</td>
</tr>
</tbody>
</table>

*Figure 20 – Risk Analysis Table*
Strength, Weaknesses, Opportunities, and Threats (SWOT) analysis was utilized to explore the potential breadth of the utility in relation to its applicability. This analysis is pivotal in assessing the strengths, weaknesses, opportunities, and threats of the Agile methodology in systems engineering, providing a comprehensive view of Agile’s impact beyond software development. SWOT analysis employs a diagnostic method to identify significant elements influencing the success or failure of a strategy or system. SWOT has a wide spectrum of applications for conducting a strategic assessment of organizations in a variety of domains studied by different researchers. (Noorani, Zamani, Alenezi, Shameem, & Singh, 2022). Three representative scenarios were examined: Discrete Package generation for the PM&S team, Law/Tax Firm, and Education. The primary differences seen across the examples were in the weaknesses and opportunities blocks, although there were small deltas in the opportunities block as well. SWOT analysis for the three different implementations can be seen below.
### Product Management & Sustainment

#### Strengths
- Automated
- Easy to Use
- Multiple use cases
- Quality
- Execution time
- Preservation of source data directories
- Only sub-directories are zipped (e.g. any other files present are preserved)
- Not limited to local machine (e.g. network share locations are accessible)

#### Weaknesses
- UI elements are rudimentary
- Batch caller must be staged in same location as main PowerShell Utility
- Selections are final on first choice
- All or nothing based on which sub-directories are present
- Matching directory yields forced overwrite

#### Opportunities
- Improved GUI operations
- Improved logging
- Improved metrics capture
- Improved debug capability
- Improved error checking (user checks)
  - Matched directory discovery
- Add single sub-directory option
- Add exclusion capability

#### Threats
- Non-compiled code can be re-written easily
- End user can select wrong location to zip
- Overwrite risk
**Strengths**
- Automated
- Easy to Use
- Multiple use cases
- Quality
- Execution time
- Preservation of source data directories
- Only sub-directories are zipped (e.g. any other files present are preserved)
- Not limited to local machine (e.g. network share locations are accessible)

**Weaknesses**
- UI elements are rudimentary
- Batch caller must be staged in same location as main PowerShell Utility
- Selections are final on first choice
- All or nothing based on which sub-directories are present
- Matching directory yields forced overwrite
- Lack of encryption capability
- Sub-directories require unique filenames

**Opportunities**
- Improved GUI operations
- Improved logging
- Improved metrics capture
- Improved debug capability
- Improved error checking (user checks)
  - Matched directory discovery
- Add single sub-directory option
- Include encryption capabilities for personal data

**Threats**
- Non-compiled code can be re-written easily
- End user can select wrong location to zip
- Personal data at risk

**Law/Tax Firm**
Strengths
- Automated
- Easy to Use
- Multiple use cases
- Quality
- Execution time
- Preservation of source data directories
- Only sub-directories are zipped (e.g. any other files present are preserved)
- Not limited to local machine (e.g. network share locations are accessible)

Weaknesses
- UI elements are rudimentary
- Batch caller must be staged in same location as main PowerShell Utility
- Selections are final on first choice
- All or nothing based on which sub-directories are present
- Matching directory yields forced overwrite
- Sub-directories require unique filenames

Opportunities
- Improved GUI operations
- Improved logging
- Improved metrics capture
- Improved debug capability
- Improved error checking (user checks)
- Matched directory discovery
- Add single sub-directory option
- Automatic upload of completed zips to server environment (e.g. Canvas)

Threats
- Non-compiled code can be re-written easily
- End user can select wrong location to zip

Education

50
Chapter 6: Results

The outcomes of this thesis closely align with the predefined research objectives. Notably, the development and implementation of the PowerShell utility for this case study showcased a tangible enhancement in systems engineering processes, effectively fulfilling the primary objective of improving efficiency through Agile methodologies. The substantial reductions in task completion time and error rates directly fulfill the objective of introducing more effective operational methods in systems engineering. This not only affirms the practical value of the utility but also lends support to the theoretical framework proposed in this research. Moreover, the successful application of the PowerShell utility for the case study serves as a prime example of Agile methodologies' adaptability in a complex engineering environment. The substantial reductions in task completion time and error rates directly fulfill the objective of introducing more effective operational methods in systems engineering. This not only affirms the practical value of the utility but also lends support to the theoretical framework proposed in this research. Moreover, the successful application of the PowerShell utility in this case study serves as a pivotal example of Agile methodologies' adaptability in a complex engineering environment. This achievement directly addresses the objective of exploring Agile methodologies beyond traditional software development contexts, thereby contributing to a broader understanding and application of these methodologies in varied industries.

In contrast, the utility outperformed manual processes in speed, accuracy, and consistency, highlighting the benefits of automation in systems engineering. This result aligns with the literature's identified need for more efficient, error-resistant systems engineering processes. The broader impact of these findings is substantial. They illustrate the practical application of Agile methodologies in complex engineering environments, addressing the
literature gaps concerning Agile's adaptability and efficiency. This case study suggests potential scalability and adaptability of these methods across various sectors. The results of data analysis indicate that the inclusion of the PowerShell utility in the case study's systems engineering processes not only enhances operational efficiency but also contributes significantly to the ongoing development of Agile methodologies in systems engineering.

**Metrics**

The completion of tasks and sub-tasks mostly adhered to the initially proposed schedule and minor deltas were typically related to the re-organization of sub-tasks to be worked on. This was often dictated by available time to work on these different parts or the need to vary the type of work (e.g., take a break from coding to give time to re-frame an encountered roadblock). This remains in line with the agile value of not placing too high of an emphasis on a plan, but rather responding to change. This alignment underscores the research finding that Agile methodologies can enhance flexibility and responsiveness in project execution, a key contribution to the literature on Agile in systems engineering. Progress was reported weekly during a zoom call and always available via the Jira board which tracked tasks through their states of To-Do, In Progress, or Completed. The breakdown of the distribution of tasks and sub-tasks can be seen in Figure 21 below and the related Data Table.
Data Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Issues</th>
<th>Percentage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>20</td>
<td>76%</td>
</tr>
<tr>
<td>Testing</td>
<td>8</td>
<td>30%</td>
</tr>
<tr>
<td>Scripting</td>
<td>4</td>
<td>15%</td>
</tr>
<tr>
<td>Simulation</td>
<td>2</td>
<td>7%</td>
</tr>
</tbody>
</table>

The categories are represented by labels that were defined at the start of the project and assigned to each task and sub-task. From the chart, it is clear to see that the majority of the tasks are related to Documentation. Additionally, Figure 22 indicates that the Documentation tasks also accounted for the most time spent, followed by Testing and Scripting. Six hours of the
Documentation label were directly related to MC-3: Literature Review, which was the task that encompassed all reference inquiries related to coding or capability analysis.

![Workload Pie Chart Report](image)

**Figure 22 – Workload by label**

### Data Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Hours</th>
<th>Percentage %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>22</td>
<td>55%</td>
</tr>
<tr>
<td>Testing</td>
<td>8</td>
<td>21%</td>
</tr>
<tr>
<td>Scripting</td>
<td>7</td>
<td>19%</td>
</tr>
<tr>
<td>Simulation</td>
<td>1</td>
<td>3%</td>
</tr>
</tbody>
</table>
Data Analysis

In evaluating the effectiveness of the PowerShell utility, a comprehensive data extrapolation model was employed instead of a simulation. To establish this model, a dataset was systematically collected, encompassing tasks performed by individuals with varying workloads and experience levels. The data, sourced from historical records of task completion times, incorporated a wide range of conditions, ensuring the formation of a robust and representative sample. This approach facilitated a more accurate approximation of average completion times across various scenarios.

The analysis employed statistical methods such as regression analysis to unravel the relationships between variables like experience level and task completion time. Additionally, descriptive statistics were employed to concisely summarize the data, presenting a clear snapshot of task performance across different conditions. The model's output, as depicted in Figures 23 to 25, acts as an aggregated summary of these extrapolated datasets. The choice to term this a data extrapolation model rather than a simulation is deliberate, providing a more precise characterization of the methodology employed to analyze the effectiveness of the PowerShell utility in various operational scenarios.

The data analysis consisted of a preliminary sampling of aggregate historical data – estimated time it had taken to complete the zip packages manually in the past. From this initial data set, the values were extrapolated out to include more samples to create the model in excel. The extrapolated data sets included the following aspects of the current implementation: manual discovery of relevant data files, manual creation of discrete zip packages, manual re-name of file (if needed) to match customer identifier, and manual validation of completed package. The
increased data set yielded a better approximation of average times to complete these zips. The model was necessary to incorporate the variability of the tasks being performed by different individuals with different workloads and experience levels. The extrapolated datasets can be seen in Figure 23, Figure 24, and Figure 25 below.

<table>
<thead>
<tr>
<th>Total packages</th>
<th>150</th>
</tr>
</thead>
<tbody>
<tr>
<td># of files/package</td>
<td></td>
</tr>
<tr>
<td>Range</td>
<td>5 to 20</td>
</tr>
<tr>
<td>Aggregate Average Minutes</td>
<td>118</td>
</tr>
</tbody>
</table>

*Figure 23 – Total Extrapolated Data Set (Summary)*
<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Time in minutes (rounded)</th>
<th>Time in minutes (rounded)2</th>
<th>Time in minutes (rounded)3</th>
<th>Time in minutes (rounded)4</th>
<th>Time in minutes (rounded)5</th>
<th>Time in minutes (rounded)6</th>
<th>Time in minutes (rounded)7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>121</td>
<td>46</td>
<td>201</td>
<td>44</td>
<td>76</td>
<td>55</td>
<td>76</td>
</tr>
<tr>
<td>2</td>
<td>176</td>
<td>51</td>
<td>54</td>
<td>87</td>
<td>71</td>
<td>83</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>175</td>
<td>52</td>
<td>293</td>
<td>82</td>
<td>64</td>
<td>169</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>138</td>
<td>70</td>
<td>219</td>
<td>23</td>
<td>76</td>
<td>215</td>
<td>59</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td>43</td>
<td>71</td>
<td>39</td>
<td>89</td>
<td>67</td>
<td>69</td>
</tr>
<tr>
<td>6</td>
<td>163</td>
<td>59</td>
<td>169</td>
<td>67</td>
<td>61</td>
<td>218</td>
<td>68</td>
</tr>
<tr>
<td>7</td>
<td>105</td>
<td>87</td>
<td>129</td>
<td>97</td>
<td>82</td>
<td>101</td>
<td>22</td>
</tr>
<tr>
<td>8</td>
<td>180</td>
<td>111</td>
<td>72</td>
<td>55</td>
<td>87</td>
<td>284</td>
<td>59</td>
</tr>
<tr>
<td>9</td>
<td>167</td>
<td>92</td>
<td>53</td>
<td>88</td>
<td>60</td>
<td>53</td>
<td>71</td>
</tr>
<tr>
<td>10</td>
<td>121</td>
<td>83</td>
<td>237</td>
<td>97</td>
<td>73</td>
<td>169</td>
<td>86</td>
</tr>
<tr>
<td>11</td>
<td>157</td>
<td>83</td>
<td>156</td>
<td>16</td>
<td>99</td>
<td>289</td>
<td>48</td>
</tr>
<tr>
<td>12</td>
<td>143</td>
<td>68</td>
<td>62</td>
<td>19</td>
<td>87</td>
<td>55</td>
<td>84</td>
</tr>
<tr>
<td>13</td>
<td>148</td>
<td>44</td>
<td>293</td>
<td>66</td>
<td>98</td>
<td>210</td>
<td>28</td>
</tr>
<tr>
<td>14</td>
<td>79</td>
<td>32</td>
<td>274</td>
<td>94</td>
<td>80</td>
<td>128</td>
<td>85</td>
</tr>
<tr>
<td>15</td>
<td>158</td>
<td>116</td>
<td>265</td>
<td>78</td>
<td>77</td>
<td>134</td>
<td>77</td>
</tr>
<tr>
<td>16</td>
<td>83</td>
<td>58</td>
<td>229</td>
<td>59</td>
<td>99</td>
<td>54</td>
<td>56</td>
</tr>
<tr>
<td>17</td>
<td>168</td>
<td>98</td>
<td>54</td>
<td>98</td>
<td>64</td>
<td>185</td>
<td>53</td>
</tr>
<tr>
<td>18</td>
<td>45</td>
<td>103</td>
<td>168</td>
<td>64</td>
<td>79</td>
<td>164</td>
<td>60</td>
</tr>
<tr>
<td>19</td>
<td>86</td>
<td>32</td>
<td>137</td>
<td>70</td>
<td>78</td>
<td>244</td>
<td>60</td>
</tr>
<tr>
<td>20</td>
<td>110</td>
<td>43</td>
<td>191</td>
<td>27</td>
<td>66</td>
<td>133</td>
<td>87</td>
</tr>
<tr>
<td>21</td>
<td>89</td>
<td>52</td>
<td>263</td>
<td>66</td>
<td>95</td>
<td>101</td>
<td>62</td>
</tr>
<tr>
<td>22</td>
<td>103</td>
<td>80</td>
<td>67</td>
<td>82</td>
<td>86</td>
<td>56</td>
<td>39</td>
</tr>
<tr>
<td>23</td>
<td>79</td>
<td>56</td>
<td>182</td>
<td>64</td>
<td>98</td>
<td>118</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>101</td>
<td>79</td>
<td>108</td>
<td>27</td>
<td>67</td>
<td>225</td>
<td>52</td>
</tr>
<tr>
<td>25</td>
<td>103</td>
<td>64</td>
<td>289</td>
<td>91</td>
<td>76</td>
<td>231</td>
<td>20</td>
</tr>
</tbody>
</table>

Average Time: 123, 68, 169, 64, 80, 150, 57

*Figure 24 – Extrapolated Data Set Runs (a)*
<table>
<thead>
<tr>
<th>Time in minutes (rounded)8</th>
<th>Time in minutes (rounded)9</th>
<th>Time in minutes (rounded)10</th>
<th>Time in minutes (rounded)11</th>
<th>Time in minutes (rounded)12</th>
<th>Time in minutes (rounded)13</th>
<th>Time in minutes (rounded)14</th>
<th>Time in minutes (rounded)15</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>77</td>
<td>510</td>
<td>118</td>
<td>60</td>
<td>89</td>
<td>140</td>
<td>123</td>
</tr>
<tr>
<td>77</td>
<td>81</td>
<td>443</td>
<td>108</td>
<td>132</td>
<td>112</td>
<td>88</td>
<td>63</td>
</tr>
<tr>
<td>66</td>
<td>33</td>
<td>334</td>
<td>36</td>
<td>143</td>
<td>137</td>
<td>77</td>
<td>69</td>
</tr>
<tr>
<td>92</td>
<td>38</td>
<td>399</td>
<td>120</td>
<td>65</td>
<td>120</td>
<td>141</td>
<td>71</td>
</tr>
<tr>
<td>88</td>
<td>81</td>
<td>570</td>
<td>98</td>
<td>109</td>
<td>120</td>
<td>185</td>
<td>113</td>
</tr>
<tr>
<td>87</td>
<td>72</td>
<td>561</td>
<td>74</td>
<td>81</td>
<td>141</td>
<td>98</td>
<td>124</td>
</tr>
<tr>
<td>109</td>
<td>95</td>
<td>513</td>
<td>56</td>
<td>111</td>
<td>109</td>
<td>138</td>
<td>174</td>
</tr>
<tr>
<td>68</td>
<td>58</td>
<td>221</td>
<td>111</td>
<td>53</td>
<td>149</td>
<td>174</td>
<td>157</td>
</tr>
<tr>
<td>73</td>
<td>80</td>
<td>544</td>
<td>91</td>
<td>159</td>
<td>124</td>
<td>183</td>
<td>123</td>
</tr>
<tr>
<td>97</td>
<td>69</td>
<td>347</td>
<td>143</td>
<td>47</td>
<td>127</td>
<td>73</td>
<td>161</td>
</tr>
<tr>
<td>90</td>
<td>36</td>
<td>172</td>
<td>107</td>
<td>137</td>
<td>99</td>
<td>60</td>
<td>92</td>
</tr>
<tr>
<td>61</td>
<td>75</td>
<td>325</td>
<td>65</td>
<td>124</td>
<td>150</td>
<td>76</td>
<td>149</td>
</tr>
<tr>
<td>65</td>
<td>77</td>
<td>116</td>
<td>38</td>
<td>86</td>
<td>120</td>
<td>96</td>
<td>61</td>
</tr>
<tr>
<td>61</td>
<td>87</td>
<td>109</td>
<td>116</td>
<td>174</td>
<td>97</td>
<td>122</td>
<td>108</td>
</tr>
<tr>
<td>113</td>
<td>74</td>
<td>97</td>
<td>131</td>
<td>169</td>
<td>89</td>
<td>115</td>
<td>115</td>
</tr>
<tr>
<td>93</td>
<td>55</td>
<td>235</td>
<td>89</td>
<td>98</td>
<td>138</td>
<td>137</td>
<td>97</td>
</tr>
<tr>
<td>104</td>
<td>40</td>
<td>481</td>
<td>48</td>
<td>146</td>
<td>148</td>
<td>104</td>
<td>133</td>
</tr>
<tr>
<td>106</td>
<td>70</td>
<td>179</td>
<td>133</td>
<td>97</td>
<td>107</td>
<td>150</td>
<td>131</td>
</tr>
<tr>
<td>71</td>
<td>94</td>
<td>253</td>
<td>85</td>
<td>95</td>
<td>97</td>
<td>128</td>
<td>155</td>
</tr>
<tr>
<td>73</td>
<td>84</td>
<td>360</td>
<td>71</td>
<td>165</td>
<td>132</td>
<td>92</td>
<td>109</td>
</tr>
<tr>
<td>68</td>
<td>38</td>
<td>328</td>
<td>113</td>
<td>95</td>
<td>140</td>
<td>135</td>
<td>128</td>
</tr>
<tr>
<td>95</td>
<td>77</td>
<td>95</td>
<td>145</td>
<td>88</td>
<td>73</td>
<td>66</td>
<td>153</td>
</tr>
<tr>
<td>117</td>
<td>49</td>
<td>544</td>
<td>52</td>
<td>90</td>
<td>145</td>
<td>172</td>
<td>152</td>
</tr>
<tr>
<td>119</td>
<td>51</td>
<td>507</td>
<td>58</td>
<td>153</td>
<td>93</td>
<td>100</td>
<td>85</td>
</tr>
<tr>
<td>83</td>
<td>40</td>
<td>575</td>
<td>74</td>
<td>74</td>
<td>143</td>
<td>99</td>
<td>159</td>
</tr>
<tr>
<td>86</td>
<td>65</td>
<td>353</td>
<td>91</td>
<td>110</td>
<td>120</td>
<td>118</td>
<td>120</td>
</tr>
</tbody>
</table>

*Figure 25 – Extrapolated Data Set Runs (b)*
Multiple factors were manipulated in the generated datasets to provide a broad spectrum of variability to simulate real world differences in execution strategy, based on apparent skill level, and potential outside factors, such as computer processing capability (likely negligible assuming current industry or commercial standards on issued computers) or availability due to workload and prioritization.

Once the utility was operational, a series of test runs were executed to capture the execution times of the automated capabilities – discrete zip package generation and source data cleaning, with the added feature of automated ‘self-validation’. Since the automation yielded consistent, repeatable results, the sample size of the test data set was able to be significantly smaller while still being an accurate representation of the average timing. Testing was limited to being performed on a single device, based on the assumption of the negligible delta attributed to hardware selection as referenced during the simulated data capture. Results of the sampled utility execution runs can be seen in Figure 26 below. Basic random file generation logic was utilized to create the file samples, due to the absence of a readily available repository of varying file sizes. For the purpose of the study, the type of file was deemed not important in comparison to the number of files or overall file size as the compression action is not impacted by file type. An example of the command utilized to generate the random files at a given size (with the file size passed in bytes) is as follows:

```
fsutil file createnew .\1gb_test.txt 1073741824
```
<table>
<thead>
<tr>
<th>Test Run</th>
<th># of Packages</th>
<th>Time in seconds (rounded)</th>
<th>Time in seconds (rounded)2</th>
<th>Time in seconds (rounded)3</th>
<th>Aggregate Average seconds</th>
<th>Average Clean Time seconds</th>
<th>Total File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>150</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>41</td>
<td>6</td>
<td>650 MB</td>
</tr>
<tr>
<td>2</td>
<td>150</td>
<td>38</td>
<td>45</td>
<td>37</td>
<td>40</td>
<td>6</td>
<td>650 MB</td>
</tr>
<tr>
<td>3</td>
<td>150</td>
<td>34</td>
<td>39</td>
<td>41</td>
<td>38</td>
<td>5</td>
<td>650 MB</td>
</tr>
<tr>
<td>4</td>
<td>150</td>
<td>39</td>
<td>34</td>
<td>43</td>
<td>39</td>
<td>6</td>
<td>650 MB</td>
</tr>
<tr>
<td>5</td>
<td>150</td>
<td>44</td>
<td>46</td>
<td>31</td>
<td>40</td>
<td>6</td>
<td>650 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Run</th>
<th># of Packages</th>
<th>Time in seconds (rounded)</th>
<th>Time in seconds (rounded)2</th>
<th>Time in seconds (rounded)3</th>
<th>Aggregate Average seconds</th>
<th>Average Clean Time seconds</th>
<th>Total File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>450 KB</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>450 KB</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>450 KB</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>450 KB</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>450 KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Run</th>
<th># of Packages</th>
<th>Time in seconds (rounded)</th>
<th>Time in seconds (rounded)2</th>
<th>Time in seconds (rounded)3</th>
<th>Aggregate Average seconds</th>
<th>Average Clean Time seconds</th>
<th>Total File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>1 GB</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>25</td>
<td>25</td>
<td>26</td>
<td>25</td>
<td>2</td>
<td>3 GB</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>26</td>
<td>26</td>
<td>25</td>
<td>26</td>
<td>1</td>
<td>3 GB</td>
</tr>
</tbody>
</table>

*Figure 26 – Utility Data (Sampled)*
The increased throughput and decreased execution time of the automated utility route over the manual execution route is extreme. The aggregate average minutes of execution time for 150 discrete packages with the number of files/package range of 5 to 20 was **118 minutes** ranging from 16 minutes to 575 minutes with a median value of 94 minutes. In comparison, the average aggregate time for deliverables with the same parameters utilizing the automated utility dropped to **40 seconds** for creation of the zips and **6 seconds** to clean the source files yielding a total execution time of **less than 1 minute**. This is an improvement of approximately **117 minutes or 99%**. More impressive is the percentage improvement, **15,291%**, on the total amount of packages that could be generated by the automated utility in the same amount of time 150 packages could be generated via the manual method. Furthermore, it was interesting to note that the combination of the overall file size and number of files is what actually appeared to drive the execution time, with a greater time increase noticeable if the package count was higher. This was likely attributed to the actions being single-threaded and serial. An example of this is 150 packages with a total file size of 650 MB taking approximately 40 seconds and 6 seconds to zip and clean, respectively, versus 1 package with a larger file size of 1GB taking 8 seconds and 1 second to zip and clean, respectively. Further testing shows that 1 package with a total file size of 3GB versus 3 packages with an equal total file size yielded execution times that were only 1 second apart. The observed disparity scales with the number of packages.
Results and Comparative Analysis of Project Management Methodologies

This study's results provide compelling insights into the effectiveness of Agile methodologies compared to traditional project management approaches in the context of systems engineering. The research focused on the development of a utility for automating the creation of zip packages, a task that was originally performed manually, serving as a practical case study to evaluate the research objectives.

1. Operational Efficiency Improvement

The Agile methodology's impact on operational efficiency was significant. The automated solution developed using Agile principles resulted in a dramatic decrease in the time required to create zip packages. Whereas the traditional approach took over an hour of uninterrupted work, the Agile-developed utility reduced this time drastically, demonstrating a 99% improvement in execution time. This efficiency gain is a direct testament to Agile's iterative, responsive, and user-focused approach, which enabled rapid development and refinement of the utility.

2. Error Reduction

Another critical outcome was the reduction in error rates. The manual process was prone to human errors due to its repetitive and labor-intensive nature. The automated utility, however, introduced a high level of accuracy and consistency, virtually eliminating manual errors. This error reduction underscores Agile's effectiveness in creating reliable and user-friendly tools, even in complex systems engineering tasks.
3. Agile vs. Traditional Project Management

The study's comparative analysis highlights the stark differences between Agile and traditional project management methodologies. Traditional methods, while structured and predictable, often lack the flexibility to adapt to changing user needs and rapid iterations. In contrast, Agile methodologies, with their emphasis on continuous improvement and stakeholder involvement, facilitated a more dynamic and adaptive development process. This adaptability was crucial in tailoring the utility to effectively meet user requirements and improve the process efficiently.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Traditional Project Management</th>
<th>Agile Project Management</th>
<th>Case Study Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure</td>
<td>Rigid, linear phases (Waterfall model)</td>
<td>Iterative, incremental approach</td>
<td>Development of the utility in iterative cycles, allowing for adjustments and improvements at each stage</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Limited flexibility to changes</td>
<td>Highly flexible to changes and new requirements</td>
<td>Adapted to changing user requirements during utility development</td>
</tr>
<tr>
<td>Feedback Mechanism</td>
<td>Feedback mostly at the end of each phase</td>
<td>Continuous feedback throughout development</td>
<td>Regular feedback from users and stakeholders integrated into each iteration</td>
</tr>
<tr>
<td>Stakeholder Engagement</td>
<td>Initial stages involve stakeholders, later stages less so</td>
<td>Consistent stakeholder engagement throughout</td>
<td>Continuous engagement with the project advisor and users throughout the development process</td>
</tr>
<tr>
<td>Risk Management</td>
<td>Risk assessed early, with limited adaptability to new risks</td>
<td>Continuous risk assessment and adaptability</td>
<td>Agile approach allowed for quick response to challenges, minimizing potential delays</td>
</tr>
<tr>
<td>Documentation</td>
<td>Extensive initial documentation, with fewer updates</td>
<td>Evolving documentation with regular updates</td>
<td>Documentation updated regularly in Confluence, reflecting ongoing changes and decisions</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Efficiency dependent on initial planning, less adaptable</td>
<td>Highly efficient with adaptability to changes</td>
<td>The automated solution reduced the time for creating zip packages from over an hour to just minutes</td>
</tr>
</tbody>
</table>

*Table 1 – Traditional vs Agile Comparison*
Chapter 7: Conclusion

This thesis makes significant contributions to the literature in systems engineering and Agile methodologies. The creation and implementation of the PowerShell utility, particularly within the context of this case study, showcase significant enhancements in operational efficiency and error reduction, effectively addressing critical gaps identified in current research. The insights derived from this case study not only deepen the academic understanding of Agile methodologies in non-software contexts but also offer practical insights for their application in diverse engineering scenarios.

The research objectives, which were specifically targeted at evaluating the impact of Agile methodologies and digital automation tools in systems engineering, have been successfully fulfilled. The proven effectiveness of the PowerShell utility in automating routine tasks, substantiated by quantitative analysis, directly meets the literature's requirement for empirical evidence in this realm. Additionally, the case study's insights into the practical implementation of these methodologies significantly contribute to bridging the existing gaps in academic publications, particularly in illustrating Agile's scalability and adaptability in complex engineering systems.

The case study examination clearly indicates a successful merging of systems engineering principles with the Agile methodology resulting in a clear example of digital transformation to drive process improvement. The use of tools such as Jira and Confluence lent a level of ease to project management and organization due to the enhanced collaboration capabilities coupled with the interconnected real-time reporting. This practical application of Agile in a systems engineering context demonstrates its effectiveness beyond the realm of software development,
providing a valuable case study for the application of Agile methodologies in broader operational processes.

It is clear why many software projects use Agile and track efforts via sprints with additional tasks being managed via a backlog. There are plenty of commercial products available that offer similar or even greater capability, but for the scope of this study, the tools used more than met the need. A good example is the clear representation of scheduling without the potentially steep learning curve of utilizing other tools such as Microsoft Project or manually building out a timeline via a Gantt chart. Confluence provided a centrally located delivery vehicle for all aspects of the study – from high level planning, to work instructions, and even meeting notes with tracked action items.

The case study presented in this research was meticulously designed to explore the practical application of Agile methodologies within the realm of systems engineering. This study goes beyond merely aligning with the core subjects of Systems Engineering and delves into the empirical assessment of Agile's effectiveness in enhancing engineering processes. The integration of theoretical concepts with practical implementation in this research exemplifies how the case study contributes to the overarching research objective of evaluating Agile's impact in systems engineering. This approach not only demonstrates a successful application of Agile principles, but also extends their reach beyond typical domains.

A notable aspect of this research was the utilization of Agile’s retrospective process at the end of each sprint and the completion of the case study. This method was instrumental in assessing the effectiveness of Agile methodologies from both a technical and organizational perspective. The reflective nature of these retrospectives aligns with rigorous academic research
methodologies, providing crucial insights into the areas of continuous improvement and process optimization, which are essential components in academic research.

The analysis conducted in this case study revealed potential areas for enhancement, particularly in the realm of generating more in-depth metrics, such as completion time estimates. These insights are valuable additions to the academic discourse, highlighting new avenues for future research in Agile methodologies within systems engineering contexts. They suggest that larger or team-based projects could significantly benefit from such detailed analyses, enabling more effective project management through trend analysis and resource allocation adjustments.

The summary page in Jira exemplified a powerful tool for providing a comprehensive overview of task statuses and workloads, reinforcing the value of Agile tools in managing complex engineering tasks.

In conclusion, this research has made a substantial contribution to the academic literature by demonstrating a real-world application of Agile methodologies in systems engineering, resulting in remarkable improvements in operational efficiency and execution time. This improvement of 15391% in package generation and 99% in execution time for a repeatable process not only signifies substantial savings in man-hours and cost, but also reinforces the broader applicability and potential of Agile methodologies. The case study serves as a significant model for future research and application in this field, expanding the current understanding of Agile methodologies and their impact on systems engineering processes.
Future Work and Applications

Execution of the thesis and case study successfully demonstrated satisfaction of the requirements set forth for the specific use case presented by the Product Management & Sustainment team. A key accomplishment of the thesis not tied to the requirements but implemented in the design phase following careful analysis so as not to cause delays that could be attributed to scope creep, was the modular design of the utility and the inclusion of a GUI utility launcher. The launcher can be easily modified to include buttons to launch any future, stand-alone utilities. There are no restrictions requiring the original AutoZip utility to be modified to include the capability to launch different utilities that accomplish different desired functions; the launcher would be modified to build a respective button for each new utility and the overall package would be updated to include the necessary files, respectively.

Another design element which opened the breadth of the utility to be applicable for other industries and use cases was the generic implementation. The utility was not locked down with hard variables for the “customer identifier” or specific file paths that would require re-work for any subsequent deliveries for different use cases such as tax firm customer packages, student assignment submissions requiring discrete package deliverables, or any efforts that may have required the creation of multiple zip packages. Both of these design elements served to meet one of the objectives of the case study, which was to explore the scalability and adaptability of digital tools in diverse systems engineering processes, thereby contributing to the research field by demonstrating practical applications of these concepts.

Many companies already have initiatives related to Digital Transformation or are in the process of adopting processes that support this. Another key objective examined in this thesis
was to analyze the reduction in error rates associated with the adoption of automation tools in routine engineering tasks. Arguably, automation, the key benefit of the developed utility for this case study, is becoming far more prevalent due to the significant benefits it presents for increased consistency and reduced rework. The case study was able to quantitatively compare error rates between the manual process of creating zip packages versus the automated process. This further strengthens the shift towards automation due to the visible impact on enhanced accuracy and a reduction in associated errors in systems engineering processes, providing empirical evidence to support this transition in the field. Tasks that can be categorized as repeatable and constant are the most applicable. The case study focused on a specific task, automated zip package creation, but further insights can be derived on the scalability of not only the PowerShell utility examined, but also other script-based process improvements for a myriad of similar tasks within systems engineering, further highlighting the significant potential of a broader and more diverse application of similar tools. This case study also serves to decrease the gap in research on these types of process improvements. In terms of the methodology employed in the case study, Agile has traditionally found its application in software development, but the principles, when applied correctly, are beginning to indicate a much broader usability. The case study highlighted the use of Agile methodologies for the development and implementation of the utility. Furthermore, through the lens of the case study, we were able to see how Agile practices can play a pivotal role in effective project management, specifically by enabling iterative development, adapting to changing requirements, and promoting collaborative teamwork. Agile provides the incremental way of working that makes it possible to smoothly make the trade-offs converge as we move from upfront phases to more detailed design phases and make them flow across the whole organization (Alt & Le Mouëlli, 2023).
Future investigations could center on the application of Agile methodologies and automation tools within distinct sectors, such as renewable energy or the automotive industry. Conducting comparative analyses across various industries would provide insights into the universal applicability of Agile principles in diverse engineering contexts. Thorough exploration is warranted for the potential integration of cutting-edge technologies like artificial intelligence and machine learning into systems engineering automation tools. Subsequent research could focus on developing sophisticated predictive tools to significantly enhance operational efficiency in systems engineering. Additionally, conducting an in-depth study of the organizational and cultural transformations necessary for implementing Agile methodologies in conventionally structured industries could offer valuable insights into the broader implications of these methodologies. Finally, an academic examination of the incorporation of Agile methodologies within systems engineering education and curriculum development is essential for shaping future educational strategies and training programs in this field.
References


Appendix: Zip Utility PowerShell Code

# Title: Zip Utility
# Author: Anthony Maiello
# Scope: Utility will automatically create discrete zip packages on end user defined source date (via user prompt)
#
# Variable Declaration

$script:userFolder = ""
$script:path = ""
$script:destination = ""
$script:source = ""
$script:fullSource = ""
$script:clean = $FALSE

[timespan]$script:zipTime = 0
[timespan]$script:cleanTime = 0

$choice = ""
$Version = "1.5"
# Assembly Declaration

# Add in zip capability

add-type -assembly "system.io.compression.filesystem"

# Add in forms capability

Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

# Add in message box with choice inputs

Add-Type -AssemblyName PresentationCore,PresentationFramework

# Functions

function Show-MessageBoxDialog()
{

    [CmdletBinding(PositionalBinding = $True)]

    Param (  

        [Parameter(Mandatory = $True, Position = 0)]

        [String]$Message,


function AutoZip
{

[CmdletBinding(PositionalBinding = $False)]

Param (  

[Parameter(Mandatory = $False, Position = 0)]
[String]$path = $script:path,

[Parameter(Mandatory = $False, Position = 1)]
[String]$destination = $script:destination,

[Parameter(Mandatory = $False, Position = 2)]
[Object]$source = [Object]$script:source,

[Parameter(Mandatory = $False, Position = 3)]

return [System.Windows.Forms.MessageBox]::Show($Message, $WindowTitle)
}
# Folder Browser pop-up

```powershell
$FolderBrowser = New-Object System.Windows.Forms.FolderBrowserDialog
$FolderBrowser.Description = 'Select the parent directory'
$FolderBrowser.ShowNewFolderButton = $False
$result = $FolderBrowser.ShowDialog((New-Object System.Windows.Forms.Form -Property @{TopMost = $true })))
```

# If Folder is selected and OK is clicked, set variable to user input, else exit the script

```powershell
if ($result -eq [Windows.Forms.DialogResult]::OK)
{
    $path = $FolderBrowser.SelectedPath
}
else
{
    exit -1
}
```

# Create $source variable from directory names

```powershell
$source = Get-ChildItem -Path "$path" -Directory
```

# Check if parent directory has sub-folders staged

```powershell
if ($null -eq $source) {
```
# Inform the user that parent directory was empty then exit

Show-MessageBoxDialog -Message “Parent directory does not contain sub-folders. Investigate and re-try.”

-WindowTitle “Zip Utility exiting”

exit 1

# Determine path and store contents of sub-directories for possible use in clean up function reporting

$script:fullSource = Get-ChildItem -Path “$path” -Recurse

$script:path = $path

$script:source = $source

# Create timer variable

[timespan]$script:zipTime = Measure-Command {

  # Inform the user what is happening

  Show-MessageBoxDialog -Message “The utility will now create discrete zip packages based on directory files”…”

  -WindowTit“e "Zip Utility runn”ng"

  # Main loop
Foreach ($s in $source) {

    $destination = Join-path -path $path -ChildPath "$(s.name).ip"
    $destinationFile = split-path $destination -Leaf

    Write-Host 'n$destinationFile' n -ForegroundColor Green

    # Remove files on loop through if they exist
    If(Test-path $destination) {
        Remove-item $destination
    }

    # Create the zip packages
    [io.compression.zipfile]::CreateFromDirectory($s.fullname, $destination)
}

Write-Host 'Zip actions comple-e - review for completeness' 'n' -ForegroundColor Cyan

function SourceClean
{

    [CmdletBinding(PositionalBinding = $False)]

    Param (}
[Parameter(Mandatory = $False, Position = 0)]
[String]$path = $script:path,

[Parameter(Mandatory = $False, Position = 1)]
[String]$choice = "",

[Parameter(Mandatory = $False, Position = 2)]
[Object]$source = [Object]$script:$source,

[Parameter(Mandatory = $False, Position = 3)]

)

set-location -Path $script:path

# Show the pop-up and take the user selection
$choice = [System.Windows.MessageBox]:Shw("Remove the child directories","Source Clean Confirmat’o’","Ye’N’","Quest’on’)

# Create timer variable
[timespan]$script:cleanTime = Measure-Command {


# Take user input and action

if ($choice -i"q '"es') {

    # Set the variable to true for use in utility reporting logic
    $script:clean = $TRUE

    # Show the user confirmation of choice
    Show-MessageBoxDialog -Messa"e "You selected: "es"
    -WindowTit"e "Cleaning source fi"es"

    # Write the removed directories and their contents to a file for review
    get-childitem ($script:source) | Out-file RemovedFiles.txt
    remove-item $script:source -recurse -force

    # Show the user that actions have been completed
    Show-MessageBoxDialog -Messa"e "Directories removed. `n RemovedFiles.txt created. `n Exiting".."
    -WindowTit"e "Zip Utility exit"ng"

} elseif ($choice -i"q '"No') {

    # Show the user confirmation of choice
    Show-MessageBoxDialog -Messa"e "You selected: No...exit"ng"
-WindowTitle "Zip Utility exiting"

}

}

#

MAIN
#

# Write the title of this utility to the console Window
$host.ui.RAWUI.WindowTitle = "Zip Utility - Version: $Version"

# Create a new form
$UtilityForm = New-Object System.Windows.Forms.Form

# Define the size, title and background color
$UtilityForm.ClientSize = '350,150'
$UtilityForm.Text = "Utility Caller"
$UtilityForm.BackColor = '#ffffff'
$UtilityForm.FormBorderStyle = 'Fixed3D'
$UtilityForm.StartPosition = 'CenterScreen'
$UtilityForm.MaximizeBox = $false
$UtilityForm.MinimizeBox = $false

# Zip Utility Button
$ZipUtilBtn.BackColor = "#FF6600"
$ZipUtilBtn.text = "AutoZip Utility"
$ZipUtilBtn.width = 320
$ZipUtilBtn.height = 25
$ZipUtilBtn.location = New-Object System.Drawing.Point(10, 30)
$ZipUtilBtn.Font = 'Consolas,8'
$ZipUtilBtn.ForeColor = "#ffffff"

$UtilityForm.Controls.Add($ZipUtilBtn)

# Cancel Button
$cancelBtn.BackColor = "#ffffff"
$cancelBtn.text = "Exit"
$cancelBtn.width = 320
$cancelBtn.height = 25
$cancelBtn.location = New-Object System.Drawing.Point(10, 100)
$cancelBtn.Font = 'Consolas,10'
$cancelBtn.ForeColor = "#000"


$UtilityForm.CancelButton = $cancelBtn

$UtilityForm.Controls.Add($cancelBtn)

# Custom Button Actions to run the other Utils

$ZipUtilBtn.Add_Click({
    AutoZip
    SourceClean
})

# Display the form

[void]$UtilityForm.ShowDialog()

if (-not $cancelBtn)
    # Report execution times to user
    if ($script:clean)
        if (($script:zipTime -lt 1) -and ($script:cleanTime -lt 1)) {
            # Format variables for output in function call
            $zipMilli = $script:ziptime.Milliseconds
$cleanMilli = $script:cleanTime.Milliseconds

Show-MessageBoxDialog -Message "Utility took $zipMilli milliseconds to create zip packages'\nUtility took $cleanMilli milliseconds to clean-up source data"
-WindowTitle "Utility Execution Time"

} else {

$zipSecs = $script:ziptime.seconds
$cleanSecs = $script:cleanTime.seconds

Show-MessageBoxDialog -Message "Utility took $zipSecs seconds to create zip packages'\nUtility took $cleanSecs seconds to clean-up source data"
-WindowTitle "Utility Execution Time"

} }

} elseif (!$($script:clean)) {

if (($($script:zipTime -lt 1)) {

# Format variables for output in function call
$zipMilli = $script:ziptime.Milliseconds
Show-MessageBoxDialog -Message "Utility took $zipMilli milliseconds to create zip packages"

-WindowTitle "Utility Execution Time"

} else {

$zipSecs = $script:ziptime.seconds

Show-MessageBoxDialog -Message "Utility took $zipSecs seconds to create zip packages"

-WindowTitle "Utility Execution Time"

}

}

}

Show-MessageBoxDialog -Message "Exiting utility without executing"

-WindowTitle "Exit"

start-sleep 1

exit